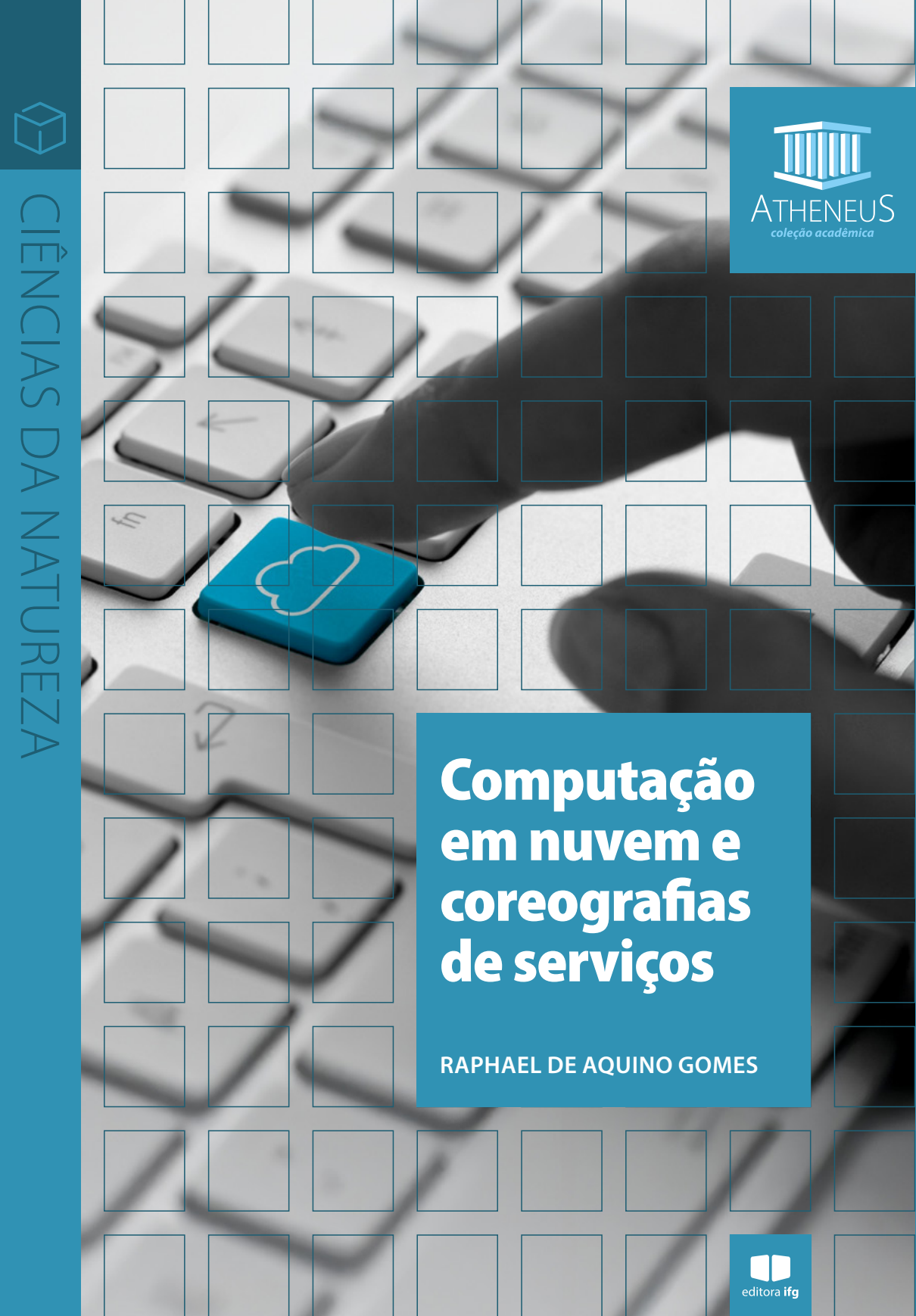


CIÊNCIAS DA NATUREZA

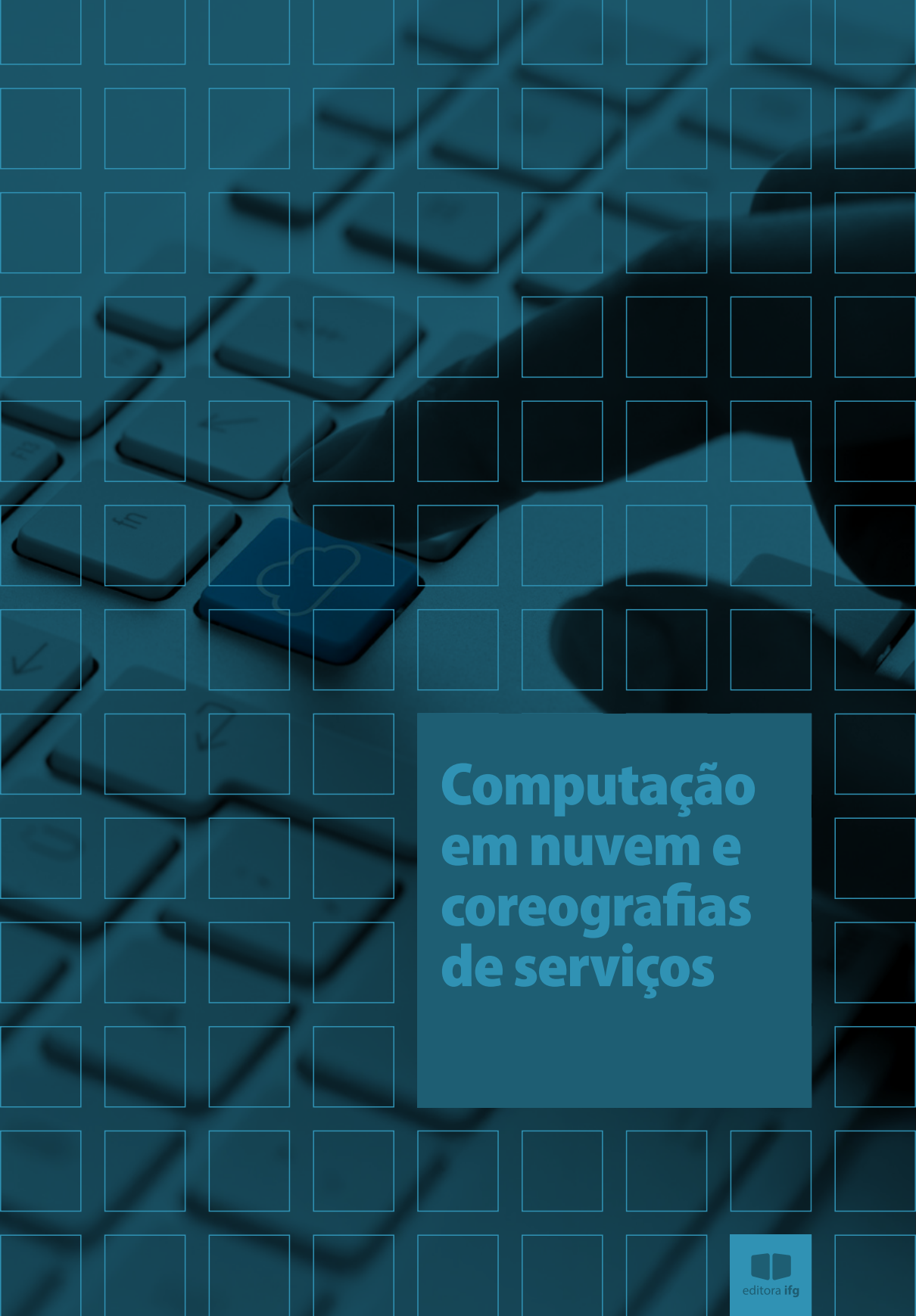
ATHENEUS
coleção acadêmica

Computação em nuvem e coreografias de serviços

RAPHAEL DE AQUINO GOMES

editora ifg

As infraestruturas digitais atuais dependem cada vez mais de serviços distribuídos capazes de atender a demandas complexas de integração, segurança e desempenho. Este livro apresenta uma proposta original para lidar com esses desafios: a representação de múltiplas coreografias com restrições não funcionais em ambientes de nuvem híbridos. Ao longo dos capítulos, o leitor encontra uma exposição clara dos fundamentos que sustentam esse campo e uma metodologia que alia consistência teórica e aplicabilidade prática. Mais do que reunir conceitos, a obra amplia o debate sobre sistemas distribuídos ao oferecer novas formas de compreender e enfrentar questões críticas. Destinada a estudantes, pesquisadores e profissionais, fornece subsídios para o desenvolvimento de soluções inovadoras e aponta caminhos concretos para o avanço tecnológico, constituindo-se como referência no âmbito da discussão contemporânea sobre arquiteturas orientadas a serviços.



Computação em nuvem e coreografias de serviços

editora ifg

ATHENEUS
coleção acadêmica

CIÊNCIAS DA NATUREZA

Computação em nuvem e coreografias de serviços

RAPHAEL DE AQUINO GOMES

editora ifg

ISBN 978-65-83687-01-2

G633c	Gomes, Raphael de Aquino Computação em nuvem e coreografias de serviços / Raphael de Aquino Gomes. -- Goiânia: Editora IFG, 2025. 260 p.: il. – (Coleção Atheneus) ISBN 978-65-83687-01-2 (impresso) ISBN 978-65-83687-00-5 (digital) 1. Computação em nuvem. 2. Coreografia de serviços – computação. 3. Modelagem de recursos. I. Gomes, Raphael de Aquino. II. Título. III. Coleção. CDD 004.6

Catalogação na publicação:
Maria Aparecida Andrade de Oliveira Tsu – Bibliotecária-documentalista – CRB-1/1604

Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Editora IFG

Avenida C-198, Qd. 500, Jardim América
Goiânia/GO | CEP 74270–040
(62) 3612-2251
editora@ifg.edu.br

Sumário

Prefácio 7

FÁBIO MOREIRA COSTA

Introdução 11

A necessidade de uma estratégia adequada para alocação de recursos 14

1. Conceitos fundamentais na implantação de coreografias de serviços 21

Terminologia adotada no livro 22

Computação em nuvem 25

Coreografia de serviços 40

Restrições sobre serviços 52

Exemplo de cenário 56

2. Abordagens para implantação de composições de serviços 65

Modelagem de coreografias de serviços 67

Gerenciamento de recursos 73

Implantação de composições de serviços 83

3. Representação de múltiplas coreografias de serviços com restrições não funcionais 93

Modelando uma coreografia com restrições 94

Representação de múltiplas coreografias e restrições 115

4. Modelagem de recursos 131

Tipos de recurso 132

Geração de tipos canônicos de recurso 136

Exemplo de geração do modelo canônico de recursos 139

5. Síntese de recursos **143**

O problema de síntese de recursos 144

Estimativa de recursos 152

Seleção de recursos 158

Casos de insucesso da síntese de recursos 176

6. Arquitetura e implementação **181**

Arquitetura 182

Cenário de uso da arquitetura 186

Implementação da arquitetura 188

7. Desdobramentos da abordagem desenvolvida **209**

Maior nível de abstração na avaliação de restrições não funcionais 209

Priorização de restrições não funcionais 211

Inclusão do usuário no *loop* 212

Aperfeiçoamento da avaliação de restrições de QoS 213

Aperfeiçoamento do modelo de recursos 217

Análise dos efeitos do compartilhamento de recursos 219

Adaptação dinâmica 221

Expansão da abordagem para outros componentes da pilha de software 223

Descentralização da arquitetura 224

Adequação da abordagem para o estilo arquitetural de microsserviços 225

Considerações finais **227**

Referências **233**

Prefácio

Esta obra, escrita pelo professor Raphael de Aquino Gomes, surge quando a computação em nuvem e a arquitetura orientada a serviços se consolidam como pilares fundamentais para o desenvolvimento de sistemas distribuídos modernos. Aborda, de forma competente, a complexidade da implantação de coreografias de serviços em ambientes de nuvem, propondo soluções para desafios que vão desde a modelagem de serviços até a alocação eficiente de recursos, consideradas as restrições não funcionais e a heterogeneidade de provedores de nuvem.

O livro se destaca por sua abordagem prática e teórica equilibrada, oferecendo ao leitor uma visão abrangente sobre como automatizar e otimizar o gerenciamento de recursos em ambientes de nuvem híbridos. Raphael Gomes não apenas discute os conceitos fundamentais da computação em nuvem e das coreografias de serviços, mas também propõe uma metodologia inovadora para a representação de múltiplas coreografias com restrições não funcionais, o que ainda não havia sido explorado de forma tão sistemática, abrindo caminho para novas investigações e aplicações em diferentes contextos. Trata-se, portanto, de uma leitura altamente recomendada para pesquisadores, profissionais e estudantes que desejam aprofundar seus conhecimentos na temática em discussão.

Ambientes de computação em nuvem têm como uma de suas principais utilidades a hospedagem de aplicações distribuídas que provêm de diferentes clientes e são tipicamente construídas na forma de composições de serviços Web. A alocação de recursos para hospedagem dos serviços que compõem uma aplicação é feita de

forma dinâmica e elástica, de modo que o cliente seja cobrado apenas pela quantidade de recursos efetivamente utilizada. Nesse contexto, um problema fundamental consiste na escolha do melhor tipo de recurso na nuvem para hospedar cada serviço da aplicação, considerando os requisitos impostos por serviço e o uso eficiente dos recursos. O problema torna-se interessante quando se considera um ambiente de nuvem híbrida, com vários provedores de nuvem que podem ser considerados nessa escolha, além de, possivelmente, uma nuvem privada pertencente ao próprio cliente. Isso permite explorar um espaço de busca muito maior para obter alocações de recursos mais vantajosas, porém suscita o *trade-off* entre alocação eficiente de recursos e introdução de latência na aplicação, uma vez que diferentes serviços da mesma aplicação podem ser alocados em nuvens diferentes.

Na pesquisa que deu origem a este livro, o autor vai um passo além na solução desse problema, ao explorar a constatação de que diversas aplicações podem fazer uso dos mesmos serviços, o que, por sua vez, motiva o compartilhamento de serviços entre elas (opondo-se à ideia já obsoleta de alocar instâncias separadas do mesmo serviço para cada aplicação). Considere, como exemplo disso, um serviço de conversão de moedas, útil em qualquer aplicação que lide com finanças internacionais. O compartilhamento permite obter taxas mais altas de utilização dos serviços, já que uma única aplicação pode não conseguir aproveitar toda a capacidade do recurso em que um serviço estiver alocado, deixando capacidade ociosa, que pode ser utilizada por outras aplicações. Além disso, o compartilhamento torna viável a alocação de serviços em recursos com maior capacidade, nos quais o custo por unidade de tarifação é menor. Como resultado, pode-se alocar um maior número de aplicações empregando uma menor capacidade total de recursos e com um menor custo por unidade, podendo-se ainda revisar a alocação de forma adaptativa à medida que os requisitos de recursos das

aplicações variam. Isso requer, contudo, que a seleção do recurso apropriado para hospedar um serviço compartilhado respeite o limite de capacidade do recurso, ou seja, que a carga agregada de todas as aplicações que usam o serviço não ultrapasse sua capacidade. A seleção também deve observar o nível de qualidade de serviço exigido por aplicação (por exemplo, tempo de resposta).

Atento a toda essa problemática, Raphael Gomes descreve a arquitetura e a implementação de um motor de seleção e alocação compartilhada de recursos em nuvens híbridas para aplicações distribuídas baseadas no modelo de coreografia de serviços. Desse modo, a seleção dos recursos alocados considera os requisitos de qualidade de serviço das aplicações, com a capacidade dos recursos e a existência de outras aplicações que competem pelo uso dos mesmos serviços. Nesse sentido, resultados experimentais demonstram uma considerável economia de recursos, por meio da maximização do uso da nuvem privada e da redução da quantidade de recursos alocados na nuvem pública, ao mesmo tempo que são atendidos os requisitos de qualidade de serviço das aplicações.

Esta obra tem, assim, o potencial de fazer repercutir a pesquisa por ela veiculada, habilitando cenários do tipo ganha-ganha entre provedores de nuvem e seus clientes. Se, por um lado, os provedores de nuvem, especialmente aqueles que empregam os modelos de plataforma como serviço (PaaS) e software como serviço (SaaS), podem fazer um uso mais racional dos recursos subjacentes para executar as mesmas aplicações, inclusive contribuindo para atingir objetivos de natureza ambiental ao proporcionar a redução do consumo total de energia; por outro, os clientes desses provedores se beneficiam com a redução de custos para execução, com qualidade de serviço, de suas aplicações.

Ao abrir novas perspectivas de estudo – como a análise das inter-relações entre computação em nuvem, alocação elástica de recursos e integração de sistemas de gerenciamento –, esta obra se

revela leitura essencial para quem busca compreender os rumos da computação distribuída e aprofundar-se em soluções inovadoras para o gerenciamento de serviços em ambientes de nuvem.

Fábio Moreira Costa
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Introdução

Recentes avanços no desenvolvimento de sistemas de software evidenciaram uma mudança de paradigma, partindo de soluções convencionais (que se baseiam no uso de bibliotecas ligadas ao programa, com chamadas a funções por meio do compartilhamento de memória) em direção à adoção de aplicações baseadas em serviços, que são criadas usando implementações sob medida ou componentes previamente desenvolvidos. Nessa nova realidade, a possível funcionalidade limitada que um serviço isolado oferece nem sempre satisfaz as necessidades complexas das aplicações ou reflete de maneira apropriada seus emaranhados processos de negócio (Alonso *et al.*, 2004).

Nesse cenário, composições de serviços representam uma abordagem mais apropriada. Contudo, devido à extensa quantidade de serviços interconectados e à existência de muitos pontos de interação, manter um único ponto de coordenação não é uma boa estratégia. Em virtude disso, uma solução promissora é o uso de serviços coordenados e descentralizados, divididos por meio de coreografias.

Coreografias são composições de serviços que implementam processos de negócio distribuídos, visando reduzir as trocas de mensagens de controle e distribuir a lógica do negócio. Nesse modelo de composição, não há a necessidade de controladores centralizados, uma vez que cada serviço "sabe" quando executar suas operações e com quais outros serviços interagir (Barker; Walton; Robertson, 2009). Quando os serviços participantes executam seus papéis, é dito que a coreografia foi encenada (Howard *et al.*, 2006).

A criação de coreografias de serviços é guiada não somente pelas propriedades funcionais dos serviços e pelas dependências entre eles, mas também por restrições não funcionais, como qualidade de serviço (*Quality of Service* – QoS), regulamentações ou obrigações legais impostas sobre a execução dos serviços, propriedades associadas ao ambiente de execução etc. Essas restrições podem ser especificadas objetivando a execução ou a encenação da coreografia como um todo (fim-a-fim) ou a execução de serviços específicos.

As decisões tomadas antes de os serviços serem executados afetarão como as restrições serão avaliadas, uma vez que essas decisões indicarão o ambiente em que esses serviços serão implantados e, portanto, estabelecerão como a execução do serviço ocorrerá. Isso gera a necessidade de uma estratégia eficiente de provisionamento de recursos. Por estratégia eficiente, nesse caso, entende-se a seleção de um conjunto de recursos que satisfaça todas as restrições especificadas e favoreça algum critério que seja relevante do ponto de vista do implantador da coreografia, por exemplo, o custo de sua encenação.

O aperfeiçoamento da tecnologia da informação (TI) possibilitou que a implantação de componentes de software seja realizada extrapolando os recursos disponíveis na infraestrutura privada da organização, por intermédio do modelo de negócio conhecido como computação em nuvem [*Cloud Computing*] (Dillon; Wu; Chang, 2010; Zhang; Cheng; Boutaba, 2010). Esse modelo faz com que a computação passe a ser utilizada da mesma forma que serviços como eletricidade, água e telefonia (Buyya *et al.*, 2009), permitindo acesso sob demanda, por meio de rede, a um conjunto de recursos computacionais configuráveis, que podem ser rapidamente oferecidos e liberados com um mínimo de esforço, gerenciamento e interação por parte de seu provedor (Mell; Grance, 2011). Mais especificamente, destaca-se a categoria de infraestrutura como serviço (*Infrastructure as a Service* – IaaS), que oferece acesso ubíquo e sob demanda a um ilimitado conjunto de recursos com propriedades distintas.

Nesse cenário, estratégias de gerenciamento de recursos devem permitir a tradução das restrições não funcionais para parâmetros de infraestrutura e o seu mapeamento para tipos pré-definidos de máquinas virtuais (*Virtual Machines* – VMs), o que inclui características como capacidade de CPU, memória, armazenamento e largura de banda. Além dessa tradução, é preciso decidir onde alocar o tipo de VM selecionado, a fim de reduzir o atraso de comunicação entre os serviços, uma vez que esse atributo pode ter grande impacto no desempenho e em outros critérios de qualidade da composição (Huang; Shen, 2015).

No cenário considerado para implantação de coreografias de serviços, ilustrado na Figura 1, o usuário é responsável por todas as atividades relacionadas ao gerenciamento de recursos. A primeira é a especificação das coreografias de serviços, que inclui a indicação dos serviços que serão compostos e como a interação entre eles ocorre. Em complemento a isso, é considerada, também, a especificação de um conjunto de restrições não funcionais que devem ser satisfeitas na execução dos serviços, pois influenciam sua implantação.

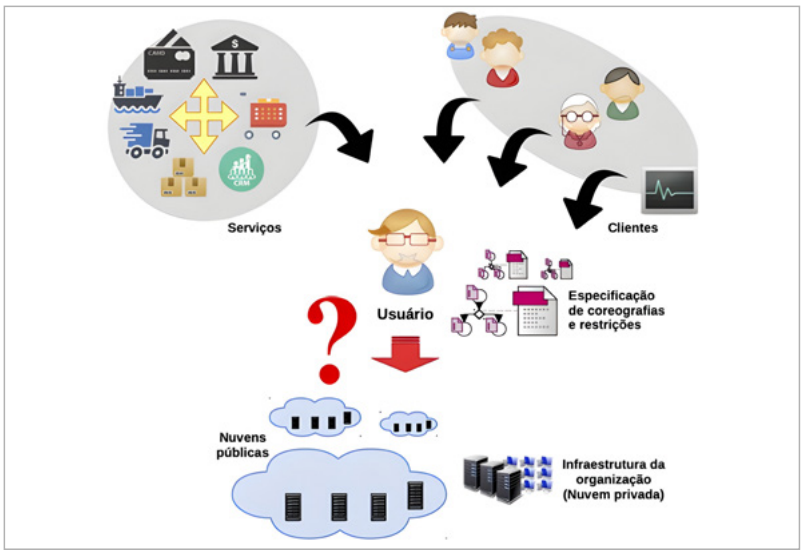


Figura 1 – Cenário de implantação de coreografias de serviços
Fonte: Elaboração própria.

Com base na especificação estabelecida e nas restrições que devem ser satisfeitas, é realizada a estimativa de recursos que consiste em determinar a capacidade necessária de recurso para implantar cada serviço. Em seguida, essa capacidade é utilizada para guiar a seleção dos tipos de VM mais apropriados para atender essa demanda. Essa seleção ocorre considerando um ambiente de nuvem híbrido, sobre o qual os recursos selecionados são alocados.

Durante a execução dos serviços, o usuário monitora atributos relacionados aos recursos, como nível de utilização; assim como obtém *feedback* dos clientes que utilizam os serviços, como nível de satisfação. Com base nos dados de monitoramento, adaptações são realizadas no provisionamento de recursos, com o objetivo de satisfazer as restrições.

Neste livro, é apresentada uma solução para automação das atividades relacionadas ao gerenciamento de recursos, após a definição dos serviços que compõem as coreografias e das restrições não funcionais associadas. Essa automação tem como objetivo permitir que essas atividades possam ser realizadas de maneira eficiente, sem interação direta do usuário.

A necessidade de uma estratégia adequada para alocação de recursos

Uma estratégia para o provisionamento eficiente de recursos é considerar que aplicações orientadas a serviços podem apresentar um conjunto comum de funcionalidades, o que torna desejável a criação de composições com compartilhamento de serviços. Esta necessidade torna a satisfação de restrições ainda mais desafiadora, principalmente devido à degradação de *Quality of Service* (QoS) em serviços compartilhados e à possível existência de múltiplas restrições sobre um mesmo serviço. Como exemplo, é possível utilizar

uma mesma métrica de QoS para restringir um determinado serviço que participa de duas ou mais coreografias usando diferentes valores-alvo em cada uma delas. Por esse motivo, a estimativa de recursos deve ser realizada de forma a decidir a capacidade mais adequada ao recurso que será usado para implantar cada serviço, satisfazendo restrições e balanceando, de maneira eficaz, a contribuição do serviço na satisfação de restrições fim-a-fim especificadas para cada coreografia da qual este participa.

A possibilidade de expressar requisitos de recursos com base nas necessidades dos serviços que eles deverão hospedar favorece um gerenciamento mais preciso da capacidade computacional disponível. Como consequência, torna-se possível alocar de forma mais precisa a capacidade necessária à coreografia, além de facilitar o redimensionamento dinâmico da medida alocada em face de mudanças.

Nesse cenário, o provisionamento de recursos não pode ser baseado em um único provedor de nuvem, uma vez que a habilidade de executar e gerenciar sistemas que usam múltiplos ambientes de nuvem permite explorar as peculiaridades de cada um deles e, consequentemente, otimizar o desempenho, a disponibilidade e o custo das aplicações (Ferry *et al.*, 2013a). Em adição a isso, o provisionamento deve considerar os recursos existentes na infraestrutura da própria organização e o redirecionamento de requisições para nuvens públicas apenas quando a infraestrutura privada estiver totalmente alocada (geralmente em períodos de grande demanda), o que é conhecido como *cloud bursting*.

Para efetivamente tirar vantagem da elasticidade disponível ao utilizar ambientes de nuvem, o mapeamento de serviços para recursos deve considerar o modelo de cobrança baseado em funções de utilidade empregado por provedores de nuvem (Zhang; Cheng; Boutaba, 2010) e encontrar um equilíbrio entre as restrições não funcionais e o custo. Essa tarefa constitui um grande desafio porque,

geralmente, um mesmo tipo de VM pode ser instanciado em diferentes localidades com custo variável, criando um vasto conjunto de opções. Delegar essa decisão (e as demais relacionadas ao gerenciamento de recursos) para a pessoa responsável por implantar as coreografias não é uma tarefa simples, pois essa é uma atividade suscetível a erros, dado o número de opções, e porque o gerenciamento manual de recursos é uma tarefa demorada e complexa (Eisa *et al.*, 2016). Apesar destas limitações, muitas organizações ainda realizam essas tarefas de forma empírica, tornando o processo moroso, propenso a erros e irreproduzível (Dolstra; Bravenboer; Visser, 2005). Diante disso, surge um problema relacionado ao gerenciamento de recursos para a implantação de coreografias de serviços com restrições não funcionais associadas: dado um conjunto de coreografias de serviços, com possíveis compartilhamentos de operações entre elas e restrições não funcionais associadas, deve-se decidir a configuração ideal de recursos para implantar cada serviço e mapear esses recursos para ambientes de nuvem apropriados. A estimativa, o provisionamento de recursos e a implantação de serviços devem ser realizados de maneira autônoma e eficiente.

Apesar da existência de muitos trabalhos que visam o gerenciamento de recursos em nuvem (Bouras *et al.*, 2020; Hummaida; Paton; Sakellariou, 2016; Manvi; Shyam, 2014; Singh; Chana, 2016), a implantação de composições de serviços em ambientes de nuvem com o mínimo de intervenção humana e levando em consideração aspectos não funcionais associados aos serviços não é explorada em todas suas dimensões. Porém, a estimativa e o provisionamento autônomo de recursos são uma necessidade cada vez mais eminente, pois essa funcionalidade está alinhada à filosofia que rege o modelo de nuvem, que tem por objetivo facilitar e tornar transparente a forma como os recursos são gerenciados.

Ao desenvolver a pesquisa que originou este livro, buscou-se propor uma solução que, em nível semelhante à abstração oferecida na alocação de recursos em ambientes de nuvem, abstrai, também, a estimativa e as demais etapas do provisionamento de recursos. Essa solução considera desafios adicionais relacionados à implantação de múltiplas composições de serviços, como a necessidade de satisfazer simultaneamente uma quantidade maior de restrições e a degradação de QoS causada pelo compartilhamento de serviços entre elas. Para tal, as questões que nortearam o estudo foram: a) como automatizar a implantação de múltiplas coreografias de serviços visando a satisfação de restrições com uso eficiente de recursos? b) quais fatores e métodos devem ser considerados para decidir (de maneira autônoma) a configuração de recursos adequada para implantar um conjunto de coreografias de serviços, dada sua especificação e restrições não funcionais associadas? c) como abstrair a estimativa e a seleção de recursos em um ambiente de nuvem híbrido com infraestrutura privada e múltiplos provedores de nuvem pública?

A investigação dessas questões é motivada por uma série de dificuldades encontradas na satisfação de restrições ao implantar composições de serviços. Por exemplo, uma característica importante da estratégia de gerenciamento de recursos é que ela deve garantir que haja capacidade computacional suficiente para atender a demanda sobre o serviço, satisfazendo as restrições requisitadas, ao mesmo tempo em que favorece o critério adotado para guiar o provisionamento (como redução do custo). Sem utilizar uma estratégia como a que é proposta, o mapeamento das restrições para recursos de infraestrutura é geralmente realizado com base em experiências prévias dos desenvolvedores, dificultando esta tarefa quando a aplicação é um produto inédito.

Tópicos discutidos neste livro

Este livro tem como principal objetivo discutir estratégias para a implantação automatizada e eficiente de múltiplas coreografias de serviços, sujeitas às restrições não funcionais associadas aos serviços e à sua encenação. Essa estratégia considera o compartilhamento de serviços entre as coreografias e se beneficia da multiplicidade de tipos de recursos disponíveis em um ambiente híbrido formado por uma nuvem privada e múltiplas públicas. Essas estratégias visam automatizar as atividades relacionadas à estimativa, à descoberta, à seleção e à alocação de recursos. A consecução desse objetivo geral se desdobra nos seguintes objetivos específicos:

- apresentar formas de modelagem de um conjunto de coreografias de serviços levando em consideração o compartilhamento de operações entre elas e restrições não funcionais associadas à implantação e ao funcionamento dos serviços e das coreografias;
- discutir mecanismos para a representação dos tipos disponíveis de recursos, de forma que seja possível tratar a heterogeneidade na representação de recursos e modelar, de maneira consistente, múltiplos provedores de nuvem. Também, deve ser viável o uso dessa representação para seleção de recursos sem comprometer o tempo necessário para realizar a implantação;
- investigar e propor mecanismos para estimar a capacidade adequada de recursos necessários para implantar um conjunto de coreografias, dadas suas estruturas e restrições;
- investigar e propor mecanismos para seleção de recursos em uma nuvem híbrida, tendo como base a capacidade estimada e as restrições previamente estabelecidas;

- definir uma arquitetura que permita a implantação eficiente e autônoma de múltiplas coreografias de serviços usando os mecanismos propostos; e
- difundir um protótipo funcional da arquitetura definida.

A principal contribuição desta obra é apresentar uma abordagem para abstrair, simplificar e automatizar decisões sobre gerenciamento de recursos virtualizados em ambientes de nuvem, e que satisfaça as restrições não funcionais na implantação de múltiplas coreografias de serviços. Dada a descrição em alto nível de uma ou mais coreografias de serviços e restrições associadas, a abordagem proposta estima e provisiona de maneira autônoma a configuração de recursos mais apropriada para satisfazer as restrições. O problema apresentado é modelado como um problema de otimização, e é apresentada uma solução que busca minimizar o custo financeiro da implantação das coreografias e a sobrecarga de comunicação entre os serviços. Tal solução se destaca por ser a primeira abordagem para implantação de múltiplas composições de serviços, considerando todas as atividades relacionadas ao gerenciamento de recursos, satisfazendo diferentes categorias de restrições não funcionais.

O livro apresenta uma solução para provisionamento de recursos levando em conta o compartilhamento de serviços e recursos. Espera-se que o uso dessa solução possibilite aos usuários implantar composições de serviços, aproveitando, de forma otimizada, o vasto potencial de provedores de nuvem e tipos de recursos, ao permitir que eles foquem nos aspectos do negócio enquanto delegam à solução os tecnicismos relacionados à alocação de recursos e implantação de serviços. Outras contribuições desta obra são descritas a seguir:

- propor um modelo de coreografia de serviços que ajuda no processo de identificação, categorização e especificação de integração de recursos em nuvem. A abordagem permite

encontrar os recursos virtuais necessários em um ambiente híbrido formado por uma nuvem privada, e múltiplos provedores de nuvem pública, por meio de um modelo de recursos gerado a partir do modelo da coreografia;

- propor um arcabouço para especificação de restrições sobre coreografias de serviço que facilita a inclusão de métricas de QoS, oferece uma linguagem para descrição de restrições e implementa a geração automática de filtros a serem aplicados na seleção de recursos;
- apresentar uma solução eficiente e coordenada para a seleção de recursos que contempla necessidades comerciais emergentes com o aumento no número de provedores de nuvem. Essa solução agrega um conjunto de estratégias para mapeamento de especificações de coreografias, determinando os recursos necessários para sua implantação e levando em consideração restrições impostas sobre seu funcionamento.

1

Conceitos fundamentais na implantação de coreografias de serviços

Este capítulo versa sobre alguns conceitos relevantes para a implantação de coreografias de serviços. Como forma de facilitar o entendimento e contextualizar os termos utilizados, inicialmente é apresentada a terminologia adotada neste livro. Essa terminologia foi apresentada em Gomes *et al.* (2016) e estendida aqui. Assim, o objetivo deste capítulo não é expor uma descrição profunda de todas as soluções existentes nas áreas discutidas, mas propiciar uma breve introdução a essa terminologia.

Entre os conceitos discutidos, inicialmente são apresentados as principais características e os desafios de sistemas baseados em nuvem. Em especial, são descritas as principais atividades relacionadas ao gerenciamento de recursos nesse tipo de ambiente. Em seguida, são abordados os tópicos de modelagem e implantação de coreografias de serviços, em que serão discutidas algumas definições sobre restrições não funcionais na implantação de serviços, destacando as particularidades dessas definições no contexto de composições de serviços.

Por fim, é descrito um exemplo de cenário que visa destacar a necessidade de expressar restrições não funcionais na implantação

de coreografias de serviços, e de considerar o compartilhamento de serviços entre elas. Com base nesse exemplo, são apresentados alguns dos principais desafios encontrados acerca do problema discutido neste livro.

Terminologia adotada no livro

A terminologia adotada neste livro usa a definição padrão de *aplicação*, que consiste em um programa de computador desenvolvido com um propósito específico, por exemplo, relacionada ao agendamento de consultas médicas no sistema público de saúde. Uma aplicação pode ser implementada como um único serviço ou ser uma composição de dois ou mais deles. Para *serviço*, é adotada a mesma definição de serviço Web (Austin *et al.*, 2004), segundo a qual um serviço é uma entidade de software autocontida, cujas interfaces e ligações são definidas, descritas e localizadas por artefatos que seguem especificações como *Simple Object Access Protocol* (SOAP) (Gudgin *et al.*, 2007) e *Representational State Transfer* (REST) (Fielding, 2000).

Neste livro assume-se que todos os serviços são *stateless*, seguindo o padrão arquitetural REST. Esta decisão foi tomada visando vantagens como (1) maior confiabilidade, uma vez que falhas são mais facilmente recuperadas; (2) maior escalabilidade, dado que não há necessidade de manter o estado das solicitações, permitindo que os serviços não necessitem de recursos além daqueles relacionados ao processamento das requisições; (3) requisições autocontidas, pois a mensagem da chamada a uma operação contém todas as informações necessárias para processar a solicitação; e (4) simplificar o desenvolvimento da abordagem.

O custo associado à utilização dos serviços, como pagamento de licenças ou aluguel de software, é negligenciado, de forma que a natureza dos serviços utilizados e a quantidade de instâncias de

serviço envolvidas não tem impacto sobre o custo de execução de uma composição. Outra simplificação adotada foi abstrair a arquitetura de um serviço como sendo uma única camada cujos elementos não são explorados individualmente, apesar da possibilidade da implantação de serviços ser realizada de maneira mais precisa, caso o gerenciamento de recursos seja feito separadamente para cada camada da arquitetura (Bi *et al.*, 2010; Grozev; Buyya, 2015; Strauch *et al.*, 2012).

Apesar de o termo "serviço" ser amplamente usado em sistemas baseados em nuvem para se referir a qualquer componente virtualizado (inclusive de infraestrutura), esse significado não é empregado neste livro para evitar qualquer ambiguidade entre serviços de nuvem e serviços (Web) que são elementos que compõem coreografias. Uma vez que o escopo desta obra se limita ao modelo de infraestrutura como serviço (*Infrastructure as a Service* – IaaS), o termo *recurso* é usado para referir aos serviços de nuvem que representam recursos virtualizados (VMs) em provedores de IaaS, e o termo "serviço" é usado para referir aos elementos que compõem coreografias; salvo algumas exceções em que o uso do termo *serviço de nuvem* é explicitamente adotado para evitar mal entendimento.

Um serviço (Web) implementa uma ou mais operações, devendo ser capaz de interagir com outros serviços ou aplicações pela troca de mensagens utilizando os protocolos de comunicação padrão atualmente disponíveis na Internet. Uma *operação*, por sua vez, define alguma tarefa executada pelo serviço, que requer uma quantidade de poder computacional para ser processada. Cada operação se refere a um *papel* adotado na definição de coreografias, o qual constitui uma função cuja responsabilidade é assumida pelo serviço que a executa e que define as ações e a interação desse serviço com os demais na composição.

Geralmente, os serviços que compõem uma coreografia podem ser descritos de maneira abstrata, devido ao uso do papel

desempenhado pelo serviço na interação. Esses papéis podem, então, ser atribuídos a responsáveis usando entidades concretas, isto é, identificando uma implementação compatível para cada serviço. A abstração de serviços é particularmente importante em ambientes com múltiplas nuvens, uma vez que a especificação, usando serviços concretos, pode restringir de maneira excessiva a definição da composição, e porque certas implementações podem ser específicas para um provedor de nuvem ou tecnologia. Apesar disso, devido à alta complexidade de propor uma abordagem que também contemple a seleção de serviços, no contexto deste livro, assume-se que a especificação de coreografias é realizada usando entidades concretas.

Também, se assume que informações sobre serviços são gerenciadas por um ou mais *catálogos de serviços*, comumente conhecidos como *brokers* de serviços (Liu; Ngu; Zeng, 2004). Cada catálogo classifica serviços em duas categorias:

- *Serviço implantável*: descreve a implementação de um serviço. Para cada serviço nessa categoria há um conjunto de informações necessárias à implantação de uma instância desse serviço (por exemplo, seu URI, protocolo de comunicação usado etc.) e atributos das operações por ele implementadas (como papéis executados, número de instruções do programa etc.).
- *Serviço legado*: descreve serviços de terceiros que já foram previamente implantados, e para os quais não há possibilidade de interferência. Nesse caso, assume-se que o catálogo disponibiliza informações sobre as propriedades não funcionais garantidas pelo serviço, assim como dados sobre seu ambiente de execução.

Usuário refere-se às pessoas responsáveis pela composição da aplicação e sua manutenção, o que inclui o gerenciamento de recursos. Essas atividades devem ser realizadas de forma a fazer com que

os requisitos funcionais e não funcionais para uma classe de clientes sejam satisfeitos durante a execução da aplicação. Outra função realizada pelo usuário é decidir quais alternativas devem ser adotadas para adaptação da alocação de recursos, com base em mudanças nas condições do sistema e nas expectativas dos clientes. O *cliente*, ou usuário final, é uma entidade que interage com essa aplicação.

Por último, tem-se um *sistema* que é um conjunto de componentes interdependentes que interagem entre si, formando um conjunto integrado. Este termo é aqui usado para referir ao conjunto das aplicações gerenciadas, juntamente com os componentes que constituem a solução que será apresentada.

Computação em nuvem

Computação em nuvem (*Cloud Computing*) é um modelo que permite acesso a recursos computacionais sob medida e sob demanda, de maneira similar a recursos como água e eletricidade, o que leva esse modelo a ser chamado, também, de computação utilitária (*Utility Computing*). Este paradigma possibilita acesso, por intermédio de rede, a um conjunto de recursos computacionais (rede, armazenamento, processamento, plataformas de desenvolvimento e aplicações), que podem ser rapidamente aprovisionados e liberados com um mínimo esforço de gerenciamento e interação por parte do provedor (Mell; Grance, 2011).

Uma nuvem também pode ser definida como um sistema paralelo e distribuído que consiste em uma coleção de componentes virtualizados e interconectados, que são provisionados dinamicamente e apresentados como um ou mais recursos computacionais unificados (Buyya *et al.*, 2009).

Novas organizações sem o alto capital necessário para criar sua própria infraestrutura utilizam este modelo para ter acesso a um vasto conjunto de recursos. Além disso, empresas como

Amazon (2022a) logo perceberam a vantagem de oferecer o seu poder de computação excedente na forma de serviços em nuvem, pois com isso poderiam continuar a superdimensionar sua infraestrutura para atender aos picos de demanda, enquanto proveem sua capacidade excedente para os usuários na forma de recursos virtualizados.

Além desses dois cenários, uma alternativa bastante promissora é a utilização de uma abordagem híbrida, na qual a organização possui sua própria infraestrutura, mas utiliza recursos de terceiros em períodos de grande demanda, o que é conhecido como *cloud bursting* (Javadi; Abawajy; Buyya, 2012; Kim *et al.*, 2011). Nesse caso, surge a necessidade de maximizar o uso dos recursos privados por questões de segurança e economia.

Entre os principais fatores que contribuíram para a popularização de computação em nuvem está a diminuição de custo de hardware, aliada a um aumento no poder de computação e capacidade de armazenamento. Além disso, paralelamente há um crescimento exponencial no volume de dados e uma ampla adoção de aplicações e serviços na Web. Nesta nova realidade, as corporações se esforçam para disponibilizar plataformas de nuvem mais poderosas, confiáveis e financeiramente atrativas devido à ascensão de novos serviços que apresentam requisitos rigorosos. Dessa forma, há um crescente interesse na oferta de propriedades não funcionais. Essa necessidade existe porque a utilização de um software é determinada não só por suas características funcionais, mas também por suas propriedades não funcionais, sendo que algumas funcionalidades podem ser inúteis caso certos atributos não funcionais não forem oferecidos (Chung; Leite, 2009).

Apesar da importância dos aspectos não funcionais, atualmente há garantias bem reduzidas sobre atributos dessa categoria em serviços em nuvem, sendo que o foco maior ocorre em desempenho e disponibilidade (Baset, 2012). A oferta de qualidade de serviço (*Quality of Service* – QoS) em computação em nuvem é um dos principais

desafios dessa área (Blair *et al*., 2011). Essa necessidade se dá principalmente porque serviços providos em nuvem devem ser altamente confiáveis, escaláveis e autônomos. O próprio modelo de computação em nuvem já prevê que a alocação de recursos oferecida nesse ambiente pode ser dinamicamente redimensionada para se ajustar a cargas variáveis de utilização, o que permite obter uma utilização ótima dos recursos físicos (Vaquero *et al.*, 2008).

A seguir, são apresentadas as principais características de ambientes de computação em nuvem.

Características

As principais características que definem computação em nuvem são apresentadas a seguir:

- *Autoatendimento sob demanda*: o usuário pode obter acesso ao conjunto de recursos no momento que necessitar, sem a necessidade de interação humana com o provedor. Isto é feito por meio de interfaces disponíveis na Web ou de APIs que podem ser utilizadas programaticamente.
- *Independência de localização*: os recursos são acessados por meio da Internet, utilizando diferentes plataformas (como desktop, tablet e smartphone) de forma ubíqua.
- *Abstração de recursos*: os recursos providos são utilizados simultaneamente por múltiplos usuários por meio de virtualização, com diferentes recursos físicos e virtuais atribuídos e retribuídos de acordo com a demanda dos usuários. O resultado é que recursos físicos se tornam "invisíveis" para os usuários, que não possuem controle ou conhecimento da localização, formação e origem desses recursos. Isso diminui os riscos de negócio para os usuários desse paradigma, além de eliminar seus gastos com manutenção de recursos.

- *Elasticidade*: o usuário tem acesso à quantidade de recursos que precisar, sem a necessidade de que contratos rigorosos sejam preestabelecidos, podendo liberá-los assim que terminar o uso. A elasticidade garante a escalabilidade, o que significa que a alocação de recursos pode ser redimensionada ascendentemente para demandas de pico, e descendentemente para demandas mais leves.
- *Serviço medido*: a infraestrutura de nuvem é capaz de usar mecanismos adequados para medir o uso dos recursos para cada usuário, que paga somente aquilo que consome.

Computação em nuvem é considerada uma generalização de computação em grade (*Grid Computing*), conforme Camargo *et al.* (2004), Foster *et al.* (2003) e Foster, Kesselman e Tuecke (2001). Uma grade computacional pode ser definida como uma infraestrutura de software capaz de interligar e gerenciar diversos recursos computacionais (capacidade de processamento, dispositivos de armazenamento, instrumentos científicos etc.), possivelmente distribuídos por uma grande área geográfica, de maneira a oferecer ao usuário acesso transparente a tais recursos, independentemente da sua localização (Goldchleger, 2004). Os paradigmas de grade e nuvem são similares no sentido de utilizarem recursos distribuídos para oferecer serviços. Contudo, computação em nuvem pode ser considerada um avanço por facilitar o compartilhamento e provisionamento dinâmico de recursos (Zhang; Cheng; Boutaba, 2010).

Além de computação em grade, computação em nuvem tem como base outras tecnologias:

- *Web 2.0*: consiste no segundo estágio de desenvolvimento da *World Wide Web*, caracterizado principalmente pelo aumento na quantidade de páginas Web dinâmicas em vez de páginas estáticas, maior presença de conteúdo gerado pelos clientes e crescimento de mídias sociais. Essa evolução permite o oferecimento de serviços pela Internet,

disponibilizando via aplicações Web que antes existiam somente como programas para desktop (O'Reilly, 2005).

- *Virtualização*: é uma técnica que possibilita a representação baseada em software (ou virtual) de um elemento físico, como servidores, armazenamento e rede. Em computação em nuvem este conceito possibilita que recursos virtuais sejam usados da mesma forma que os recursos reais equivalentes, permitindo o particionamento de um recurso físico para vários usuários simultaneamente por meio do uso de máquinas virtuais.
- *Conteinerização*: contêineres surgiram como uma alternativa mais leve à virtualização usando máquinas virtuais. Conteinerização é uma técnica de virtualização no nível do sistema operacional usada para implantar e executar aplicações distribuídas sem a necessidade de criar uma máquina virtual inteira para cada aplicação. Em vez disso, múltiplos sistemas isolados, chamados contêineres, são executados em uma máquina, compartilhando um único *kernel* (Pahl, 2015). Com isso, contêineres são mais eficientes que máquinas virtuais.
- *Computação autonômica*: um sistema computacional autonômico controla as funcionalidades de sistemas e aplicações com o mínimo de intervenção humana, da mesma forma que o sistema nervoso autônomo regula o sistema corporal, sem intervenção do indivíduo. O objetivo da computação autonômica é criar sistemas com maior grau de independência, capazes de implementar funcionalidades complexas, enquanto mantém esta complexidade transparente ao usuário. Como resultado, profissionais de TI podem focar em tarefas que agregam mais valor ao negócio (An architectural [...], 2005).

A Figura 2 apresenta a relação de computação em nuvem com estes domínios. A Web 2.0 cobre quase todo o espectro de aplicações orientadas a serviços. Com base nela, ou seja, sobrepondo esta tecnologia está a computação em nuvem que geralmente oferece larga escala. Essas tecnologias, por sua vez, são baseadas em computação autonômica e virtualização, que podem ser utilizadas para desenvolver soluções orientadas a aplicações ou a serviços, com escala que dependerá da solução desenvolvida. Por outro lado, a conteinerização também representa uma tecnologia baseada em virtualização, mas constitui uma alternativa de maior escala. Assim como computação em nuvem, computação em grade sobrepõe, ou seja, baseia-se em todas as demais tecnologias, mas é voltada para aplicações convencionais. Subpondo quase totalmente as tecnologias descritas, estão as tecnologias de sistemas distribuídos em geral.

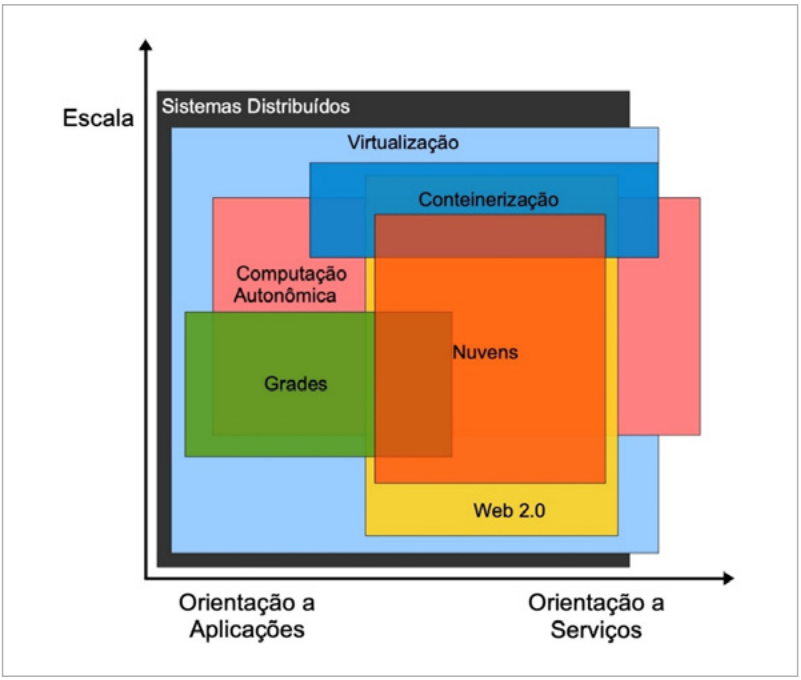


Figura 2 – Relação entre computação em nuvem e outras tecnologias
Fonte: Adaptada de Foster *et al.* (2008-2009).

Categorias de serviços

Mais comumente, um provedor de nuvem oferece recursos de hardware (processamento, armazenamento, rede etc.), mas é possível encontrar outros tipos de recurso, como serviços de software, aplicações, APIs e ferramentas de desenvolvimento. De acordo com a categoria de serviço que oferece, os provedores de nuvem são geralmente classificados em três categorias principais:

- *Infraestrutura como serviço (Infrastructure as a Service – IaaS)*: usuários usam diretamente a infraestrutura provida. Virtualização e contêinerização são extensivamente utilizadas para integrar/decompor recursos físicos para atender a demanda dos usuários. Ex.: Amazon Web Services (AWS) (Amazon, 2022b) e Nimbus (Keahey, 2009).
- *Plataforma como serviço (Platform as a Service – PaaS)*: permite aos usuários desenvolverem serviços e aplicações que irão executar na nuvem, utilizando linguagens de programação, bibliotecas, serviços e ferramentas oferecidas pelo provedor. O usuário não tem que gerenciar elementos da infraestrutura de nuvem, como rede, servidores, sistemas operacionais, ou armazenamento, mas tem controle sobre as aplicações implantadas e a configuração do ambiente de hospedagem (Mell; Grance, 2011). Ex.: Google App Engine (Google, 2022a) e Microsoft Azure[1] (Microsoft, 2022a).
- *Software como serviço (Software as a Service – SaaS)*: o fornecedor do serviço hospeda as aplicações, de modo que não é preciso instalá-las, gerenciá-las, ou comprar hardware para sua execução. Com isso, as ações que devem ser realizadas

1 Apesar de inicialmente o Microsoft Azure (Windows Azure) ter sido desenvolvido visando à PaaS, atualmente IaaS é um dos principais focos desse provedor.

pelo usuário se limitam a acessar o serviço e proceder com sua utilização. Ex.: Google Drive e SAP Business ByDesign (SAP, 2022).

Esta categorização é geralmente construída seguindo uma arquitetura de camadas, conforme indicado na Figura 3. Diferentes categorias podem ser oferecidas por um mesmo provedor de nuvem e, como pode ser visto, é possível ao usuário o acesso a qualquer uma das camadas. Porém, é importante ressaltar que o oferecimento conjunto de múltiplas camadas dessa arquitetura por um mesmo provedor de nuvem não representa um padrão, uma vez que, conforme exemplificado acima, há provedores que são específicos de uma certa categoria.

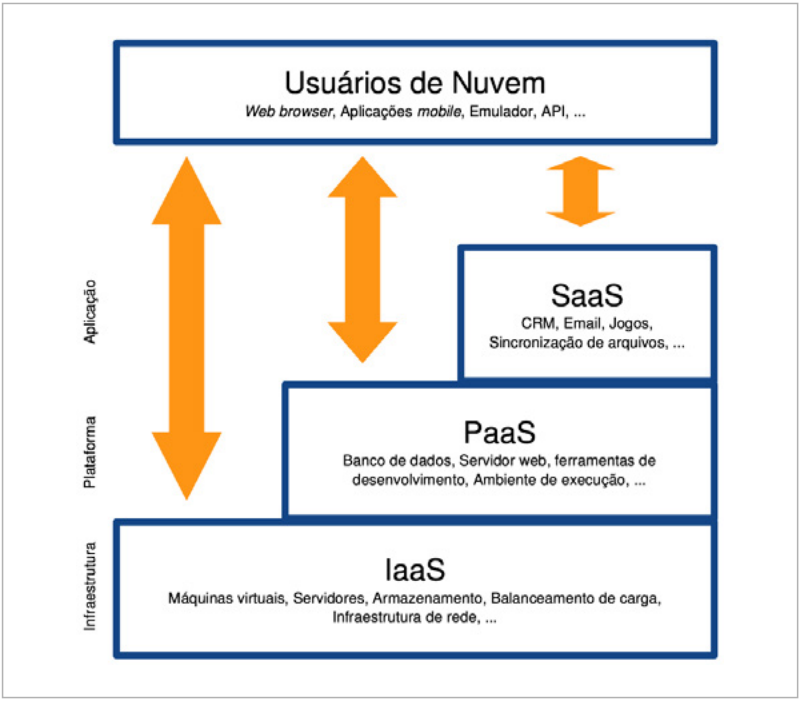


Figura 3 – Categorização e relacionamento dos serviços em nuvem
Fonte: Adaptada de Foster *et al.* (2008-2009).

Com a popularização de computação em nuvem, outras categorias de serviço passaram a ser oferecidas na Web utilizando esse modelo de negócio. Com isso, atualmente, o termo mais comum para representar de forma genérica as categorias de serviços em nuvem é: *Tudo como Serviço* (*Anything as a Service* – XaaS). Além dos serviços já citados, este termo engloba, também, a provisão de comunicação, infraestrutura de rede, monitoramento e objetos como serviços (Alam; Chowdhury; Noll, 2010; Parwekar, 2011).

Ao mover uma aplicação para ambientes de nuvem há muitas questões a considerar. Por exemplo, algumas organizações estão mais interessadas em reduzir o custo de execução da aplicação, enquanto outras podem preferir alta confiabilidade e segurança. Consequentemente, existem diferentes modelos de implantação de nuvem, cada um com seus próprios benefícios e suas desvantagens:

- *Nuvem privada (private cloud)*: a infraestrutura é mantida e gerenciada pela própria organização, visando maximizar o uso de seus recursos e evitar problemas, como transferência de dados e segurança.
- *Nuvem pública (public cloud)*: este é o modelo dominante, no qual a infraestrutura de uma organização é utilizada pelo público em geral, sendo que as políticas de uso e cobrança são definidas pelo provedor de recursos.
- *Nuvem híbrida (hybrid cloud)*: é a combinação dos dois modelos anteriores, em que uma organização usa sua própria infraestrutura, geralmente para armazenamento de dados e execução de serviços essenciais, e usa os recursos de uma nuvem pública para aumentar sua produtividade ou obter um serviço que não possui.

O trabalho apresentado neste livro tem como escopo nuvens híbridas, com múltiplos provedores de nuvens públicas, focando na camada de IaaS.

Gerenciamento de recursos em nuvem

O objetivo desta seção é prover uma análise das atividades desempenhadas pelo usuário do ambiente de nuvem no controle dos recursos virtualizados. São investigadas, também, as principais ferramentas disponíveis para realizá-las. Embora as mesmas atividades façam sentido do ponto de vista do provedor de nuvem, ao gerenciar os recursos físicos, não são levados em consideração os aspectos relevantes somente nesse contexto, como a economia de energia (Buyya; Beloglazov; Abawajy, 2010), uma vez que o objetivo é isolar o ponto de vista do usuário de serviços em nuvem.

Conforme ilustrado na Figura 4, o gerenciamento de recursos envolve atividades desde a estimativa até o monitoramento de recursos, sendo que as principais atividades nessa sequência geralmente são tratadas como sendo uma única atividade, identificada como provisionamento de recursos.

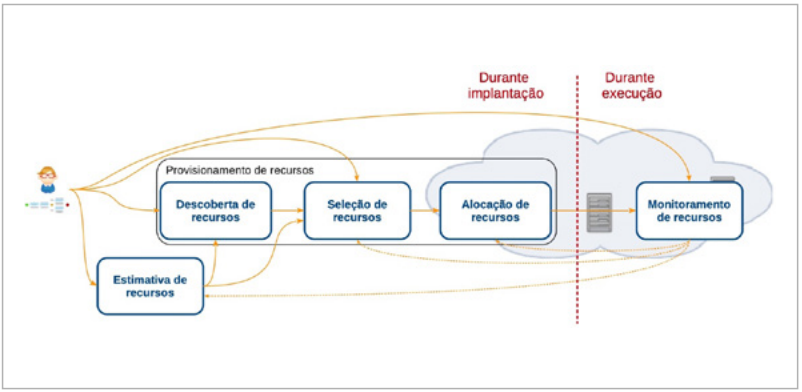


Figura 4 – Atividades envolvidas no gerenciamento de recursos em nuvem e como são acionadas
Fonte: Elaboração própria.

A primeira atividade, estimativa de recursos, visa identificar a capacidade ideal de recursos e o custo esperado ao realizar a implantação da aplicação. Essa atividade deve levar em consideração as características funcionais da aplicação, assim como as restrições não funcionais impostas a ela. As ferramentas atualmente existentes só auxiliam no cálculo do custo estimado em um provedor específico, por exemplo, Microsoft Azure Pricing Calculator (Microsoft, 2022b) ou AWS Total Cost of Ownership (TCO) Calculator (Amazon, 2022c), mas a capacidade e quantidade de recursos necessários devem ser fornecidos como parâmetros. Mesmo as soluções que permitem análise semelhante considerando múltiplos provedores, conforme Cloudorado (2022) e RightCloudz (2023), limitam-se a estimar o custo em provedores públicos comparando-o com o gasto necessário para uma infraestrutura privada, requerendo que o usuário defina possíveis alternativas, objetivos e critérios de escolha. Além disso, nenhuma dessas soluções auxilia na tarefa de decidir a capacidade dos recursos para implantar a aplicação.

Após estimar o necessário, é iniciado o provisionamento com a atividade de descoberta de recursos, que consiste em encontrar uma lista de recursos disponíveis nos ambientes considerados. A atividade seguinte, chamada de seleção, é o processo de escolher o melhor recurso da lista gerada pela descoberta, tendo como base as restrições impostas pelo usuário (Singh; Chana, 2016). Nesse caso, é assumido que o resultado da seleção é um conjunto de tipos de VM.

Na abordagem aqui apresentada, o custo e a localização da VM são importantes aspectos na solução. Em virtude disso, estes atributos são incluídos como parte da descrição do tipo de VM, diferente dos provedores de nuvem pública, que consideram apenas atributos de capacidade de hardware para classificar um tipo. Ao adotar nesta obra essa estratégia, o custo e a localização são usados

para distinguir tipos. Assim, VMs com custos e localizações diferentes são consideradas como de tipos distintos, ainda que sejam do mesmo provedor e tenham capacidade idêntica de hardware.

Dado os tipos de recursos selecionados, a última atividade no provisionamento de recursos é a alocação, que consiste na instanciação do tipo escolhido e na instalação dos componentes de software que serão executados nele. Por simplicidade, a seleção das imagens (*Virtual Appliances*) que serão usadas na implantação da aplicação é ignorada nessa atividade, pois assume-se que elas são fornecidas como entrada ao fazer a alocação de recursos.

É importante ressaltar que, na análise realizada, considera-se que os recursos são instanciados sob demanda, ou seja, não há um conjunto de instâncias de VM pré-criadas ou reservadas, o que justifica a inclusão da alocação de recursos como uma das etapas que necessariamente ocorre no provisionamento de recursos. Uma alternativa a essa suposição seria considerar que há um conjunto de VMs pré-instanciadas, que é, também, considerado na atividade de seleção de recursos. Dessa forma, se o recurso selecionado já possuir uma instância alocada, a atividade de alocação é substituída pela atividade de escalonamento de recursos. Contudo, ele não é incluído nesta análise porque esta atividade está fora do escopo do trabalho a ser apresentado.

Realizar a descoberta, a seleção e a alocação de recursos, de maneira automatizada, é um desafio devido à variedade de tecnologias envolvidas. Soluções da indústria que são específicas de provedor ou solução de nuvem, como AWS Cloud Formation (Amazon, 2022d), não são suficientes para solucionar completamente esse problema, uma vez que são limitadas a um ambiente específico e são incapazes de integrar múltiplos provedores, o que é essencial para satisfazer um conjunto de restrições limitado. Em contrapartida, soluções da indústria para o gerenciamento de recursos em ambientes com múltiplas nuvens, como RightScale (Flexera, 2022)

e Terraform (Hashicorp, 2023), propõem-se a oferecer provisionamento e gerenciamento de infraestrutura sobre múltiplos provedores de IaaS usando scripts de automação. Contudo, a adaptação desses scripts para cenários específicos não é uma tarefa simples. APIs de abstração de ambientes de nuvem, como OpenStack, proveem uma maneira de desacoplar as dependências do provedor de nuvem, facilitando o desenvolvimento de aplicações multinuvem. Contudo, elas não resolvem o problema de integrar diferentes APIs de gerenciamento e tecnologias, como é proposto na abordagem a ser apresentada.

Ferramentas para alocação de recursos como Puppet (Perforce, 2023) e Chef (Progress Chef, 2022), e arcabouços como Marionette Collective (Perforce, 2011) e Spiceweasel (Ray, 2022), permitem o gerenciamento de configuração de forma mais complexa. Além disso, há ferramentas como Juju (Canonical, 2022a), que permitem a orquestração de scripts de gerenciamento de configuração. Contudo, essas abordagens são limitadas a tarefas de instalação e configuração de componentes de software em VMs preexistentes. O provisionamento automatizado, usando essas estratégias no desenvolvimento de aplicações complexas (como composição de serviços com dependências entre si), não é trivial, uma vez que requer a escrita e a atualização dos scripts de integração.

A última atividade considerada é o monitoramento de recursos, que visa identificar, durante a execução das aplicações, falhas nos recursos alocados, bem como coletar medições sobre o uso de recursos, por exemplo, o percentual médio de uso da CPU. Essa atividade está diretamente relacionada ao monitoramento dos serviços, que consiste em verificar dinamicamente os parâmetros de QoS relacionados às aplicações (Alhamazani *et al.*, 2015), como tempo de resposta. Apesar dessa diferenciação, é preciso destacar que, além dos parâmetros que não são explicitamente relacionados a QoS, o monitoramento de recursos também pode ser usado para

identificar parâmetros de QoS, por exemplo, nível de ociosidade dos recursos. Contudo, na Figura 4, e no texto que a segue, consta somente o monitoramento de recursos, que não considera a priori aspectos de QoS.

Atualmente, praticamente todos provedores de nuvem pública oferecem aos seus usuários a capacidade de monitorar seus recursos usando ferramentas de monitoramento disponíveis para CPU, armazenamento e rede. No entanto, estas ferramentas geralmente são fortemente integradas com o provedor, como AWS CloudWatch (Amazon, 2022a), não permitindo o monitoramento de componentes implantados em outro provedor de nuvem. Para tal, existem soluções que são multiplataforma, como CloudHarmony (2023), mas que oferecem apenas métricas preestabelecidas.

Conforme ilustrado na Figura 4, as atividades relacionadas ao gerenciamento de recursos são comumente executadas em sequência (com exceção de estimativa e descoberta de recursos, que podem ser realizadas em paralelo), sendo que cada atividade depende dos resultados obtidos nas precedentes. Ao implantar uma aplicação, o usuário pode executar cada uma delas ou ignorar algumas atividades. Nesse caso, assume-se que os resultados esperados já foram previamente processados ou são conhecidos. Apenas as atividades de alocação e monitoramento de recursos devem necessariamente ser executadas na infraestrutura do provedor de nuvem, ao passo que as demais podem ser realizadas completamente fora desse ambiente, apesar de algumas requererem interação com o provedor.

A solução aqui apresentada considera todas as atividades relacionadas ao gerenciamento de recursos em nuvem. Embora sua contribuição mais significativa seja relacionada à seleção de recursos, também foram incorporados na solução aspectos relacionados às outras atividades

Principais desafios

Apesar da adoção cada vez mais ampla de computação em nuvem e da existência de diversos trabalhos que exploram este paradigma, há necessidades emergentes que constituem desafios a serem solucionados. Os principais são:

- *(Auto)gerenciamento*: a elasticidade e abstração de uso dos recursos encontradas em um ambiente de nuvem requer gerenciamento eficiente. É necessário que as tarefas de manutenção sejam realizadas de forma autônoma (Rahman *et al.*, 2011), fazendo com que os usuários não precisem conhecer detalhes da infraestrutura e haja o mínimo de esforço por parte do provedor dos recursos.
- *Gerenciamento de energia*: enquanto infraestruturas físicas, nuvens são essencialmente *datacenters* que requerem um grande volume de energia para se manter em operação. Reduzir o custo de energia pode ser uma alternativa para aumentar o lucro (Garg *et al.*, 2011). Contudo, manter a disponibilidade de recursos e, ao mesmo tempo, construir um modelo sustentável representa outro desafio.
- *Privacidade e segurança*: migrar os dados de uma organização para recursos gerenciados por terceiros ainda representa um dos principais obstáculos ao uso de nuvens. Há um elevado grau de preocupação com a privacidade e segurança das informações compartilhadas, sendo que o oferecimento de um ambiente seguro e confiável é um tema frequentemente discutido.
- *Gerenciamento de dados*: com a geração contínua de um volume cada vez maior de dados, o gerenciamento se torna extremamente complexo, sobretudo, ao se considerar recursos heterogêneos, como geralmente existem em nuvens.

- *Interoperabilidade*: a comunicação e a migração de aplicações e dados entre provedores de nuvem ainda não é possível na maioria dos cenários. Mecanismos que permitam esse intercâmbio constituem uma necessidade prioritária, principalmente em nuvens híbridas (Javadi; Abawajy; Buyya, 2012).
- *Qualidade de serviço (QoS)*: o oferecimento e o controle de QoS em ambientes de nuvem representa um dos maiores desafios atuais, uma vez que há a necessidade de acesso amplo por diferentes aplicações e categorias de clientes. A falta de padronização e a alta heterogeneidade encontrada nesses ambientes constituem os principais problemas.

Apesar da proposta aqui apresentada estar mais fortemente relacionada ao oferecimento de qualidade de serviço (aliada à satisfação de demais restrições), busca-se, também, contribuir, mesmo que indiretamente, com outros desafios aqui listados. A modelagem de recursos abstrai detalhes da infraestrutura e guia as atividades do sistema, provendo um nível mais autonômico de gerenciamento. Os critérios de segurança relacionados aos atributos dos recursos podem ser tratados na proposta como restrições em uma aplicação. Além disso, a possibilidade de utilização de diferentes provedores de nuvem na implantação de aplicações favorece a interoperabilidade.

Coreografia de serviços

O desenvolvimento de sistemas orientados a serviços é complexo, principalmente devido à existência de serviços fortemente pervasivos e com níveis cada vez mais elevados de heterogeneidade em termos de paradigmas de interação, protocolos de comunicação e modelos de representação de dados (Georgantas *et al.*, 2013).

Além disso, a rapidez com a qual as expectativas dos usuários e do ambiente evoluem exige soluções cada vez mais autônomas e dinâmicas. Consequentemente, novos paradigmas de composição que permitem adaptação dinâmica são constantemente investigados. Coreografia de serviços é uma forma de composição promissora e flexível, que provê uma metodologia para especificação adaptável do comportamento esperado dos serviços, sua colaboração e as mensagens trocadas (Bartolini *et al.*, 2012a).

A visão integrada de serviços passou a ser o modelo de referência para a Internet (Papazoglou *et al.*, 2007) e se torna ainda mais evidente na chamada Internet do futuro, resultante da evolução da atual com a união e cooperação da Internet de conteúdos (Daras *et al.*, 2009), Internet de serviços (Papadimitriou, 2009) e Internet das coisas (Atzori; Iera; Morabito, 2010). Coreografias de serviços são uma alternativa para satisfazer as necessidades emergentes nesta nova realidade. A popularização de soluções desenvolvidas por meio de cooperação entre serviços, no entanto, destacou alguns problemas que não eram facilmente perceptíveis nos esforços de integração anteriores, pois dificilmente alcançavam a escala que os sistemas possuem agora (Vincent *et al.*, 2010).

Coreografias constituem um paradigma de composição na qual o protocolo de interação entre os serviços participantes é definido em uma perspectiva global. Cada papel na coreografia especifica o comportamento esperado dos participantes que irão executá-lo por meio da sequência e da periodicidade das mensagens produzidas e consumidas por eles (Qiu *et al.*, 2007). Em tempo de execução cada participante na coreografia executa seu papel de acordo com o comportamento de outros participantes (Peltz, 2003), o que é conhecido como encenação da coreografia.

Esta forma de criar processos de negócios é comumente confundida com orquestração de serviços, uma estratégia complementar na qual um único agente é responsável por controlar e coordenar

as interações. Como ilustrado na metáfora da Figura 5, diferente de orquestrações, que têm fluxo de controle centralizado, coreografias envolvem um arcabouço de composição de serviços compartilhados, em que somente as funcionalidades dos participantes e as mensagens associadas passadas são descritas. Mais especificamente, coreografias se baseiam em um estilo de comunicação par-a-par, ou seja, cada serviço envolvido na composição sabe exatamente quando executar suas operações e com quem interagir, sem um controle centralizado (Chung; Leite, 2009). Nesse sentido, uma coreografia pode ser vista como um contrato entre um conjunto de serviços que rege como uma colaboração deve ocorrer.

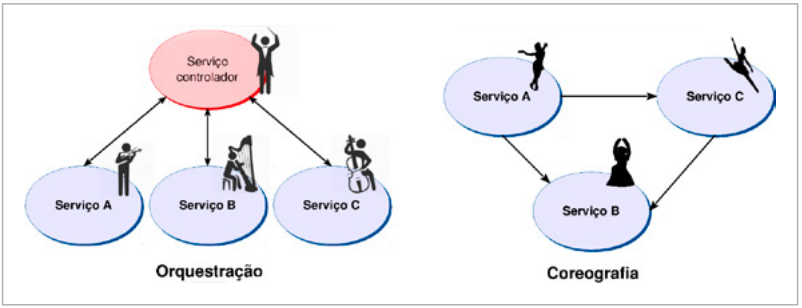


Figura 5 – Diferentes formas de composição de serviços
Fonte: Elaboração própria.

Embora o paradigma de coreografia pressuponha um modelo global de interação entre participantes, não é necessário que a implementação de cada serviço participante tenha conhecimento do fluxo de negócio completo da coreografia. É suficiente que cada serviço tenha conhecimento de sua parte no fluxo. Assim, cada participante da coreografia pode ter o seu comportamento modelado usando uma linguagem de orquestração. Com isso, uma coreografia pode também ser modelada como um conjunto de orquestrações distribuídas que interagem entre si, de forma que apenas os orquestradores precisam estar cientes de condições impostas pela coreografia (Poulin, 2011).

Cada papel em uma composição pode ser executado por um conjunto de serviços alternativos que podem servir como substitutos um para o outro. Dessa forma, coreografias têm emergido como uma alternativa promissora no contexto da internet do futuro, na qual milhões de serviços, coisas e recursos participam em cenários complexos e de grande escala (Issarny *et al*., 2011). Esse interesse também é aumentado pelo fato de que esta é uma alternativa promissora para formar sistemas complexos, dada a ampla aceitação do paradigma de arquitetura orientada a serviços.

Modelagem de coreografias

As linguagens para modelagem de coreografia de serviços podem ser categorizadas utilizando dois critérios (Decker; Kopp; Barros, 2008): linguagens independentes de implementação e linguagens específicas de implementação. As linguagens independentes de implementação são utilizadas principalmente para descrever processos da perspectiva de negócios, sem necessariamente expressar aspectos de como isso é implementado. Por outro lado, as linguagens específicas de implementação definem os formatos de mensagens concretas ou protocolos de comunicação utilizados.

Existem duas abordagens de modelagem para linguagens de coreografia de serviços (Chung; Leite, 2009): modelos de interação e modelos de interconexão. Os modelos de interação usam interações atômicas como blocos de construção básicos e os fluxos de dados e de controle estão baseados nas dependências entre essas interações de uma perspectiva global. Os modelos de interconexão, por sua vez, 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 desse modo as interações.

A expressividade em cada uma dessas duas abordagens de modelagem é diferente (Decker; Kopp; Barros, 2008). Por um lado, modelos de interconexão permitem a modelagem de processos incompatíveis ou mesmo processos para os quais nenhum parceiro existe (Wolf, 2009). Como exemplo, o serviço *A* aguarda uma mensagem, mas o serviço *B* nunca envia esta mensagem. Por outro lado, modelos de interação podem representar restrições que precisam ser satisfeitas com uma comunicação adicional que não corresponde ao modelo de interação original (Zaha *et al.*, 2006a). Por exemplo, se o serviço *A* envia uma mensagem ao serviço *B* e o modelo de interação exige que o serviço *C* subsequentemente envie uma mensagem para o serviço *D*, o serviço *C* tem que saber quando o serviço *B* recebeu a mensagem.

A Figura 6 mostra a categorização das principais linguagens para modelagem de coreografia segundo os critérios e as abordagens citados. A seguir, essas linguagens são descritas resumidamente.

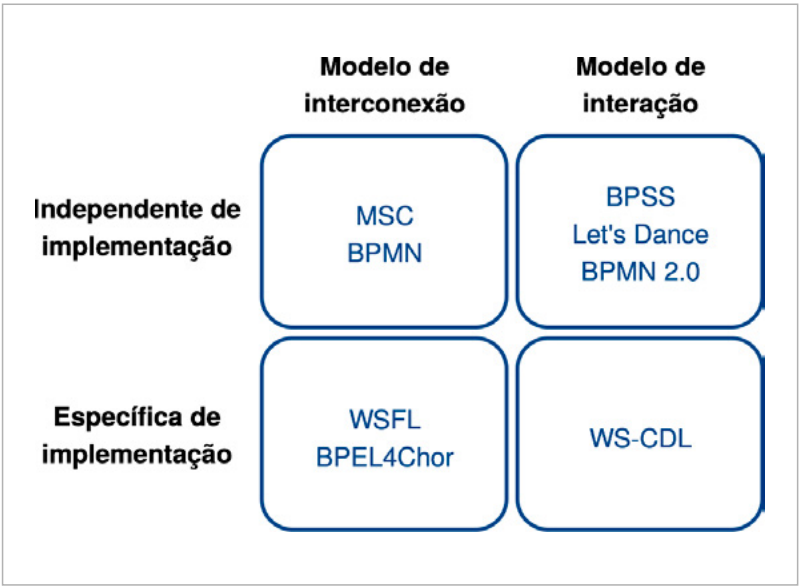


Figura 6 – Categorização de linguagens para modelagem de coreografias de serviços
Fonte: Adaptada de Engler (2009).

Message Sequence Chart (MSC) (Message [...], 2000) é uma notação gráfica para representar cenários de execução, modelando exemplos de fluxos convencionais ou alternativos da execução do sistema. Essa linguagem é mais adequada para modelar sequências de interações simples em vez de coreografias complexas, já que não suporta fluxos condicionais, fluxos paralelos e interações (Decker; Kopp; Barros, 2008).

Business Process Modeling Notation (BPMN) (Business [...], 2011) é uma linguagem gráfica proposta pelo *Object Management Group* (OMG) para modelar processos. Ela oferece uma notação em alto nível que permite o usuário focar no papel dos serviços na composição, sem ter que se preocupar em como esse papel é implementado. Essa 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. Uma desvantagem de BPMN 1.x é que ela permite apenas a especificação de coreografias por meio da interconexão de interfaces (usando diagramas de colaboração), sem descrever como a interação de fato ocorre. Essa deficiência foi corrigida na versão 2.0 da linguagem através de diagramas que modelam coreografias como entidades de primeira classe. Essa linguagem passou a ser o padrão de fato para modelar graficamente processos de negócio (Decker; Kopp; Barros, 2008).

A *Web Services Flow Language* (WSFL) (Leymann, 2001) foi proposta pela IBM e teve como objetivo contribuir para um futuro padrão nesta área. Esta linguagem possui dois tipos de descrições. O primeiro, modelo de fluxo (*Flow Model*), visa descrever fluxos de trabalho que interagem com outros, sendo as interações modeladas como serviços Web, cuja implementação é especificada usando uma descrição WSDL. O outro tipo, modelo global (*Global Model*), permite indicar ligações entre as operações de processos que sejam definidos por WSDL. Dessa forma, WSFL pode ser usada apenas para definir a estrutura da coreografia.

Business Process Execution Language (BPEL) (Barreto, 2007) é a linguagem mais comumente utilizada para implementar processos de negócios baseados em serviços Web (Decker *et al.*, 2008). A BPEL permite especificar a ordenação das mensagens trocadas entre os serviços, mas seu foco é descrever o comportamento da comunicação para um serviço individual. Como consequência, essa linguagem não pode ser usada para representar coreografias. A fim de contornar essa limitação, em "BPEL4Chor: Extending BPEL for modeling choreographies", Decker *et al.* (2007) propuseram extensões em BPEL para representação de coreografias, criando a linguagem BPEL4Chor. Nela, os processos BPEL abstratos são utilizados para descrever o comportamento entre participantes, formando uma topologia a partir da sua união por meio de mensagens.

A linguagem ebXML *Business Process Specification Schema* (BPSS) (Clark *et al*., 2001) visa descrever a colaboração entre dois participantes, proporcionando um conhecimento preciso acerca das mensagens trocadas, do seu sequenciamento e do estado final das interações. Sua desvantagem é coibir a interação de um número maior de participantes.

A linguagem *Let's Dance* (Zaha *et al*., 2006b) foi criada visando capturar a interação entre serviços sob uma perspectiva comportamental nas fases de análise e projeto de sistemas. Seu objetivo principal é abstrair completamente detalhes de implementação. *Let's Dance* suporta mais cenários de coreografias do que BPMN2, mas não é amplamente adotada na indústria.

A *Web Services Choreography Description Language* (WS-CDL) (Kavantzas *et al*., 2005) é baseada em XML e descreve colaborações fim-a-fim modeladas como serviços Web. Esta linguagem define a composição de um ponto de vista global às várias entidades participantes, a partir da qual pode-se extrair as definições dos processos que cada participante deve implementar. Essa visão global tem a vantagem de permitir a separação do processo a ser seguido por uma organização individual, da definição da sequência segundo a

qual ela trocará mensagens com as outras organizações. Contudo, WS-CDL é criticada por não separar de maneira apropriada o metamodelo da sintaxe, por oferecer suporte insuficiente a algumas categorias de cenários e por não ser de fácil compreensão (Barros; Dumas; Oaks, 2005).

Essas linguagens têm os seguintes elementos em comum (Kattepur; Georgantas; Issarny, 2013): (1) mensagens: tipos e conteúdo das mensagens trocadas entre os participantes; (2) ordenação das mensagens: restrições de tempo para mensagens síncronas e assíncronas com pré/pós-condições associadas; (3) endpoints: participantes da coreografia que devem ser acessados com protocolos específicos. Porém, nenhuma delas possui sintaxe para modelagem de restrições não funcionais sobre a coreografia. Na solução aqui apresentada é proposta uma representação para contornar essa limitação. Antes, no entanto, são discutidos maiores detalhes de BPMN 2.0 em virtude de sua ampla adoção na academia e na indústria.

MODELAGEM DE COREOGRAFIAS COM BPMN 2.0

Em BPMN 2.0, um processo de negócios é representado usando um diagrama de processo de negócio (*Business Process Diagram* – BPD), que é composto por um conjunto de atividades parcialmente ordenadas sendo executadas pelos participantes do processo.

Um BPD é essencialmente uma coleção de divisões identificadas como raias, de objetos e de fluxos de sequência e de mensagens. Raias representam os participantes do processo de negócios. Objetos podem ser eventos, tarefas ou conectores. Fluxos de sequência determinam a ordem de execução entre dois objetos em uma mesma raia, enquanto fluxos de mensagens representam mensagens trocadas entre dois objetos de raias diferentes.

A Figura 7 apresenta um subconjunto dos principais elementos em BPMN 2.0 para modelagem de coreografias de serviços.

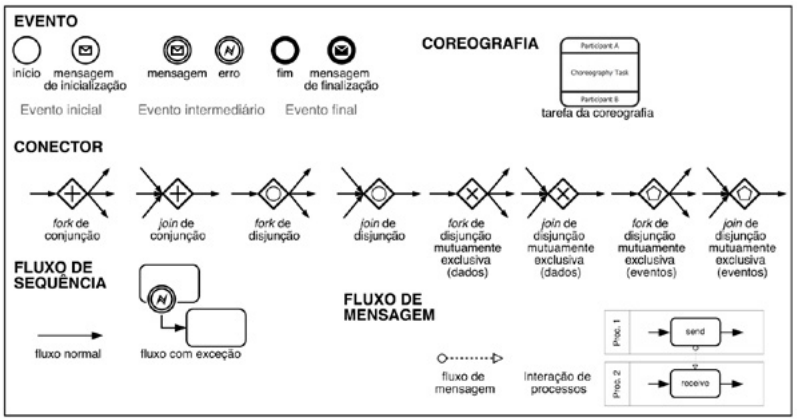


Figura 7 – Subconjunto dos principais elementos BPMN 2.0 para modelagem de coreografias de serviços
Fonte: Adaptada de Dijkman, Dumas e Ouyang (2008).

Eventos podem representar o início de um processo (evento de inicialização), o fim (evento de finalização) ou algo que deve acontecer durante o processo (evento intermediário). Há diferentes tipos de evento. Na figura, são apresentados três: eventos sem conotação especial (usados quando o modelador não especifica o tipo); eventos de mensagem (usados quando mensagens são usadas como gatilho do processo, para indicar envio ou recebimento de mensagens durante o processo, ou para indicar o fim do processo); eventos de erro (podendo indicar que o fluxo dispara ou captura um erro).

Uma tarefa é uma atividade atômica que faz parte de uma coreografia. Cada tarefa é representada como um componente que apresenta três divisões. A divisão superior representa o participante que inicia a interação, a inferior representa o participante que é o dono da tarefa e a divisão intermediária é usada para representar a tarefa propriamente dita.

Conectores são usados para controlar fluxos de sequência. Um conector *fork* de conjunção é utilizado para criar fluxos paralelos, ao passo que um conector *join* de conjunção é usado para sincronizar fluxos paralelos.

Um conector *fork* de disjunção divide o fluxo de sequência e ativa um ou mais alternativas de acordo com uma condição.

O conector *join* de disjunção aguarda que todos os ramos de entrada ativos sejam concluídos antes de acionar o fluxo de saída. O conector *fork* de disjunção mutuamente exclusiva divide o fluxo de sequência usando como referência o valor de dados ou a ocorrência de eventos, encaminhando-o para exatamente um dos ramos de saída. Por outro lado, o conector equivalente *join* aguarda a conclusão de um ramo de entrada antes de acionar o fluxo de saída.

Neste livro, considera-se apenas o modelo de interação em BPMN 2.0. Dessa forma, os modelos discutidos no restante do livro apresentam apenas o fluxo de sequência, uma vez que o fluxo de mensagens é inferido pelo próprio diagrama da coreografia.

A Figura 8 apresenta um exemplo de coreografia modelada usando essa notação e a representação do mesmo processo em um diagrama de atividades UML.

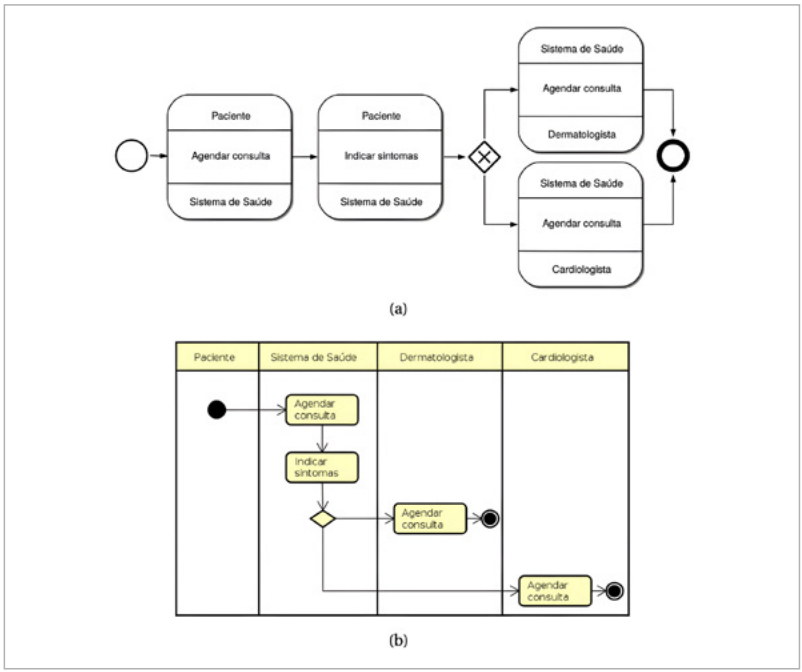


Figura 8 – Modelagem de coreografias de serviços
(a) Exemplo de coreografia modelada com BPMN 2.0.
(b) Representação das tarefas da coreografia em um diagrama de atividades.
Fonte: Elaboração própria.

Como pode ser visto, as setas no BPD, ao contrário do diagrama em UML, são usadas para representar unicamente a sequência de tarefas e não indicar como a interação ocorre. Essa informação está implícita na representação da tarefa.

Implantação de coreografias

Enquanto a implantação de orquestrações de serviços baseadas em BPEL é bem consolidada, existindo vários ambientes de execução, coreografias são, na maior parte das vezes, consideradas artefatos de projeto em vez de artefatos de implementação (Issarny *et al.*, 2011). Para contornar esse problema, muitos trabalhos na literatura propuseram a composição automática de serviços com base em coreografias (Flügge; Tourtchaninova, 2004; Sirin; Hendler; Parsia, 2003; Su *et al.*, 2008).

A ideia comum por trás dessas abordagens é utilizar uma especificação de requisitos que a coreografia tem que satisfazer com uma especificação do comportamento dos serviços participantes na coreografia. Por meio dessas duas especificações, a descrição dos serviços é gerada visando satisfazer os requisitos. Além dessas propostas, há outras que sugerem transformação entre modelos e linguagens (para permitir a geração de interfaces por meio de coreografias) (Hofreiter; Huemer, 2009; Rosenberg *et al.*, 2007).

A implantação de composições de serviços deve lidar com diversos desafios, sendo que o principal está relacionado ao provisionamento de recursos. Ao implantar uma composição, é preciso decidir qual provedor de recursos é mais adequado para cada serviço ainda não implantado, levando em conta sua relação com os serviços de terceiros já em execução. As perguntas que precisam ser respondidas são:

1. Para cada serviço, qual a capacidade de hardware que o recurso selecionado deve possuir? Em outras palavras, que tipo de VM deve ser escolhido para cada serviço de forma a atender sua demanda?
2. Quantas instâncias do tipo de recurso selecionado precisam ser criadas para atender as restrições impostas sobre as composições?
3. Onde alocar cada uma das instâncias de recurso? Nessa última decisão está o problema da heterogeneidade, quando se usa um ambiente com múltiplos provedores, o qual deve ser resolvido tanto na seleção de recursos quando na sua alocação.

A implantação automatizada e autônoma de coreografias de serviços ainda é tratada de maneira muito incipiente. Geralmente, essa atividade era realizada empiricamente, o que acarretava muitos problemas em casos que a pessoa responsável pela implantação mantém a documentação relacionada incompleta, adotando pressupostos não compartilhados por todo o time responsável por uma aplicação ou serviço (Humble; Farley, 2010). Na revisão bibliográfica realizada neste livro, foi possível notar que na maioria das abordagens encontradas há uma preocupação apenas superficial com os recursos necessários para implantar os serviços que compõem as coreografias, sendo a satisfação de restrições não funcionais tratada apenas na seleção desses serviços, assumindo que há recursos que ofereçam o nível de qualidade suposto.

Para implantar uma ou mais composições usando a abordagem proposta neste livro não há uma diferenciação conceitual de composições que usam o modelo de coreografia ou outro modelo. Isso porque, eventuais coordenadores em outros modelos também

são implementados como serviços Web. Dessa forma, em acordo com Leite (2014), utilizam-se os termos "coreografia" e "composição de serviços" indistintamente no restante do texto.

Restrições sobre serviços

Um produto de software é desenvolvido com o objetivo de atender a um conjunto de funcionalidades e atributos gerais que dizem respeito a limitações ou à qualidade com que estas funcionalidades são realizadas. Estes requisitos são comumente chamados de requisitos não funcionais (RNF) (Roman, 1985), atributos de qualidade (Boehm; Brown; Lipow, 1976; Keller, 1990) ou objetivos (Mostow, 1985). Contudo, estes termos são geralmente empregados em atividades relacionadas ao desenvolvimento do produto de software ou quando sua utilização está limitada a um escopo pré-conhecido. Uma vez que na abordagem apresentada é proposto que serviços podem ser utilizados em cenários distintos, e dado que se refere somente a aspectos relacionados à execução dos serviços (que não necessariamente foram pensados durante seu desenvolvimento), foi considerado adotar o termo *restrição*.

As restrições mais comuns são aquelas associadas à qualidade de serviço (QoS). Este termo se refere a quão bem um serviço ou produto é executado, levando em consideração critérios como desempenho, confiabilidade, segurança e usabilidade, entre outros. A QoS pode englobar desde o hardware, passando pelos sistemas operacionais e protocolos de rede, até chegar aos serviços da camada de aplicação (Siqueira; Cahill, 2000).

Neste livro, classifica-se QoS em três categorias, que são ilustradas na Figura 9:

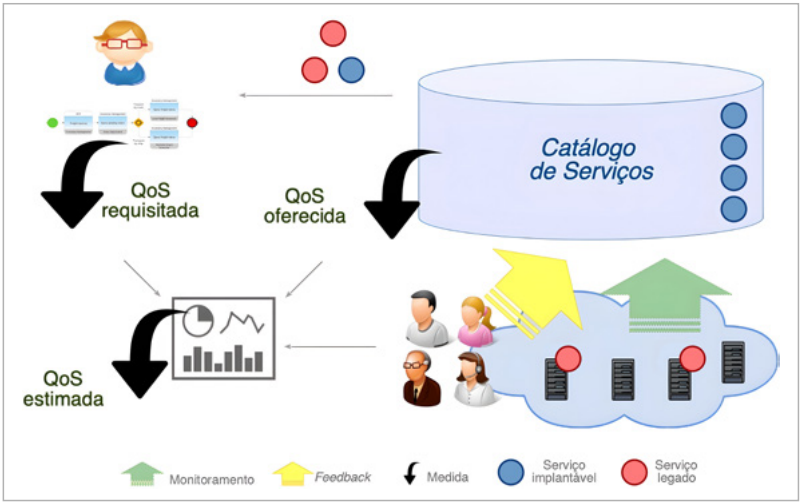


Figura 9 – Categorização de tipos de QoS e sua fonte de obtenção
Fonte: Elaboração própria.

- QoS requisitada: refere-se ao nível de QoS fornecido como restrição na descrição de um serviço ou coreografia. Está relacionado ao que o usuário espera obter como qualidade ao solicitar a implantação de uma aplicação, sendo normalmente traduzida para um acordo de nível de serviço (*Service Level Agreement* – SLA) entre o usuário e o provedor de recursos.
- QoS estimada: é o nível de QoS esperado para um serviço que ainda não foi implantado. Esse valor pode ser obtido considerando execuções prévias do serviço ou usando alguma estratégia de predição (Goldman; Ngoko; Milojicic, 2012; Reig; Alonso; Guitart, 2010; Ye; Bouguettaya; Zhou, 2012). Assume-se que medições dessa categoria podem ser usadas como parâmetro de decisão no provisionamento de recursos. Contudo, é reconhecido que, por se tratarem de estimativas, adaptações em tempo de execução podem ser necessárias caso a estimativa seja feita de forma imprecisa.

- QoS oferecida: é a QoS garantida para um determinado serviço por meio de um SLA. Os níveis de QoS estabelecidos em SLAs com provedores de nuvem se encaixam nessa categoria. Por exemplo, a Amazon oferece em seu serviço EC2 uma disponibilidade de 99,99% considerando o tempo de disponibilidade (Amazon, 2022e). Este SLA, por sua vez, pode ser firmado como resultado da negociação prévia entre o provedor e o usuário, usando a QoS requisitada como parâmetro; ou ser uma oferta estabelecida pelo provedor, comum a todos os usuários. Usa-se, neste trabalho, essa categoria principalmente para se referir a serviços de terceiros, para os quais não se tem controle. A proposta considera um catálogo de serviços (Liu; Ngu; Zeng, 2004), que contém a descrição de um conjunto de serviços disponibilizados, com informações sobre o nível de QoS garantido pelo provedor de cada serviço. Valores de QoS oferecida também podem ser utilizados para estimar a QoS para cenários semelhantes ou cenários que dependem desse serviço.

Não há consenso sobre quais são os critérios de qualidade esperados para um serviço disponibilizado na Web (Aiello; Giorgini, 2004). A visão tradicional herdada da comunidade de redes coloca somente desempenho e disponibilidade no conjunto de propriedades de QoS, mas outras propriedades são relevantes também, como manutenibilidade, integridade, confiabilidade e segurança (Mani, 2002).

O oferecimento de QoS é um dos principais desafios em computação em nuvem, e este aspecto é ainda mais crítico ao implantar coreografias de serviços nesses ambientes, devido à existência de múltiplos serviços cooperando entre si e ao possível compartilhamento desses serviços entre as coreografias. Os mecanismos de elasticidade oferecidos por soluções de nuvem, usualmente,

buscam controlar aspectos de qualidade de qualquer uma das partes que interagem entre si. Dessa forma, linguagens de especificação de coreografias de serviços devem ser enriquecidas com notações que expressam as propriedades não funcionais que o serviço coreografado deve apresentar (Bartolini *et al.*, 2012a). Essas notações facilitam que os níveis de QoS, estabelecidos entre as partes envolvidas, sejam, então, formulados em termos de SLAs, mais técnicos e monitoráveis (Bertolino *et al.*, 2012), uma vez que seguem uma sintaxe e semântica preestabelecidas.

Não somente restrições de QoS devem ser observadas na implantação e execução de serviços, mas também aquelas relacionadas a outros aspectos, como privacidade e segurança. Além disso, uma vez que composições de serviços podem ser formadas extrapolando fronteiras nacionais, deve-se considerar toda restrição relacionada à legislação e aos mecanismos regulatórios com respeito à privacidade, controle de acesso aos dados, segurança etc.

Relatórios governamentais e pesquisas acadêmicas esboçam as complexidades da criação de serviços em nuvem referentes à conformidade com regulamentações e enfatizam a necessidade de soluções que permitam tal conformidade (Brasil, 2018; Singh; Chatterjee, 2017; Vaquero; Rodero-Merino; Morán, 2011). Por exemplo, as Diretrizes de Proteção a Dados da União Europeia (Luxemburgo, 1995) proíbem a transferência de dados pessoais para países não membros, a não ser que sejam garantidos os níveis adequados de proteção. No Brasil, medida semelhante foi adotada com o Marco Civil da Internet (Brasil, 2014), estabelecendo que em operações realizadas por meio da Internet, em território nacional, deverão ser obrigatoriamente respeitados a legislação brasileira e os direitos à privacidade, à proteção dos dados pessoais e ao sigilo das comunicações privadas e dos registros.

Exemplo de cenário

Em sistemas orientados a serviços, estes são entidades de software autônomas que, interagindo com outros serviços em um estilo par-a-par, podem tomar decisões proativamente e engajarem em tarefas objetivando suas próprias necessidades e colaboração global (Autili; Inverardi; Tivoli, 2014). Para ilustrar a complexidade na implantação e na manutenção de um sistema com esta característica, é apresentado a seguir um exemplo do cenário trabalhado no livro, que constitui uma aplicação de gerenciamento de cadeia de suprimentos.

O gerenciamento da cadeia de suprimentos é a integração dos principais processos de negócio por meio de fornecedores que proveem produtos e informações que agregam valor aos clientes (Lambert; Cooper, 2000). À medida que as empresas fortalecem seus relacionamentos e colaboram em um nível intra e inter-organizacional, a própria cadeia ganha mais ligações e, portanto, aumentam os esforços de gestão e coordenação. Como resultado, o foco do gerenciamento da cadeia de suprimentos é cada vez mais transferido da eficiência da produção para abordagens voltadas aos clientes e direcionadas à sincronização de parcerias (Wang *et al.*, 2008).

A tecnologia da informação transformou a maneira como as empresas implementam essas abordagens, resultando em diferencial competitivo. Ao adotar soluções orientadas a serviços, por meio de coreografias implantadas em ambientes de nuvem, as cadeias de suprimentos podem, com mais rapidez e eficiência, colher os benefícios a custos reduzidos.

Uma implantação eficiente de coreografias de serviços nesse cenário é determinada pelo desempenho de toda a cadeia de suprimentos, e não somente de suas entidades individuais. Com isso, a implantação deve levar em conta diferentes restrições impostas em cada serviço. Essa pluralidade de restrições acontece porque cada serviço pode ser direcionado a clientes com necessidades heterogêneas e complexas (motivadas por idade, condição social etc.) e possuir carga sujeita a padrão sazonal e outras variáveis, como aumento da demanda devido a

promoções (ex.: *Black Friday*). Além disso, a implantação dos serviços deve enfrentar restrições derivadas de processos internos e externos.

Para simplificar, no exemplo discutido, são analisados apenas dois processos: o cumprimento de pedidos e o rastreamento de frete, descritos na Figura 10 usando a linguagem BPMN. O primeiro processo (descrito na Figura 10(a)) objetiva o cumprimento da ordem de compra após a seleção do produto. Ele inclui o planejamento de frete e gerenciamento de pagamentos, e é implementado usando uma composição de seis serviços. Para essa coreografia, assume-se que a opção de pagamento é selecionada pelo cliente antes de iniciar o processo (ao submeter o pedido). O segundo processo é interno (sem interação com clientes). Ele inclui as tarefas realizadas para obter um relatório de estado do frete contratado para produtos vendidos e do inventário da organização.

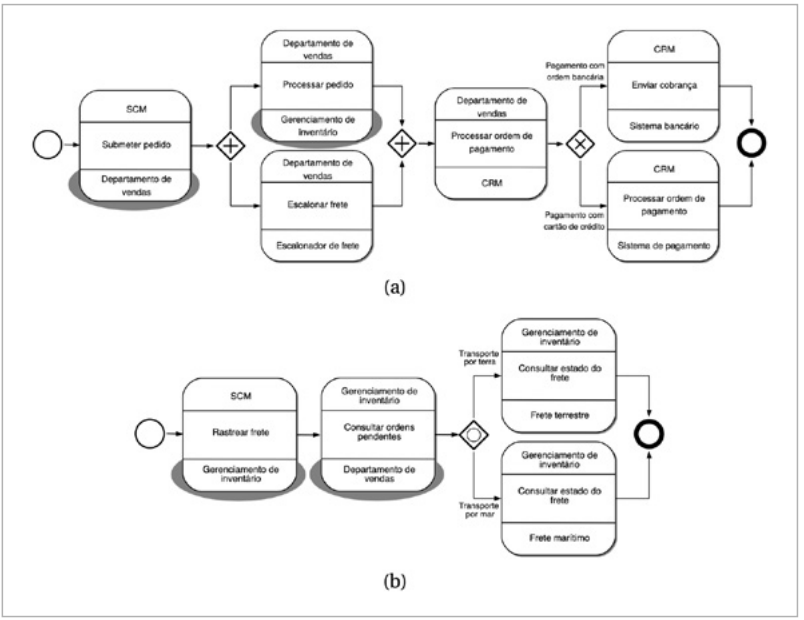


Figura 10 – Diagrama BPMN das coreografias apresentadas no exemplo do cenário
(a) Coreografia 1: Cumprimento de pedido.
(b) Coreografia 2: Rastreamento de frete.
Fonte: Elaboração própria.

Conforme ilustrado na Figura 10(b), assume-se que a empresa usa dois tipos de transporte (terrestre e marítimo) e que o relatório é gerado usando uma composição de quatro serviços. O Quadro 1 apresenta a descrição dos serviços para ambas as coreografias.

QUADRO 1
Exemplo do cenário: descrição dos serviços

Serviço	Descrição
Gerenciamento de inventário	Controla a encomenda e o armazenamento de produtos para venda
Departamento de vendas	Responsável por tarefas relacionadas a produtos destinados à venda
Frete terrestre	Interage com uma empresa que envia bens do fabricante ou produtor até o ponto final de distribuição por terra
Frete marítimo	Interage com uma empresa que envia bens do fabricante ou produtor até o ponto final de distribuição por mar
Escalonador de frete	Organiza remessas de produtos vendidos usando uma ou mais empresas de frete
CRM	Gerencia o relacionamento com o cliente (*Customer Relationship Management*)
Sistema bancário	Interage com parceiros bancários
Sistema de pagamento	Processa pagamentos feitos com cartão de crédito

Fonte: Elaboração própria.

A implantação dessas coreografias deve lidar com propriedades não funcionais requeridas em virtude de restrições inerentes do cenário considerado. Assume-se, nesse exemplo, as restrições não funcionais listadas no Quadro 2, que relaciona, também, cada serviço às restrições impostas sobre ele.

QUADRO 2
Exemplo do cenário: descrição das restrições não funcionais

Serviço	Restrições específicas a determinados serviços	Restrições comuns a todos os serviços
Gerenciamento de inventário	Na coreografia 1: objetivando as recomendações sobre experiência de usuário (Nielsen, 2009), o tempo máximo necessário para atender a cada requisição deve ser de 1 segundo. Na coreografia 2: a emissão de relatórios sobre frete deve manter uma vazão de 10 requisições por segundo. Não pode estar indisponível mais do que 12 horas por ano.	A implantação de serviços deve ser realizada usando recursos com menor custo financeiro (desde que as demais restrições sejam satisfeitas). A seleção de recursos deve ser realizada de forma a evitar provedores de nuvem com obrigação de cobrança mínima, visto que a utilização de recursos é desconhecida. A seleção de recursos deve privilegiar provedores de nuvem com melhor reputação.
Departamento de vendas	Na coreografia 1: o tempo máximo necessário para atender a cada requisição deve ser de 1 segundo. Na coreografia 2: a emissão de relatórios sobre frete deve manter uma vazão de 10 requisições por segundo.	
Frete terrestre	A emissão de relatórios sobre frete deve manter uma vazão de 10 requisições por segundo.	
Frete marítimo	A emissão de relatórios sobre frete deve manter uma vazão de 10 requisições por segundo.	
Escalonador de frete	O tempo máximo necessário para atender a cada requisição (na coreografia 1) deve ser 1 segundo.	
CRM	O tempo máximo necessário para atender a cada requisição (na coreografia 1) deve ser 1 (segundo. Dados sobre clientes brasileiros devem ser armazenados no Brasil.	
Sistema bancário	O tempo máximo necessário para atender a cada requisição (na coreografia 1) deve ser 1 segundo.	
Sistema de pagamento	O tempo máximo necessário para atender a cada requisição (na coreografia 1) deve ser 1 segundo. Deve ser mantido em uma nuvem privada, uma vez que gerencia uma base de dados sobre cartões de crédito.	

Fonte: Elaboração própria.

Considere que a empresa deseja implantar essas coreografias usando serviços preexistentes. Com isso, o primeiro desafio que surge é a seleção de serviços. Há papéis compartilhados entre as coreografias apresentadas, como gerenciamento de inventário e da venda de produtos (serviços sombreados na Figura 10). É desejável que haja compartilhamento de serviços para evitar custos adicionais, como licenças de software duplicadas e mais despesas na alocação de recursos.

O segundo e principal desafio está relacionado ao gerenciamento de recursos, principalmente às atividades de estimativa, seleção e alocação de recursos. Algumas das dificuldades são:

1. A estimativa e a seleção de recursos devem ser realizadas de forma que todas as restrições não funcionais sejam satisfeitas. Conforme resumido no Quadro 2, além de restrições comuns às duas coreografias, existem restrições específicas para cada uma delas e o compartilhamento de serviços causa interferência entre elas.
2. Serviços compartilhados podem desempenhar diferentes papéis em cada coreografia, como o serviço para gerenciamento de inventário que é usado para processar pedido e para rastrear frete. Além disso, há degradação da QoS nesses serviços causada pela concorrência proveniente da agregação da carga. Devido a essa concorrência, é preciso selecionar recursos com maior capacidade para que a QoS se mantenha dentro dos padrões esperados para o serviço.
3. As contribuições nas restrições fim-a-fim devem ser distribuídas de forma eficiente entre os serviços a que elas dizem respeito. Deve haver um balanceamento da QoS esperada para cada serviço, para que assim a agregação dos valores de QoS de todos os serviços que compõem a coreografia

fique dentro do limite requisitado. Esse balanceamento deve ser realizado de maneira global considerando todas as coreografias em que um determinado serviço é usado.

4. Dada a capacidade de recurso estimada para garantir que todas as restrições de QoS sobre o serviço a ser implantado sejam satisfeitas, é necessário decidir onde alocar essa capacidade. Esta tarefa pode iniciar com uma tentativa de alocação usando recursos disponíveis na própria organização. No entanto, quando isso não é possível é necessário fazer o mapeamento para recursos em nuvens públicas. A alocação de recursos também está sujeita às restrições que não têm relação direta com a capacidade, como a localização do recurso.

As três primeiras dificuldades relacionadas ao gerenciamento de recursos são problemas bem conhecidos. Para reforçar a motivação do trabalho, é apresentada uma análise mais detalhada sobre o quarto desafio. A seleção de recursos pode ser realizada usando um ambiente multinuvem para evitar *lock-in* de fornecedor, otimizar o desempenho e os custos e facilitar a satisfação das restrições não funcionais (Petcu *et al.*, 2013). No entanto, essa não é uma tarefa simples, devido ao grande número de opções de tipos de VM e a interseção entre eles. Atualmente, existem mais de 100 provedores de nuvem pública na categoria IaaS, que oferecem recursos em uma ampla quantidade de tipos (Cloudharmony, 2023). Por exemplo, comparamos os tipos de VM oferecidos pelos provedores líderes de mercado de acordo com o relatório sobre IaaS feito em 2021 pela Gartner (Bala *et al.*, 2021), AWS (Amazon, 2022b), Microsoft Azure (Microsoft, 2022a), Google Cloud (Google, 2022c) e Alibaba Cloud.

Como ilustrado na Figura 11, havia 671.855 tipos de VM[2] que devem ser considerados ao selecionar onde cada serviço coreografado será implantado, o que dependerá das restrições não funcionais associadas. Essa seleção não pode ser realizada apenas com base na capacidade de hardware, uma vez que tipos com exatamente a mesma capacidade possuem atributos variados (como custo e localização, por exemplo), o que pode interferir na seleção se houver alguma restrição associada a esses atributos. Em virtude disso, selecionar o recurso para cada serviço é um problema de otimização complexo, que se torna ainda mais difícil ao considerar diferentes padrões de interação entre os serviços e seu provável compartilhamento em múltiplas coreografias.

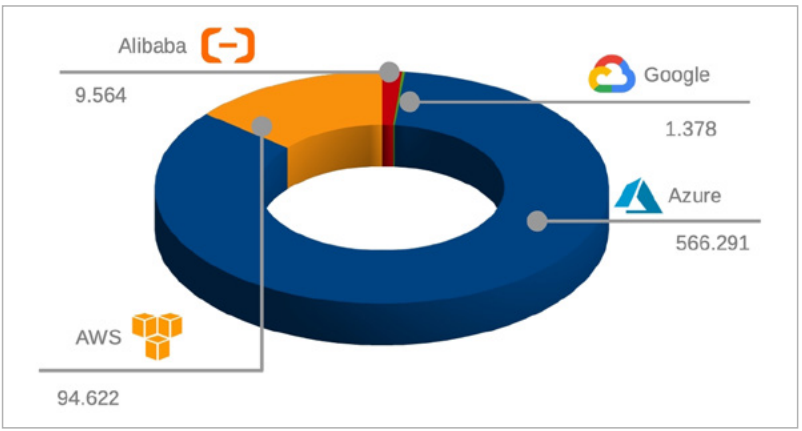


Figura 11 – Número de tipos de VM disponíveis nos quatro principais provedores de nuvem em dezembro de 2021
Fonte: Elaboração própria.

A seleção de recursos pode ainda explorar o compartilhamento dos próprios recursos, procurando implantar um conjunto de serviços em um único recurso. Isso ocorre quando o que foi selecionado para um certo serviço (ou conjunto de serviços) possui capacidade

2 Essa análise foi realizada considerando os tipos de VM disponibilizados pelos provedores considerados em dezembro de 2021.

maior que a sua demanda, permitindo que o recurso seja compartilhável. Essa estratégia é ainda mais necessária nos casos em que há grande demanda de comunicação entre os serviços, uma vez que a implantação conjunta de múltiplos serviços em um único recurso elimina a sobrecarga de transmissão de dados entre eles.

Neste capítulo foram discutidos, de maneira introdutória, alguns conceitos que serão utilizados no decorrer do livro: computação em nuvem, coreografia de serviços e restrições sobre serviços. Foram apresentados tópicos relacionados ao espaço do problema, assim como aqueles relacionados à solução proposta.

Computação em nuvem representa um avanço na TI e vem sendo empregada em cenários cada vez mais complexos, nos quais a entrega de soluções que satisfaçam os requisitos não funcionais das aplicações, representa um desafio. Para lidar com essa complexidade, diversas técnicas estão sendo propostas. Nesse sentido, propõe-se uma abordagem para o provisionamento eficiente de recursos para coreografias de serviços. Como será detalhado posteriormente, a solução se beneficia da variabilidade de tipos de recursos providos em ambientes de nuvem, assim como da elasticidade em seu uso.

Coreografia de serviços é uma maneira promissora de composição de serviços, possibilitando a interação de múltiplos serviços que cooperam entre si (Peltz, 2003). Para evidenciar a existência dos principais desafios na implantação de aplicações desenvolvidas nesse modelo, foi apresentado um cenário que será usado para ilustrar o problema no decorrer do livro. É proposto que a satisfação das restrições impostas sobre os serviços deva ser pensada desde as primeiras atividades de criação de uma coreografia, ou seja, no momento em que é feita sua especificação. Apesar disso, as soluções para modelagem e implantação de coreografias ainda são bastante incipientes para lidar com a dinamicidade inerente dos cenários atuais. No próximo capítulo são discutidas algumas dessas abordagens, destacando suas vantagens e desvantagens.

2

Abordagens para implantação de composições de serviços

Este capítulo discute o estado da arte da modelagem, da seleção e da alocação de recursos em nuvem. O objetivo é analisar como essas atividades são realizadas para serviços em geral e evidenciar as especificidades de composições de serviços descritas em coreografias. A análise das abordagens discutidas foi guiada pelas seguintes perguntas:

- As soluções existentes para modelagem de coreografias de serviços são suficientes para permitir a representação de restrições não funcionais que devem ser satisfeitas na implantação dos serviços? Caso contrário, que características devem ser incorporadas nessas soluções?
- Como o trabalho analisado determina a capacidade de recursos para implantar os serviços? Quais os critérios utilizados para selecionar o recurso mais apropriado? Qual o impacto dessa decisão, se aplicada à implantação de coreografias?
- Como as restrições não funcionais são consideradas na implantação do serviço ou na coreografia?

- As soluções propostas nos trabalhos analisados são independentes de tipos de aplicação/serviço e de provedor de nuvem? Como elas permitem abstrair a seleção e o mapeamento de recursos em nuvens híbridas ou qual seria o efeito colateral de aplicá-las em tais cenários?
- É possível utilizar essas soluções para implantar de maneira eficiente um conjunto de coreografias de serviços, de forma que as interferências nessas coreografias causadas pelo compartilhamento de serviços sejam absorvidas ou quais os efeitos colaterais de utilizá-lo para tal implantação?

A revisão bibliográfica foi realizada como primeira etapa da pesquisa que deu origem a este livro, sendo reforçada no decorrer do desenvolvimento do trabalho. Foram realizadas buscas nas bases científicas IEEE Xplore, ACM Digital Library, Scopus, SpringerLink, CiteSeerX e Web of Science. Os resultados obtidos foram complementados com buscas usando o Google Scholar.

Durante o desenvolvimento da pesquisa, ao todo foram pré-selecionados 423 trabalhos, de acordo com o título, o resumo, a introdução e a conclusão. Após uma análise mais criteriosa, esses trabalhos foram filtrados de forma a obter as propostas mais relevantes para a realização da pesquisa. Foram selecionados 22 trabalhos relacionados, cuja distribuição de acordo com o ano da publicação mais relevante e principal tema abordado é apresentada na Figura 12. Para facilitar a análise e isolar o escopo, os critérios estabelecidos como importantes no problema considerado foram divididos em três grupos – modelagem de coreografias, gerenciamento de recursos em nuvem e implantação de composições de serviços –, que serão discutidos nas próximas seções. Para cada grupo, é apresentado um comparativo desses trabalhos usando os critérios estabelecidos.

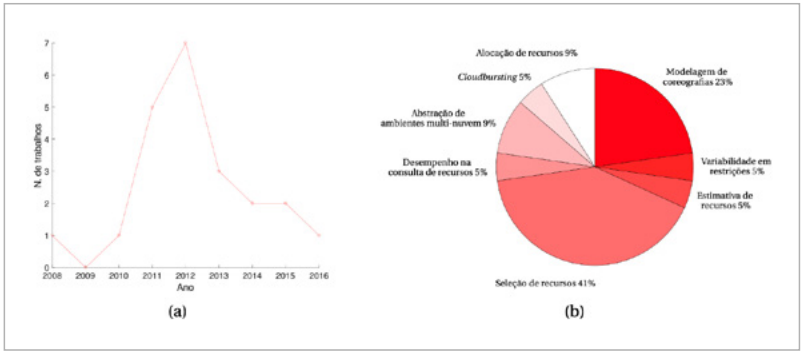


Figura 12 – Distribuição dos trabalhos relacionados
(a) Por ano da principal publicação.
(b) Por tema principal.
Fonte: Elaboração própria.

Modelagem de coreografias de serviços

No capítulo anterior foram discutidas as principais categorias de restrição não funcional, relacionadas à implantação e encenação de coreografias de serviços. Além de diferentes propriedades de QoS, as restrições não funcionais envolvem características do ambiente de implantação dos serviços, que podem ser diretamente relacionadas ao recurso ou ao provedor de nuvem que o disponibiliza. Também foram apresentadas as principais linguagens existentes para modelagem de coreografia de serviços. Embora as notações propostas sejam suficientes para representar as interações entre os serviços e expressar a lógica do processo de negócio executado, elas não permitem especificar todas as informações necessárias para realizar a implantação desses serviços, visando propriedades não funcionais. Mais precisamente, as linguagens existentes não permitem a representação das diferentes categorias de restrição de maneira não ambígua e processável. Para contornar este problema, há algumas propostas que visam estender notações preexistentes, enquanto outras visam estabelecer novas linguagens. Entre as classificadas como extensões, a maioria se baseia em BPMN (Business [...],

2011), uma vez que esta linguagem é o padrão de fato para a representação de coreografias de serviços.

A linguagem Quality for BPMN (Q4BPMN) se baseia na anotação de diagramas expressos em BPMN com requisitos de qualidade que os serviços coreografados devem satisfazer (Bartolini *et al.*, 2012b). As propriedades não funcionais podem estar relacionadas com uma tarefa, um participante ou a coreografia como um todo. Essa linguagem é uma implementação do Property Meta-Model (PMM) (Di Marco *et al.*, 2011), um metamodelo para expressar propriedades de qualidade, métricas e eventos observáveis de um sistema. Em Bartolini *et al.* (2012b), os autores propuseram o uso desse modelo BPMN anotado em conjunto com técnicas de *Model Driven Engineering* (MDE) (Kent, 2002) e a ferramenta KlaperSuite (Ciancone *et al.*, 2011) para permitir a análise automática de propriedades de qualidade sobre a coreografia.

Uma das desvantagens de usar essa linguagem é que as propriedades de qualidade ficam restritas às inerentes do metamodelo PMM. Apesar de ser completo o suficiente para expressar os tipos mais comuns de restrições de qualidade, não é possível incluir na modelagem de uma coreografia restrições não consideradas nesse metamodelo. Outra desvantagem é que essa linguagem não permite modelar restrições que são diretamente relacionadas a atributos do recurso (como sua localização), limitando-se a aspectos de qualidade de serviço. Além disso, essa linguagem não permite a modelagem ou a inspeção de múltiplos processos de negócio de maneira conjunta, tornando difícil o desenvolvimento de uma abordagem para a implantação de múltiplas coreografias de maneira coordenada.

Performability-enabled BPMN (PyBPMN) (Bocciarelli; D'Ambrogio, 2011) é uma extensão do metamodelo de BPMN para a especificação de propriedades de desempenho e confiabilidade usando o perfil UML Modeling and Analysis of Real-Time Embedded systems (Marte) (UML Profile [...], 2009) e suas extensões propostas em Bernardi, Merseguer e Petriu (2008a, 2008b).

De forma semelhante à Q4BPMN, técnicas de MDE associadas à especificação do metamodelo permitem que os artefatos desenvolvidos sejam transformados em modelos de análise de desempenho e confiabilidade. Uma limitação dessa proposta é que o mecanismo de anotação proposto para especificar os atributos de qualidade é complexo, dificultando o uso por usuários que não estão familiarizados com este padrão. Além disso, ele também só engloba restrições relacionadas ao desempenho e à confiabilidade do serviço em um único processo de negócio. Logo, essa linguagem apresenta as mesmas limitações apontadas para a Q4BPMN com relação à expressividade de restrições e tratamento conjunto de múltiplas composições, dificultando seu uso no escopo tratado para o problema.

Os autores Pavlovski e Zou (2010) propuseram o uso de duas construções apresentadas como uma extensão de BPMN para modelar requisitos não funcionais sobre o processo de negócio (Zou; Pavlovski, 2006). A primeira construção, denominada *condição operacional* (*operating condition*), denota uma restrição sobre o processo de negócio, enquanto a segunda, *caso de controle* (*control case*), define critérios de controle para mitigar riscos associados com uma condição de operação. Os autores alegam que a representação dessas características na fase inicial do processo de desenvolvimento de software permite determinar de maneira detalhada grande parte dos requisitos não funcionais. Além de atributos de qualidade, a proposta também permite modelar restrições de regulamentação. As condições operacionais propostas pelo modelo podem ser usadas para decompor as restrições, mas a especificação de ambas as construções é feita de maneira informal. Isso torna difícil garantir a implementação automatizada de análises realizadas usando essa linguagem, especialmente considerando de forma conjunta múltiplas composições, visto que a linguagem não oferece nenhuma construção com esse intuito.

A BPMN4TOSCA é uma extensão de BPMN usando OASIS *Topology and Orchestration Specification for Cloud Applications* (Tosca) (Palma; Spatzier, 2013), um padrão para representar, de maneira portátil, topologias de aplicativos em nuvem e seu gerenciamento, o que inclui provisionamento de recursos e implantação da aplicação (Kopp *et al.*, 2012). O modelo gerado usando essa linguagem não é uma descrição de um processo de negócio, mas sim um plano de gerenciamento (termo utilizado no trabalho), que descreve, por exemplo, quais componentes de software devem ser instalados e os *scripts* que orquestram essa instalação. Dessa forma, diferente das propostas anteriores que tem como escopo as etapas de análise e projeto da aplicação, BPMN4TOSCA objetiva as atividades relacionadas a sua implantação. Essa abordagem pode ser empregada quando o usuário é o responsável direto pela alocação de recursos. Contudo, o objetivo do trabalho aqui apresentado é abstrair completamente as tarefas relacionadas a esta atividade, não justificando, assim, sua adoção. Outro problema em considerar seu uso é que planos de gerenciamento são altamente acoplados a uma única aplicação, sendo difícil reusá-los e gerenciá-los (Buschmann; Henney; Schimdt, 2007).

A linguagem *Extended Process Pattern Specification Language* (EPPSL) (Khaluf; Gerth; Engels, 2011) tem como objetivo a modelagem de restrições de qualidade usando uma extensão de diagramas de atividade da UML 2.0. O modelo gerado indica quais são as restrições de qualidade e em quais atividades elas devem ser aplicadas. De acordo com os autores, a linguagem tem maior capacidade de expressividade por permitir a modelagem de restrições temporais não determinísticas usando *Computation Tree Logic* (CTL) (Emerson, 1991), como restrições que futuramente não se aplicarão a todos os fluxos de controle em um processo de negócios. Essa capacidade é relevante principalmente para instrumentar mecanismos de adaptação dos recursos face a mudanças em tempo de execução. Contudo, o principal objetivo do trabalho é permitir a verificação do modelo gerado, sendo não recomendado o uso dessa solução visando apenas a criação de modelos devido a sua alta complexidade.

Discussão

Nesta seção é realizado um comparativo dos trabalhos apresentados, considerando características tratadas como relevantes em uma linguagem de modelagem de coreografias para que ela possa guiar a implantação dos serviços de maneira eficiente e de acordo com as restrições não funcionais. O Quadro 3 divide essas caraterísticas em duas categorias: escopo, indica os aspectos que são relevantes em uma linguagem de modelagem no contexto estabelecido; e propriedades que auxiliam na implantação da aplicação, indica as características da linguagem que podem facilitar as atividades relacionadas à implantação dos serviços. Apesar de alguns atributos não fazerem parte do domínio de todas as linguagens, pode-se notar que nenhum trabalho contempla de maneira completa todos os aspectos considerados.

QUADRO 3
Comparativo das principais propostas para modelagem de coreografias de serviços

	Q4BPMN	PyBPMN	Pavlovski e Zou	BPMN4TOSCA	EPPSL
Escopo					
Modelagem do processo de negócio	√	√	√	∅	√
Modelagem de restrições sobre determinada tarefa (serviço)	√	√	√	√	√
Modelagem de restrições fim-a-fim	√	∅	∅		
Representação de múltiplas categorias de restrições			√		√
Representação de múltiplas propriedades de QoS	√	∅	√	×	×
Propriedades que auxiliam na implantação da aplicação					
Linguagem definida de maneira formal	√	√		√	√
Representação explícita das dependências entre as entidades	∅	∅	∅	×	
Representação conjunta de múltiplos processos de negócio				√	
Representação de componentes associados à avaliação das restrições					

Legenda: √ Contemplado, ∅ Contemplado com limitações, × Fora do escopo do trabalho
Fonte: Elaboração própria.

Entre as principais limitações encontradas no escopo das propostas existentes, está a impossibilidade de modelar de forma não ambígua restrições fim-a-fim, sobre o processo de negócio. Em alguns casos, como em PyBPMN, essa funcionalidade é oferecida indiretamente por meio da agregação de restrições a tarefas individuais, sendo necessário a quebra do valor-alvo da restrição entre as tarefas ou subprocessos que fazem parte dessa restrição. Outro problema é que os trabalhos são voltados principalmente para restrições de QoS, permitindo a modelagem de um subconjunto predeterminado de propriedades desse tipo, sendo outras restrições (como aquelas relacionadas à regulamentação) desconsideradas. Dessa forma, o uso dessas propostas não permite cobrir todo o possível conjunto de restrições identificadas.

Com relação ao suporte para realizar a implantação, um dos entraves visualizados é que as dependências entre os serviços não são facilmente identificadas em alguns casos. Por exemplo, nas linguagens baseadas em BPMN para identificar com quais outros participantes um certo membro interage é preciso percorrer tarefa a tarefa, o que pode tornar a avaliação das restrições ineficiente e dificultar as análises que se baseiam em uma visão global do processo. Outra informação que não é facilmente obtida é a possibilidade de compartilhamento de participantes, uma vez que modelos são exclusivos para cada processo.

Por fim, nenhuma das linguagens consideradas permite especificar componentes associados à avaliação das restrições, como indicar a implementação de uma funcionalidade que pode ser utilizada para estimar as propriedades relacionadas à verificação de uma determinada restrição. Apesar de ser natural assumir que a especificação dessas informações está fora do escopo da linguagem de modelagem, seu fornecimento pode ser útil nas decisões de implantação. Isso se dá porque nos casos em que essas informações não são especificadas se deve assumir que os valores das propriedades são previamente conhecidos ou que um mesmo componente é sempre usado para cada propriedade. Contudo, o uso de componentes únicos preestabelecidos

pode ser problemático. Por exemplo, ao especificar uma restrição relacionada à latência para processar uma determinada requisição, não é possível garantir para todos os casos quais variáveis estão envolvidas nessa estimativa e nem qual a técnica mais apropriada para obtê-la, uma vez que algumas técnicas podem não oferecer a precisão necessária em alguns casos, ao passo que são suficientes em outros.

Gerenciamento de recursos

Conforme discutido no capítulo anterior, o principal problema das ferramentas atualmente disponíveis para gerenciamento de recursos em nuvem é que elas não possibilitam, de maneira eficaz, que a estimativa e o provisionamento de recursos sejam realizadas considerando diferentes provedores. Em contrapartida, a melhor estratégia para favorecer a satisfação de todas as restrições requisitadas e obter índices melhores de desempenho e menor custo é utilizar múltiplos provedores de nuvem. Diante disso, alguns trabalhos buscam oferecer soluções nesse cenário, contornando uma ou mais limitações das ferramentas comerciais. A seguir, são apresentados algumas dessas abordagens, para os quais as aplicações constituem componentes de software ou serviços isolados, que são independentes uns dos outros. As especificidades do gerenciamento de recursos para composições de serviços serão discutidas na próxima seção.

Um dos grandes desafios no provisionamento automatizado de recursos é abstrair detalhes da interação com o provedor de recursos, o que é ainda mais complexo ao se considerar o uso de múltiplos provedores pois, nesse caso, é preciso lidar com o dinamismo típico de um ambiente multinuvem. Uma técnica comumente adotada para prover essa abstração e facilitar as atividades de gerenciamento de recursos é utilizar modelos em tempo de execução (model@runtime, ou simplesmente m@rt) que consiste em uma abstração do sistema que é manipulada em tempo de execução para um propósito

específico (Bencomo *et al.,* 2013). Um m@rt pode ser definido como uma autorrepresentação causalmente conectada do sistema que representa sua estrutura e comportamento ou objetivos do ponto de vista do espaço do problema. Isso significa que o sistema "conhece" a si mesmo e que mudanças nos modelos são propagadas para o sistema em execução e vice-versa. Com isso, adaptações em tempo de execução podem ser realizadas nos modelos sem a necessidade de se conhecer a implementação do sistema em si.

Nas abordagens baseadas em m@rt, os modelos passam a fazer parte significativa do sistema em questão. Assim, é possível o projeto, a evolução e a verificação de sistemas de *software* de maneira contínua (Blair; Bencomo; France, 2009). Além da vantagem de conferir uma maior adaptabilidade ao sistema, o uso dessa abordagem facilita tomadas de decisões, permite tarefas mais precisas de validação e monitoramento, além de melhorar a sincronização entre os artefatos de projeto e a implementação do software (Giese; Wagner, 2006). Um exemplo de abordagem nesse sentido é *Communication Virtual Machine* (CVM) (Deng *et al*., 2008), uma máquina de execução de modelos usada para apoiar a coordenação automática de serviços de comunicação centrados no usuário, baseados no uso de modelos em tempo de execução.

Entre os trabalhos que se baseiam em m@rt (Chauvel *et al.*, 2013; Ferry *et al.*, 2013a, 2013b), *Cloud Modeling Framework* (CloudMF) provê uma *Domain-Specific Modeling Language* (DSML) para implantar aplicações em sistemas multinuvem, que torna o processo mais simples em tempo de execução. Contudo, no CloudMF não há preocupação em satisfazer de forma ampla as restrições impostas sobre a aplicação, uma vez que a principal motivação em se usar um ambiente multinuvem, nesse caso, é dispor de mais opções de recurso para atender a uma capacidade de hardware preestabelecida pelo usuário. Dessa forma, é o usuário que deve especificar a capacidade dos recursos necessários para a aplicação, além de regras para

seu provisionamento. O problema em delegar essa escolha ao usuário é que os recursos podem ser super ou subdimensionados, além de dificultar a implantação da aplicação ao incluir uma tarefa adicional.

Outra limitação no CloudMF é o fato de usarem apenas nuvens públicas, fazendo com que a solução não se beneficie de recursos próprios já existentes. O CloudMF é um subprojeto do *MOdel-Driven Approach for design and execution of applications on multiple Clouds* (MODAClouds) (Ardagna *et al*., 2012; Modaclouds, 2023), que objetiva prover um sistema de suporte a decisões, uma IDE *open source* e um ambiente de tempo de execução para o projeto, a prototipação, a geração de código e a implantação automática de aplicações com garantias de QoS. Contudo, mesmo com as extensões consideradas no projeto ainda há limitações, como delegar ao usuário a responsabilidade por definir toda a infraestrutura a ser utilizada.

Open source API and Platform for multiple Clouds (mOSAIC) (Mosaic Cloud, 2022; Petcu *et al*., 2013; Sandru; Petcu; Munteanu, 2013) é um projeto no qual é proposta uma API *open source* e uma plataforma para desenvolvimento de aplicações multinuvem. Seu objetivo é prover abstração para que serviços em nuvem possam ser facilmente desenvolvidos e implantados, utilizando um sistema multiagente (Sandru; Venticinque, 2013) e computação autonômica. Nesse projeto também é proposta uma ontologia usada na anotação de recursos com propriedades funcionais e não funcionais (Moscato *et al*., 2011). Para auxiliar na estimativa da infraestrutura, mOSAIC oferece, também, um arcabouço para o desenvolvimento de *benchmarks* customizados para medir o desempenho da aplicação sobre cargas preestabelecidas. Contudo, esses *benchmarks* devem ser criados de maneira *ad-hoc* para cada aplicação.

A abstração e dinamicidade obtida com o uso de modelos em tempo de execução não é suficiente para resolver o problema de seleção de recursos. Esta atividade é tratada de maneira isolada em alguns trabalhos, usando processo analítico hierárquico (*Analytic*

Hierarchy Process – AHP) (Saaty, 1990). AHP é um método amplamente usado para resolver problemas relacionados à tomada de decisão a partir de múltiplos critérios, o que é definido como *Multiple Criteria Decision Making* (MCDM) (Zeleny, 2012). Este método consiste em decompor problemas complexos e mal estruturados, organizando os fatores que influenciam na decisão a ser tomada em uma estrutura hierárquica formada por subproblemas que podem ser analisados independentemente. Uma vez que esta estrutura é construída, o resultado é obtido pela avaliação sistemática de seus elementos, comparando-os uns aos outros, par a par, com relação ao seu impacto sobre um elemento acima deles na hierarquia. Um peso ou prioridade é derivado para cada elemento da hierarquia, permitindo que elementos diversos e muitas vezes incomensuráveis sejam comparados entre si de uma maneira racional e consistente (Saaty, 2008). O método pode ser resumido como:

1. Inicialmente é feita a modelagem do problema como uma hierarquia contendo o objetivo da decisão, as alternativas para alcançá-lo e os critérios para avaliar as alternativas.
2. O responsável pela decisão indica a significância relativa entre os atributos. Por exemplo, no cenário aqui considerado, o responsável pela decisão pode preferir o custo de utilização do recurso sobre a localização e esta sobre a reputação do provedor.
3. Similarmente, para cada atributo e par de alternativas, o responsável pela decisão especifica suas preferências (por exemplo, se a localização da alternativa *A* é melhor que a da *B*).
4. As preferências são então sintetizadas para produzir um conjunto de prioridades gerais para a hierarquia. Nessa etapa, também é verificada a consistência das preferências.
5. A decisão final é obtida com base nos resultados desse processo.

A SMICloud (Garg; Versteeg; Buyya, 2013) usa AHP para comparar parâmetros de diferentes provedores de nuvem por meio de um método de classificação que reduz o custo na utilização de VMs. O trabalho propõe um conjunto de métricas para sistematicamente medir todos os atributos de QoS propostos pelo *Cloud Service Measurement Index Consortium* (CSMIC) (Adobe, 2022), um consórcio lançado pela Carnegie Mellon University com o intuito de desenvolver o *Service Measurement Index* (SMI). O SMI é um arcabouço que padroniza um conjunto de características, atributos e medidas que tomadores de decisão podem aplicar para permitir a comparação de múltiplos provedores de nuvem. Usando esse arcabouço, é feito o ranqueamento de recursos em nuvem com base nos atributos estabelecidos.

Outra proposta baseada em AHP é CloudGenius (Menzel; Ranjan, 2012), construído sobre o arcabouço (MC2)2 (Menzel; Schönherr; Tai, 2013). Esse sistema automatiza o processo de tomada de decisão baseando-se em um modelo para migração de servidores Web para a nuvem, que estabelece o peso para cada parâmetro da decisão. Para a seleção e a combinação de soluções, CloudGenius constrói um modelo formal que descreve requisitos, além de atributos numéricos e não numéricos.

O uso de AHP em ambas as abordagens requer uma quantidade significativa de dados que devem ser fornecidos pelo usuário para priorizar os diferentes requisitos. Outra limitação em usar essas propostas é o fato de elas não constituírem estratégias de emparelhamento que leva em consideração a relação de um serviço (ou uma aplicação) com outros, ignorando a influência que pode haver entre elas. Essas abordagens apenas constituem uma forma de filtrar recursos usando critérios que devem ser estabelecidos pelo usuário. Em outras palavras, elas apenas auxiliam na seleção de recursos, o que é realizado de maneira restrita, pois não se considera o uso de nuvem privada e a seleção ocorre de forma isolada para cada serviço. As demais atividades de gerenciamento de recursos estão fora do escopo dessas abordagens. Outra limitação é não oferecer soluções para as demais atividades de gerenciamento de recursos.

Dentre os trabalhos que usam técnicas diferentes de AHP para selecionar recursos (Sundareswaran; Squicciarini; Lin, 2012), destacam-se aqueles que usam um modelo de *broker* de nuvem que extrai informações de provedores e as indexam em uma estrutura chamada de Índice *Cloud Service Provider* (CSP). Essa estrutura é usada para realizar uma busca eficiente a consultas de usuários, considerando um número grande de provedores. A partir da consulta, um número predeterminado de recursos candidatos é selecionado. Esse resultado é, então, refinado e reduzido usando um algoritmo que os autores demonstraram ser até 100 vezes mais eficiente que uma estratégia de busca exaustiva. Apesar disso, o foco do trabalho é o desempenho na obtenção dos resultados, não considerando outros critérios, como o balanceamento ou interferência de resultados sobre múltiplas consultas.

Wright *et al*. (2012) propuseram um modelo baseado em restrições para especificação e seleção de recursos em ambientes com múltiplas nuvens públicas. Da mesma maneira que a proposta aqui apresentada, o modelo é focado na aplicação, de forma que os recursos são especificados usando uma linguagem independente de provedor. Uma camada de abstração é usada para descobrir os recursos de infraestrutura mais adequados para uma aplicação (Sun *et al*., 2012), com base em uma especificação pré-elaborada.

O projeto InterCloud (Buyya; Ranjan; Calheiros, 2010) é uma das primeiras iniciativas a explorar QoS em um ambiente multinuvem. Ele estende esforços anteriores do projeto InterGrid para permitir compartilhamento de recursos entre provedores de nuvem (Clouds Laboratory, 2022; Di Costanzo; De Assunção; Buyya, 2009). A arquitetura é centralizada em uma entidade chamada *Cloud Exchange* (CEx), que age como um *marketplace*, em que provedores podem vender recursos. Os compradores podem ser outros provedores ou usuários desejando implantar aplicações. VMs podem ser implantadas em diferentes tipos de recurso, como infraestruturas privadas ou nuvens públicas.

Como pode ser notado, esses trabalhos objetivam apenas a seleção de recursos, assumindo que a estimativa da infraestrutura que será utilizada foi previamente realizada. Outra suposição assumida é que a QoS obtida ao selecionar um determinado recurso também é conhecida, não sendo possível ao usuário especificar estratégias de estimativa de QoS que sejam mais adequadas para o cenário considerado. Para permitir gerenciamento de recursos de maneira automatizada, é preciso, também, considerar as atividades relacionadas à estimativa de recursos e, consequentemente, de QoS. Entre as técnicas utilizadas para obter essa estimativa, está a adotada no projeto mOSAIC, ou seja, estabelecer *benchmarks* customizados para cada aplicação. Outra técnica é utilizar estratégias de predição de uso para atributos de hardware específicos (Lazowska *et al.*, 1984; Mello; Yang, 2009; Yang; Foster; Schopf, 2003). Entre os trabalhos que consideram a estimativa de recursos, Wu, Garg e Buyya (2011) propõem uma estratégia de mapeamento dos requisitos solicitados pelo usuário para parâmetros de infraestrutura, visando minimizar o custo da infraestrutura e das violações de SLA. Contudo, a estimativa é somente com relação à quantidade de recursos de um tipo específico, assumindo, também, que a QoS é conhecida.

Discussão

Nesta seção, é apresentado um comparativo dos trabalhos discutidos. Embora os trabalhos considerados possam ter objetivos distintos, buscou-se identificar as similaridades com a proposta aqui apresentada. Nesta análise não foram incluídos aspectos que são relevantes apenas para composições de serviços, pois estes serão discutidos na próxima seção. No Quadro 4, os critérios considerados são divididos em quatro categorias: escopo, critérios de provisionamento, abstração e características não funcionais da solução.

QUADRO 4
Comparativo das principais propostas para gerenciamento de recursos (considerando apenas serviços isolados)

	MODA Clouds	mOSAIC	SMI Cloud	Cloud Genius	Índice CSP	Sun *et al.*	Inter Cloud	Wu *et al.*
Escopo								
Estimativa de recursos		√						
Descoberta de recursos	√	√				√	√	
Seleção de recursos	⌀	√	√	√	√	√	√	√
Alocação de recursos	√	√		√			√	√
Uso de nuvem privada		√						⌀
Uso de múltiplas nuvens públicas	√	√	√	√	√	√	√	⌀
Considera a localidade do recurso	⌀	⌀	⌀	√	√	√	⌀	⌀
Critérios de provisionamento								
QoS		√	√	√	√		√	√
Restrições relacionadas ao ambiente de implantação (localidade do recurso, reputação do provedor etc.)		√	√	√	√	√		
Minimização do custo		⌀	⌀	⌀	⌀		⌀	√
Maximização da eficiência								√
Abstração								
Independência de aplicação	√	√	√	√	√	√	√	√
Independência de ambiente de nuvem	√	√	√	√	√	√	√	√
Manipulação de modelos de recursos					√			
Isolamento da estimativa de QoS								
Características não funcionais da solução								
Flexibilidade	⌀	√	×	√	×	×	√	√
Manutenibilidade	√	⌀	⌀		√		√	√
Usabilidade	√	√			√	√	√	√
Escalabilidade	√	√				⌀		

Legenda: √ Contemplado, ⌀ Contemplado com limitações, × Fora do escopo do trabalho
Fonte: Elaboração própria.

O escopo indica alguns aspectos relevantes em uma solução que considera as principais atividades relacionadas ao gerenciamento de recursos, desde sua estimativa até a alocação do recurso no ambiente selecionado. Os critérios de provisionamento indicam os parâmetros propostos para instrumentar as decisões relacionadas ao provisionamento de recursos. A abstração indica critérios que são essenciais para tratar a complexidade de ambientes de nuvem, o que inclui mecanismos para lidar de forma eficiente com a multiplicidade de tipos disponíveis e com a estimativa customizável de QoS. Portanto, as características não funcionais da solução relacionam alguns critérios que caracterizam a qualidade da solução proposta: a) flexibilidade: mostra se há independência do mecanismo de decisão do restante da plataforma; b) manutenibilidade: estabelece a capacidade de evoluir de acordo com novos cenários, c) usabilidade: sugere se o provisionamento é realizado sem que o usuário tenha que fornecer informações além daquelas essencialmente necessárias e; d) escalabilidade: indica se a abordagem considera a adição de instâncias para os serviços como estratégia para absorver cargas elevadas. Como pode ser visto, nenhuma solução contempla todos os aspectos considerados.

Relativamente ao escopo, apenas mOSAIC implementa todas as atividades que permitem uma completa automatização do gerenciamento de recursos considerando múltiplos provedores. Os demais trabalhos focam essencialmente na seleção de recursos, dado um conjunto de critérios. Grande parte das propostas assume que a estimativa e a descoberta de recursos são realizadas como atividades independentes, pelo próprio usuário ou usando alguma ferramenta desenvolvida em outro contexto. Com respeito ao ambiente de nuvem considerado, o uso de múltiplos provedores é considerado em todas as soluções, embora a exploração da heterogeneidade obtida com a disponibilidade de múltiplas regiões de um mesmo provedor seja tratada apenas de maneira superficial. Contudo, poucas soluções exploram os casos em que é usada também

uma infraestrutura privada, não considerando este cenário muito encontrado nas organizações.

Entre os critérios de decisão utilizados, a maioria se baseia em QoS e também permite a definição de restrições de outras categorias, como localidade do recurso. Foi verificado se o custo dos recursos é considerado nas decisões tomadas e a maioria dos trabalhos analisados o avalia como um atributo de QoS. Embora essa categorização seja comum, considera-se que o custo dos recursos tem um impacto relevante no resultado fornecido, uma vez que o usuário, no cenário tratado, é o provedor de serviços, que visa maximizar o lucro ao oferecer soluções.

Também foram verificados os casos em que o provisionamento de recursos é realizado visando o uso eficiente dos recursos. Esse aspecto só é tratado de maneira eficaz na proposta de Wu, Garg e Buyya (2011) que considera a consolidação de recursos. A seleção de recursos nos demais trabalhos ignora a possibilidade de haver capacidade excedente no que foi selecionado, o que faz com que a capacidade residual não seja aproveitada.

Quanto à abstração, quase todos os trabalhos se baseiam no uso de modelos para tornar a solução independente de aplicação ou de provedor de nuvem específico. Contudo, com exceção do Índice CSP que faz processamento do modelo de recursos visando melhorar o tempo de obtenção do resultado, nenhuma outra considera alguma manipulação do modelo de recursos visando melhor desempenho ou resultados mais satisfatórios. Isso se caracteriza como uma necessidade, uma vez que avaliar todo o vasto conjunto de tipos de recursos disponíveis em um ambiente multinuvem pode demandar um tempo não aceitável em alguns cenários. De maneira complementar, nenhum trabalho considera tornar a estimativa de QoS algo customizável, o que pode limitar a solução nos casos em que a estimativa disponível está desatualizada ou não é adequada para a aplicação que estiver sendo implantada.

Sobre as características não funcionais da solução, é possível verificar um bom nível de flexibilidade e manutenibilidade.

Porém, a usabilidade não é ideal em todos os casos. Em algumas soluções, além da especificação dos componentes de software a serem implantados e restrições associadas, o usuário também deve fornecer um conjunto (muitas vezes exaustivo) de informações que são usadas como critérios de decisão, como o peso das restrições.

Implantação de composições de serviços

A criação e a subsequente implantação de composições de serviços em ambientes de nuvem devem considerar a seleção de serviços e de recursos. A seleção de serviços consiste em escolher instâncias concretas de serviços que implementam as tarefas abstratas definidas na composição, o que é realizado geralmente com base na QoS garantida a cada instância. Todavia, a nomenclatura adotada em ambientes de nuvem considera que qualquer entidade oferecida ao usuário (seja ela referente a software ou a hardware) é denominada como serviço. Assim, a seleção de serviços e de recursos são comumente agregadas em um único termo denominado seleção de serviços. Nesse sentido, assume-se que os recursos necessários estarão disponíveis, de forma que a QoS obtida com cada instância seja garantida. Contudo, as escolhas no provisionamento de recursos afetam diretamente a seleção de serviços (Ye; Zhou; Bouguettaya, 2011).

Entre os trabalhos que tratam da seleção de serviços (sem considerar explicitamente os recursos), Alrifai e Rissee Nejdl (2012) propõem uma estratégia para decompor restrições QoS fim-a-fim em restrições locais com limites conservadores superiores e inferiores. Elas são resolvidas usando uma estratégia de seleção distribuída. Os autores Wang *et al*. (2017) apresentam um modelo para lidar com propriedades não funcionais quantitativas e qualitativas e dois algoritmos para seleção de serviços com base nesse modelo. O primeiro algoritmo combina otimização global com seleção local e o segundo consiste em um algoritmo genético. Wu *et al*. (2014) propõem uma estratégia que se baseia em selecionar serviços sob demanda,

de forma a escolher aqueles que oferecem QoS mais próxima da requisitada, ao contrário da estratégia comumente adotada de selecionar serviços com maior QoS (o que os autores chamam de "melhor-esforço"). Em todos estes trabalhos, ao assumir uma QoS garantida com o provisionamento de recursos, eles não conseguem explorar alguns aspectos que são possíveis ao utilizar recursos como entidades de primeira classe na solução, como a variação da QoS obtida usando a elasticidade provida em ambientes de nuvem.

O trabalho de Ye, Zhou e Bouguettaya (2011) é um dos poucos trabalhos que consideram a distinção entre seleção de serviços e de recursos. Eles propõem um algoritmo genético para fazer a seleção de recursos, no qual escolhas aleatórias são usadas para selecionar cromossomos para executar uma operação de *crossover*, tendo como base a QoS obtida sobre os recursos e a demanda de comunicação entre eles. Além da seleção de recursos, os autores também propõem uma estratégia de escalonamento de tarefas da composição, o que não é necessário no caso de coreografias, uma vez que a própria coordenação entre os serviços se encarrega dessa atividade. Uma vantagem dessa proposta é que eles consideram o provisionamento de recursos em três níveis: VM, banco de dados e rede, ao passo que no modelo aqui proposto foi considerado apenas VMs. No entanto, não está claro se essa abordagem permite a implantação visando atender a uma QoS requerida, ou apenas selecionar recursos predeterminados. Outra limitação é que a seleção de recursos é realizada para cada composição isoladamente, sem levar em consideração os efeitos causados pelo compartilhamento de serviços.

Trabalhos em áreas como escalonamento de *workflows* (Yu; Buyya; Tham, 2005, 2006) apresentam características semelhantes ao problema aqui tratado, como o objetivo de balancear a contribuição de cada serviço presente no *workflow* para satisfazer restrições fim-a-fim. Contudo, não foi incluída grande parte desses trabalhos porque eles consideram algumas suposições que não fazem sentido na implantação de coreografias. A principal delas é que a composição

por meio de *workflows* geralmente é estática, não contemplando todos os desafios aqui tratados. Ademais, o escalonamento das tarefas é coordenado por um agente central, o que só é válido no modelo de orquestração.

A QuARAMRecommender (Soltani; Martin; Elgazzar, 2014), parte do arcabouço QuARAM (Martin *et al*., 2013), é uma plataforma de seleção autoadaptativa que recomenda uma lista de recursos para implantação de aplicações em nuvem, tendo como base requisitos das aplicações e das preferências dos usuários. O processo de recomendação se inicia tentando identificar em uma base de dados casos semelhantes à requisição submetida. Depois de recuperar os casos semelhantes, o sistema de recomendação os envia, com aqueles especificados, para um adaptador que busca fazer modificações até que a seleção seja o mais próximo possível do que foi requisitado.

Economia de recursos é outro objetivo desse trabalho. Para tal, os autores propõem consolidação de recursos como forma de diminuir o número de instâncias necessárias. A consolidação é realizada com base nas preferências do usuário e atributos de custo e desempenho dos recursos, mas sem levar em consideração a demanda de comunicação entre os serviços implantados nas entidades consolidadas. Os resultados apresentados indicam que o sistema atinge uma precisão de recomendações em 71% dos casos. Contudo, apesar de este trabalho considerar que uma aplicação pode ser formada por diferentes entidades, as formas de interação entre elas não são consideradas. Isso faz com que essa abordagem não seja ideal para coreografias, dado que há diferentes padrões de composição nesse modelo, que podem influenciar na satisfação de restrições fim-a-fim. Este trabalho também não considera os efeitos do compartilhamento de serviços, podendo inviabilizar os resultados para casos em que isso ocorre.

Charrada *et al*. (2012) propõem o uso de nuvens híbridas para a implantação de serviços. Eles apresentaram um algoritmo que obtém solução próxima da ótima, tendo como base os custos de comunicação e de utilização dos recursos. O algoritmo consiste em

mover serviços que não podem ser alocados na nuvem privada para nuvem pública. De maneira complementar, serviços implantados na pública são movidos para a privada quando há recursos disponíveis. O balanceamento entre os dois ambientes de nuvem também é considerado ao implantar novos serviços. Contudo, a análise realizada no algoritmo é bem limitada pois considera apenas a demanda de recurso e a minimização dos custos de comunicação, não sendo possível estabelecer outros critérios ou restrições.

Mao *et al.* (2013) modelam a implantação de serviços como um jogo de congestionamento (Milchtaich, 1996), em que cada serviço, formado por múltiplos componentes, é considerado como um jogador que compartilha recursos com outros. A estratégia do jogador é então considerada como selecionar o subconjunto de recursos para implantar seus componentes. Com base no jogo de congestionamento, um método teórico de jogo é proposto para otimizar tanto o custo global quanto a qualidade. Para resolvê-lo, algoritmos são propostos pelos autores para alcançar o equilíbrio em tempo polinomial. Contudo, a solução é limitada ao uso dos critérios de tempo de execução e custo de implantação. Outra barreira é o reconhecimento dos recursos disponíveis como homogêneos, o que não retrata um ambiente real de nuvem (especialmente com múltiplos provedores).

Ye, Bouguettaya e Zhou (2012) consideram o provisionamento de recursos com restrições variáveis. Um modelo econômico baseado em Redes Bayesianas discretas é apresentado para caracterizar o comportamento dos usuários a longo prazo (Jensen, 1996). Em seguida, o problema de implantação de serviços com base em QoS é resolvido por Diagramas de Influência (Shachter, 1988), seguido por experimentos analíticos e simulação. De acordo com os autores, os atributos funcionais e de QoS em uma requisição de longo prazo são variáveis. Por exemplo, o usuário pode preferir um recurso com maior desempenho no primeiro ano, mas, a partir do segundo, pode passar a privilegiar recursos com menor custo para diminuir seus gastos.

Na proposta deste livro, considera-se como trabalho futuro a possibilidade de o usuário fazer adaptações nas restrições para coreografias previamente implantadas. Contudo, é necessário que mudanças sejam explicitamente requisitadas, pois a variabilidade não é tratada como entidade de primeira classe, da forma como é proposto por Ye, Bouguettaya e Zhou (2012). Em contrapartida, uma limitação desse trabalho é que eles também consideram que os recursos são homogêneos.

Huang e Shen (2015) propõem uma estratégia para implantar composições de serviços visando reduzir o tempo de execução, ao levar em conta os custos de comunicação entre os serviços e paralelismo entre as tarefas executadas. A estratégia se baseia na modelagem dos custos de comunicação entre os serviços, usando uma estrutura chamada grafo de dependências entre serviços (*Service Dependency Graph* – SDG) e do paralelismo em outra estrutura chamada grafo de concorrência de serviços (*Service Concurrence Graph* – SCG). Esses dois grafos são integrados em uma única estrutura com o nome de grafo de relacionamento entre serviços (*Service Relationship Graph* – SRG), e o problema de implantação de serviços é resolvido usando o problema k-corte mínimo (Guttmann-Beck; Hassin, 2000). A limitação desse trabalho é que ele não tem como objetivo satisfazer restrições estabelecidas pelo usuário, sendo que procura unicamente melhorar o tempo de execução. Apesar de considerarem a carga suportada ao realizar a consolidação de recursos, os autores não resolvem o problema nos casos em que instâncias únicas dos serviços são insuficientes para atender a demanda, mantendo esse aspecto como trabalho futuro. Outra limitação é que também consideram que os recursos são homogêneos.

Amato e Moscato (2017) se baseiam em soluções de uma área de pesquisa denominada orquestração de recursos (*Resource Orchestration* – RO) (Liu *et al.*, 2011; Wieder *et al.*, 2010), que desenvolve novos mecanismos para gerenciamento de recursos (principalmente alocação) visando QoS. Nesse trabalho, os autores mostraram como uma descrição baseada em padrões pode ser usada para

coordenar serviços compostos, considerando todas as categorias de serviço em nuvem (SaaS, PaaS e IaaS). A metodologia explora técnicas de transformação de modelos para construir modelos formais que são usados para a análise de propriedades da orquestração. O objetivo do trabalho é a definição da linguagem formal usada para orquestrar os recursos, sendo que decisões sobre a implantação são tratadas de maneira secundária e superficial.

O componente *Enactment Engine* (EE) (Leite, 2014; Leite *et al.*, 2013a), desenvolvido dentro do contexto do projeto CHOReOS (Vincent *et al.*, 2010), consiste em um sistema de *middleware* que fornece uma plataforma como serviço (PaaS) para a implantação distribuída e automatizada de coreografias de serviços Web de grande escala em ambientes de nuvem. CHOReOS EE recebe uma especificação declarativa da coreografia, realiza sua implantação e devolve ao usuário informações sobre a localização de cada serviço implantado. A especificação da coreografia é uma descrição arquitetural, que deve ser desenvolvida pelo usuário que faz a implantação. A consideração de aspectos de QoS, tanto na seleção dos recursos quanto no monitoramento da aplicação, foi proposta, mas não implementada. Outro entrave é que a especificação dos recursos a serem utilizados deve ser feita pelo usuário, não havendo estimativa automática de recursos. No trabalho aqui apresentado este componente foi utilizado como parte da implementação da etapa de alocação de recursos.

Discussão

Como nas seções anteriores, é apresentado um comparativo dos trabalhos discutidos considerando os aspectos mais relevantes do problema. De maneira especial, são consideradas as mesmas categorias apresentadas na análise dos trabalhos sobre gerenciamento de recursos para serviços isolados, apresentada na seção anterior. Contudo, foram acrescentados outros critérios que são exclusivos de composições de serviços. O Quadro 5 apresenta os resultados dessa análise.

QUADRO 5

Comparativo das principais propostas para gerenciamento de recursos (considerando composições de serviços).

	Ye *et al.*	QuARAM Rec.	Charrada *et al.*	Mao *et al.*	Ye *et al.*	Huang e Shen	Amato e Moscato	CHOReOS EE
Escopo								
Estimativa de recursos		∅						
Descoberta de recursos							∅	
Seleção de recursos	√	√	√	∅	√	∅	∅	∅
Alocação de recursos								√
Restrições fim-a-fim	√						∅	
Restrições locais	√	√	∅	∅	√		∅	∅
Uso de nuvem privada	∅		√	∅	∅	∅	∅	√
Uso de múltiplas nuvens públicas	∅	√	∅	∅	∅	∅	∅	√
Implantação conjunta de múltiplas composições				√		√		
Critérios de provisionamento								
QoS	√	√		∅	√	∅	√	∅
Restrições relacionadas ao ambiente de implantação (localidade do recurso, reputação do provedor etc.)	∅	√						
Minimização do custo	∅		√			√		
Minimização do atraso de comunicação	∅		√			√		
Maximização da eficiência		√	√	√		√		
Abstração								
Independência de aplicação	√	√	√	∅	√	√	√	√
Independência de ambiente de nuvem	√	√	√	√	√	√	√	√
Manipulação de modelos de recursos		√						
Isolamento da estimativa de QoS			×	×		×	×	√
Considera diferentes padrões de composição (não apenas sequencial)	√				√	∅	√	∅
Características não funcionais da solução								
Flexibilidade	×	√	√	×	×	×	×	∅
Manutenibilidade	√	√	∅	∅	√	∅	∅	
Usabilidade	∅		√	√		√	√	√
Escalabilidade			√					

Legenda: √ Contemplado, ∅ Contemplado com limitações, × Fora do escopo do trabalho
Fonte: Elaboração própria.

Com relação ao escopo, são assumidas suposições semelhantes às estabelecidas nos trabalhos da seção anterior. O foco dos trabalhos considerados está na seleção de recursos sujeita a restrições locais (que se referem apenas a serviços específicos), sendo as demais atividades de gerenciamento de recursos tratadas apenas pontualmente em algumas abordagens. Com isso, ao adotá-las, o usuário deve realizar a quebra de restrições fim-a-fim em restrições específicas sobre cada serviço para que a estratégia de seleção de recursos proposta seja utilizada; além de executar manualmente (ou considerando outra abordagem) as demais atividades de gerenciamento de recursos.

A maioria dos trabalhos descreve recursos de maneira genérica não incluindo atributos de algum provedor ou ambiente específico. Nesse caso, pode-se assumir que a proposta teoricamente seria válida para qualquer provedor e mesmo para uma nuvem privada. Contudo, ao admitir esse tipo de abordagem e não incluir tratamentos que tem relação com os atributos do recurso concreto utilizado, a solução não pode ser considerada completa pois algumas suposições podem invalidar o resultado. Por exemplo, a quantidade limitada de recursos em uma nuvem privada faz com que a suposição de disponibilidade de recursos não possa ser aplicada nesse caso.

Com relação ao compartilhamento de serviços entre múltiplas composições e o impacto disso na implantação, apenas dois trabalhos avaliados se interessam por esse aspecto. O grande problema em ignorá-lo é que a QoS esperada para cada serviço só será garantida se houver uma instância desse serviço dedicada para cada composição, o que pode representar desperdício devido à subutilização de algumas instâncias.

Quase todos os trabalhos consideram apenas restrições de QoS na implantação de composições, sendo o custo de uso dos recursos considerado explicitamente só em alguns casos. A maioria dos trabalhos ignoram restrições relacionadas a atributos do ambiente de implantação dos serviços, o que constitui uma grande limitação, já que muitas das restrições atualmente impostas sobre serviços

dizem respeito a essa categoria. Ademais, devido à possível demanda elevada de comunicação entre os serviços da composição que está sendo implantada, a seleção de recursos deveria considerar, também, esse aspecto, usando estratégias como a seleção de recursos em um mesmo ambiente ou com maior proximidade física possível.

A localidade dos recursos alocados só é levada em consideração no QuARAMRecommender e na proposta de Huang e Shen (2015), embora nesse último a localidade não seja tratada explicitamente, pois se considera apenas dados já disponíveis sobre o atraso de comunicação entre os serviços.

Um dos grandes problemas encontrados é que algumas soluções consideram que os recursos são homogêneos, fazendo com que a satisfação de restrições seja limitada ao tipo adotado como padrão. Essa abordagem não é realista ao considerar um ambiente com múltiplos provedores de nuvem, em que há uma pluralidade grande de tipos. Essa pluralidade deveria ser considerada, também, ao propor o modelo de representação dos recursos, de forma que alguma estratégia de manipulação desse modelo fosse empregada para melhorar o desempenho das análises realizadas a partir dele. Estratégia desse tipo só é proposta no QuARAMRecommender.

A existência de diferentes padrões de interação entre serviços não é considerada em alguns trabalhos, sendo o tratamento nesses casos realizado somente para fluxos sequenciais. Essa abordagem torna a solução não aplicável em cenários realistas, nos quais diferentes valores de QoS podem ser obtidos ao considerar fluxos executados em paralelo ou que são acionados com base em alguma condição ou evento.

Por fim, as soluções propostas consideram apenas o uso de instâncias únicas dos serviços, com exceção da proposta de Charrada *et al.* (2012), que aloca novas instâncias dos serviços para absorver demandas mais elevadas. A grande deficiência dos trabalhos que assumem esse posicionamento é que eles passam a ter um limite de carga suportada, acima do qual a proposta não oferece nenhuma solução.

Neste capítulo, foram apresentados os principais trabalhos relacionados à abordagem proposta. Inicialmente, foi discutido as propostas para modelagem de coreografias de serviços com restrições não funcionais associadas. Grande parte das linguagens propostas são extensões à BPMN e focam na especificação de restrições de QoS impostas sobre serviços específicos. De acordo com a análise realizada, além da deficiência na especificação de restrições, outro grande problema das notações atualmente disponíveis é o fato delas não permitirem uma análise imediata das dependências entre os serviços e, principalmente, a especificação de compartilhamento de um mesmo serviço entre múltiplas composições.

Também foram analisados trabalhos sobre provisionamento de recursos, considerando duas categorias. Inicialmente foram avaliadas propostas para aplicações que são formadas por serviços isolados e depois foram considerados trabalhos voltados para composições de serviços. As conclusões obtidas em ambas as categorias foram semelhantes. Foi possível notar que o principal problema discutido é a seleção de recursos em nuvens públicas, visando atender requisitos de QoS. Na maioria dos casos, assume-se que a estimativa de recursos já foi realizada e que a QoS garantida é conhecida, sendo que o uso eficiente de recursos não é geralmente um objetivo. Com relação aos critérios específicos de composições de serviços, é geralmente adotada uma simplificação do problema, assumindo-se que composições são criadas usando instâncias únicas de serviços que interagem apenas sequencialmente.

Diante dessa análise, é possível perceber que há diversas lacunas nas propostas encontradas. Essas limitações dizem respeito principalmente à ausência de uma visão mais pragmática do problema, na qual é assumida a necessidade de a implantação ser realizada considerando o compartilhamento dos serviços entre composições. Para preencher essas lacunas, aqui é proposta uma abordagem que leva em consideração essa questão e as demais limitações discutidas neste capítulo, contemplando todas as atividades do gerenciamento de recursos. Essa contribuição é discutida no próximo capítulo.

3

Representação de múltiplas coreografias de serviços com restrições não funcionais

Anteriormente, foi discutido o estado da arte do tema trabalhado neste livro e foi apresentado, de maneira abstrata, o problema de implantação de coreografias de serviços visando satisfazer restrições não funcionais. Conforme discutido, há alguns trabalhos que apresentam soluções relacionadas a este problema, mas sem considerar todos as necessidades desejadas.

Na proposta desenvolvida, o problema de provisionamento de recursos na implantação de coreografias de serviços foi considerado como um problema de emparelhamento. Mais precisamente, a estimativa e a seleção de recursos são obtidas usando uma estratégia que realiza o mapeamento dos serviços que compõem as coreografias para os recursos disponíveis, de acordo com as restrições não funcionais especificadas. Para tal, é necessário que todos os elementos envolvidos no problema sejam modelados usando uma representação objetiva e não ambígua. Em virtude disso, neste capítulo são formalizados os principais conceitos envolvidos na definição e na solução do problema.

Inicialmente, é apresentada a representação de serviços formalizada junto aos recursos e às restrições não funcionais. Em seguida, é proposta uma notação para a representação de coreografias de serviços e um arcabouço para as restrições não funcionais especificadas. Com base na notação proposta, é definida uma estrutura para a representação conjunta de múltiplas coreografias de serviços, com as restrições não funcionais associadas. A definição dessa estrutura e do procedimento para sua obtenção também faz parte deste capítulo.

Modelando uma coreografia com restrições

O principal objetivo desta seção é definir a formalização de uma notação para representar coreografias de serviços e mostrar um arcabouço proposto para a especificação de restrições não funcionais associadas aos serviços que fazem parte dessas coreografias. Contudo, para isso, primeiramente são formalizados os elementos fundamentais relativos ao problema tratado. Mais especificamente, na próxima seção é formalizada a representação de serviços e recursos e, em seguida, são categorizadas as restrições não funcionais consideradas na abordagem, apresentando a formalização e o processo de avaliação de restrições para cada uma das categorias definidas.

Elementos fundamentais do problema

- S – representa uma coleção de n serviços $\{s_1, s_2, \ldots, s_n\}$;
- O_{s_i} – define o papel que o serviço s_i, $(1 \leq i \leq n)$ pode desempenhar em uma coreografia, pela especificação do conjunto de operações que ele implementa;

- $o \in O_{s_i}$ – é uma operação que pode ser usada ao compor coreografias de serviços e;
- $d[\ ]_o$ – representa um vetor que contém informações usadas para estimar a demanda de recursos para o processamento de uma operação.

A informação representada em $d[\ ]_o$ é limitada à complexidade da operação (expressa em milhões de instruções – MI), ao uso de memória (expressa em *bytes*) e aos dados de saída permanentes (expresso em *bytes*). Para simplificar, é assumido que a largura de banda é grande o suficiente para ignorar o tempo de transmissão na comunicação entre serviços e não são especificadas informações sobre o tamanho dos dados transmitidos. Assume-se ainda que esse vetor é fornecido como metadado na especificação do serviço. Uma estratégia para automatizar a obtenção das informações nele representadas é usar ferramentas de perfilhamento, como VisualVM (2022) e JProfiler (EJ Technologies, 2022), em Java. Contudo, a automação desse aspecto requer um estudo detalhado sobre os serviços coreografados e suas possíveis entradas.

Os dados sobre os recursos disponíveis para serem usados na implantação dos serviços são definidos como:

- V – uma coleção de t de tipos de VM $\{v_1, v_2, \ldots, v_t\}$.
- $\zeta[\ ]_v$ – um vetor que especifica a capacidade de um tipo de VM $v \in V$. Esse vetor contém as seguintes informações:
  - capacidade de processamento: expresso em número de núcleos de CPU e velocidade de *clock* da CPU.
  - memória: expressa em GB.
  - armazenamento: expresso em GB.
- $\iota[\ ]_v$ – um vetor de atributos que caracterizam um tipo de VM $v \in V$, de forma que $\iota[\ ]_v \supset \zeta[\ ]_v$. Em outras palavras, $\iota[\ ]_v$ contém todos os atributos especificados em $\zeta[\ ]_v$, acrescentando as seguintes informações:

- c_v – custo associado ao tipo de VM $v \in V$. Este custo representa o custo monetário de utilização por uma hora de uma instância criada sob demanda com este tipo. No caso desse tipo de VM ser instanciado em uma nuvem privada, o custo de utilização é substituído por uma proporção do custo para manter em execução a máquina física em que o tipo de VM é instanciado, que é estimado pelo seu consumo de energia e demais custos associados.
- l_v – localização do tipo de VM $v \in V$, que equivale ao endereço da região onde esse tipo de VM é disponibilizado. Esse endereço, por sua vez, é formado por um identificador do país e por um complemento (quando disponível) que representa o estado e/ou a cidade na qual a região está instalada. De maneira alternativa, visando maior precisão, o endereço pode ser especificado usando as coordenadas geográficas (latitude e longitude), quando essa informação está disponível (por exemplo, quando o tipo de recurso é instanciado em uma nuvem privada). Contudo, quando se utiliza uma nuvem pública não é possível obter as coordenadas de uma determinada região pois os provedores não divulgam essa informação por questão de segurança.
- p_v – um identificador do provedor de nuvem que disponibiliza o tipo de VM $v \in V$.

Embora $\zeta\left[\square\right]_v$ contenha as principais informações sobre a capacidade de hardware, é evidente que é representada de modo simplificado, pois alguns atributos, como o tamanho da cache e largura de banda, não são representados. O aperfeiçoamento desse modelo, de forma a considerar um conjunto mais amplo de atributos, é tido como trabalho futuro. Argumenta-se que os elementos atualmente representados são suficientes para permitir a implantação dos serviços, dadas as suposições assumidas no cenário considerado.

No problema tratado, o provisionamento de recursos deve ser implementado com o objetivo de satisfazer um conjunto de restrições não funcionais estabelecidas pelo usuário responsável pela implantação da(s) coreografia(s). São distinguidas duas categorias de restrições, como ilustrado na Figura 13. Essa categorização foi estabelecida com base nas restrições comumente impostas na implantação de serviços, como aquelas citadas no exemplo do cenário.

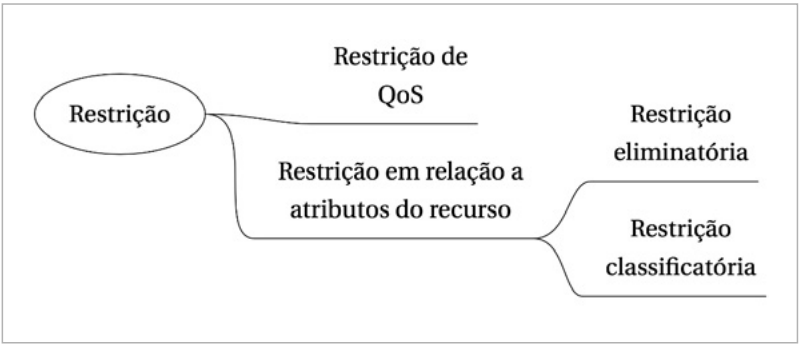


Figura 13 – Taxonomia de restrições
Fonte: Elaboração própria.

Restrições de QoS descrevem dimensões qualitativas que são aplicáveis aos serviços e que são geralmente perceptíveis pelo cliente. Elas são avaliadas usando o conhecimento sobre o contexto de uso, que inclui a capacidade dos recursos e a carga sobre os serviços.

Restrições em relação a atributos do recurso são completamente independentes da implementação dos serviços e da capacidade dos recursos. Elas são especificadas usando características do ambiente de execução, que inclui informações sobre a localização da VM e seu custo, e informações sobre o provedor de nuvem, como sua reputação. Essa categoria foi dividida em duas subcategorias: *restrições eliminatórias*, que são usadas para suprimir tipos de VM na seleção de recursos, usando um valor-alvo para um certo atributo; e *restrições classificatórias*, que ordenam os tipos de VM usando um determinado atributo. As categorias dessa taxonomia são formalizadas adiante neste capítulo.

O Quadro 6 apresenta exemplos de restrições em cada uma das categorias estabelecidas. Essas restrições estão originalmente no exemplo de cenário (Quadro 2).

QUADRO 6
Exemplo de restrições em cada uma das categorias estabelecidas

Categoria	Exemplos de Restrição
Restrição de QoS	O tempo máximo necessário para atender a cada requisição deve ser de 1 segundo. A emissão de relatórios deve manter uma vazão de 10 requisições por segundo.
Restrição Eliminatória	O recurso não pode estar indisponível mais do que 12 horas por ano. A seleção de recursos deve ser realizada de forma a evitar provedores de nuvem com obrigação de cobrança mínima. Dados sobre clientes brasileiros devem ser armazenados no Brasil. O serviço deve ser implantado em uma nuvem privada.
Restrição Classificatória	A implantação de serviços deve ser realizada usando recursos com menor custo financeiro. A seleção de recursos deve privilegiar provedores de nuvem com melhor reputação.

Fonte: Elaboração própria.

Restrições de QoS são avaliadas usando informações sobre a execução do serviço, de acordo com uma métrica de QoS. A definição de métrica de QoS segue a notação especificada por Rosario, Benveniste e Jard (2009).

DEFINIÇÃO 1 (MÉTRICA DE QOS)

Uma métrica de QoS é uma tupla $m = (D, \leq, \oplus, \wedge, \vee)$*:*

- D *– é o domínio da métrica.*
- $\leq$ *– define a ordem dos valores do domínio.*
- $\oplus$ *– é uma função* $\oplus : D \times D \rightarrow D$ *que define como QoS é incrementada. Satisfaz as seguintes condições:*

  - $\oplus$ – *possui um elemento neutro 0 que satisfaz* $\forall l \in D \rightarrow l \oplus 0 = 0 \oplus l = l$.
  - $\oplus$ – *é monotônico:* $l_1 \leq l_1^{'}$ *e* $l_2 \leq l_2^{'}$ *implica* $(l_1 \oplus l_2) \leqslant (l_1^{'} \oplus l_2^{'})$.

- $\wedge$ – *representa o limite inferior, significando que qualquer* $l \subseteq D$ *tem um único limite inferior* $\wedge_l$. *Quando a melhor QoS é obtida de acordo com a ordem* $\leq$ *a melhor QoS possível é obtida com* $\wedge$.
- $\vee$ – *representa o limite superior, significando que qualqu* $l \subseteq D$ *er tem um único limite superior* $\vee_l$. *Ao comparar dois valores da métrica, o operador* $\vee$ *é usado para obter a pior QoS de acordo com a ordem* $\leq$.

De acordo com esta definição, o domínio D considerado varia de acordo com a métrica de QoS. Por exemplo, podem ser números reais positivos R^+ para latência ou um domínio genérico $Q \in$ {baixa, média, alta} para segurança. A ordem parcial nesse domínio, definida pelo operador $\leq$, pode significar "menos é melhor" ($\leq$ em R^+ para latência) ou "mais é melhor" ($\geq$ em R^+ para vazão). A função de agregação $\oplus$ pode significar adição, para métricas como latência, ou a pior QoS, definida pelo operador $\vee$, para métricas com domínio genérico, como segurança.

A estimativa dos valores de QoS é isolada neste trabalho. Esse isolamento foi proposto por meio da definição de funções de utilidade responsáveis por obter os valores de QoS, dado um serviço e o recurso em que este serviço será implantado. O objetivo em adotar essa estratégia é permitir a generalização da técnica usada para obter os valores da métrica e, assim, permitir a personalização dessa técnica de acordo com a aplicação sendo implantada. Dessa forma, uma função de utilidade pode ser definida como:

DEFINIÇÃO 2 (FUNÇÃO DE UTILIDADE)

Uma função de utilidade é uma função $U : S \times V \rightarrow D_m$ *que fornece a estimativa de QoS de acordo com a métrica* m*, quando um serviço* $s \in S$ *é implantado no recurso* $v \in V$*.*

A estimativa de QoS é obtida de acordo com a carga agregada das operações utilizadas no serviço, considerando todas as coreografias em que ele está inserido. Isso faz com que toda a demanda ao serviço seja analisada e, assim, a sobrecarga causada pelo seu provável compartilhamento seja refletida na estimativa de QoS. Esta estratégia permite obter maior eficiência na estimativa e na seleção de recursos do que abordagens que levam em consideração apenas as operações, e suas respectivas cargas, utilizadas em uma única coreografia.

A função de utilidade é implementada de maneira distinta para cada métrica de QoS. Por exemplo, para latência ou vazão pode-se usar a Teoria de Filas (Lazowska *et al.*, 1984) para implementar as funções de utilidade, de forma que modelos analíticos sejam usados para estimar os valores de QoS para essas métricas, de acordo com propriedades do serviço e do recurso. Por outro lado, para cada métrica de QoS pode haver diferentes funções de utilidade que a implementam.

Dada a definição de métrica de QoS e de função de utilidade, uma restrição de QoS pode ser formalmente definida como:

DEFINIÇÃO 3 (RESTRIÇÃO DE QOS)

Uma restrição de QoS é uma tupla $(m, U, \Omega, \boxdot, \tau)$*:*

- m *– é uma métrica de QoS.*
- U *– é a função de utilidade usada para obter os valores da métrica de QoS. A notação* U_{ijk} *é adotada para referir ao*

valor dado pela função de utilidade usada na restrição de QoS k *quando o serviço* i *é implantado no recurso* j.

- Ω *– é um conjunto de serviços (*$\Omega \in S$*), sobre os quais a restrição deve ser aplicada.*
- $\boxdot$ *– é um operador relacional tal que:*
  - $\boxdot \in \{<, \leq, =, 6=, >, \geq\}$ *se houver relação de ordem total em* D_m*, ou*
  - $\boxdot \in \{=, \neq\}$ *se não houver relação de ordem total em* D_m*; e*
- τ *– é o valor-alvo para a métrica (*$\tau \in D_m$*).*

É assumido que restrições de QoS são especificadas usando métricas para as quais $\oplus$ é um operador comutativo e associativo, como $\oplus = +$ para latência. Em virtude disso, a sequência de execução dos serviços não é relevante para a implantação, sendo omitida mesmo quando Ω não é um conjunto unitário.

Utilizando a função de utilidade definida para a métrica é possível verificar a satisfação da restrição de QoS. Essa informação é definida como x_k^q, uma variável inteira que indica se a restrição de QoS k será satisfeita:

$$x_k^q = \begin{cases} 1, & \text{se } U_{11k} \oplus \ldots \oplus U_{1tk} \oplus U_{21k} \oplus \ldots \oplus U_{2tk} \oplus \ldots \oplus U_{ntk} \leq \tau; \\ 0, & \text{caso contrário.} \end{cases}$$

Na Definição 1 é assumido que $U_{ijk}, 1 \leq i \leq n, 1 \leq j \leq t$, é avaliado como o elemento neutro da métrica quando o par referente ao serviço s_i e ao recurso v_j não for considerado na composição de valores para a restrição k. Como pode ser visto, o valor 1 indica que a restrição será satisfeita, ou seja, a agregação dos valores obtidos com a função de utilidade ao considerar todos os pares existentes

de serviço e recurso deve estar dentro do limite imposto pelo valor-alvo da restrição. De maneira complementar, o valor 0 denota a não satisfação da restrição.

Quanto a restrições em relação a atributos do recurso, uma restrição eliminatória é definida como:

DEFINIÇÃO 4 (RESTRIÇÃO ELIMINATÓRIA)

É representada por uma tupla $(\Omega, \phi, \boxdot, \tau)$*, onde:*

- Ω *– é um conjunto de serviços (* $\Omega \subseteq S$ *) para os quais a restrição deve ser aplicada.*
- ϕ *– é um atributo que caracteriza recursos (*$\phi \subset \iota[\ ]$*). A notação* ϕ_v *é usada para referir ao valor deste atributo para o recurso* v*.*
- $\boxdot$ *– é um operador relacional tal que:*
  - $\boxdot \in \{<, \leq, =, 6=, >, \geq\}$ *se houver relação de ordem total no domínio de* ϕ*, ou*
  - $\boxdot \in \{=, \neq\}$ *se não houver relação de ordem total no domínio de* ϕ*.*
- τ *– é o valor-alvo para esta restrição.*

De forma análoga a restrições de QoS, é definido x_k^e, uma variável inteira que indica se a restrição eliminatória k será satisfeita:

$$x_k^e = \begin{cases} 1, & \text{se } \phi_v \boxdot \tau \text{ É verdadeiro para um ou mais } v \in V; \\ 0, & \text{caso contrário.} \end{cases}$$

Uma restrição classificatória é definida como:

DEFINIÇÃO 5 (RESTRIÇÃO CLASSIFICATÓRIA)

É representada por uma tupla $(\Omega, \Phi, \boxminus)$*, onde:*

- Ω *– é um conjunto de serviços (* $\Omega \subseteq S$ *) para os quais a restrição deve ser aplicada.*
- Φ *– é um atributo que caracteriza recursos (*$\Phi \subset t[\ ]$*) .*
- $\boxminus$ *– é um operador de classificação:* $\boxminus \in$ *{ASCENDANT, DESCENDANT}, onde ASCENDANT indica que os recursos devem ser ordenados em ordem ascendente de acordo com o valor do atributo; e DESCENDANT indica que os recursos devem ser ordenados em ordem descendente de acordo com o valor do atributo.*

Uma vez que restrições classificatórias são usadas somente para ordenar recursos candidatos, restrições desse tipo sempre serão satisfeitas. Mesmo assim, para uso posterior na formalização, é definido x_k^r , uma variável inteira que indica se uma restrição classificatória k será satisfeita:

$$x_k^r = \{1, \quad \text{sempre.}$$

Os elementos fundamentais formalizados nessa seção compõem a base da notação proposta para a representação de coreografias de serviços e do arcabouço proposto para a especificação de restrições não funcionais. Nas próximas seções, serão apresentadas estas propostas.

Notação para representação de coreografias de serviços

Conforme destacado anteriormente, há na literatura e na indústria um número considerável de linguagens que foram propostas para permitir a modelagem de coreografias de serviços, das quais a maioria constitui variações de BPMN (Business [...], 2011).

Isso ocorre porque BPMN se tornou o padrão de fato para notação gráfica de modelagem de processos de negócios, com ênfase no fluxo de controle em um nível independente de implementação (Decker; Kopp; Barros, 2008).

Um dos principais objetivos de BPMN é fornecer uma notação que seja facilmente compreensível por todos os usuários envolvidos na definição do processo de negócios, sendo uma boa candidata para fornecer uma notação gráfica para modelagem de coreografias de serviços. No entanto, este nem sempre é o caso. As especificações BPMN podem ser muito complexas e o padrão introduz limitações que um projetista de coreografias deve obedecer. Contudo, o padrão fornece somente uma descrição textual para essas limitações, tornando difícil seu correto entendimento (Autili *et al.*, 2014). Nas especificações de coreografias usando BPMN, as dependências entre os serviços são "ocultas", o que torna complexo seu uso na tomada de decisões sobre gerenciamento de recursos. Além disso, BPMN (e suas variações) não são capazes de modelar explicitamente a estrutura de uma coreografia, assim como as restrições não funcionais impostas sobre os serviços. Por fim, o uso de BPMN como única linguagem de modelagem restringiria a solução proposta a essa linguagem, tornando imperativa sua utilização.

Em virtude dos motivos descritos e das limitações encontradas em outras propostas existentes, uma nova notação foi indicada para modelagem de coreografias de serviços. Esta notação tem como finalidade a representação desses componentes de forma não ambígua e permitir que a demanda sobre cada serviço, assim como as dependências entre eles, seja mais facilmente identificada e processada. Outra motivação em adotá-la é que ela, com o arcabouço que será apresentado na próxima seção, permite o vínculo dos serviços às restrições não funcionais especificadas sobre as coreografias.

O objetivo ao propor essa notação não é definir uma linguagem de propósito geral, mas é considerada apenas como uma representação interna na proposta. Sua adoção não elimina o uso de BPMN ou outra linguagem de modelagem, pois ela constitui apenas uma representação abstrata da coreografia. Dessa forma, as coreografias são inicialmente especificadas usando BPMN (ou outra linguagem com esse propósito) e, então, automaticamente traduzidas para a representação interna com a inclusão das restrições associadas. Essa representação interna é utilizada para guiar as decisões sobre gerenciamento de recursos, ao passo que a encenação da coreografia continua a ser guiada pela linguagem originalmente utilizada.

Neste capítulo, é definida formalmente a notação proposta e é mostrado como transformar modelos de coreografia descritos em BPMN para modelos que usam essa representação. Apesar de BPMN ser considerada como linguagem de modelagem, qualquer outra poderia ser usada, uma vez que um adaptador esteja disponível.

A topologia de uma coreografia de serviços é representada usando um grafo de processo (Mendling; Lassen; Zdun, 2006), que é definido a seguir:

DEFINIÇÃO 6 (NÓS PREDECESSORES E SUCESSORES)

Seja N *o conjunto de nós e* $E \subseteq N \times N$ *uma relação binária sobre* N *que define as arestas. Para cada nó* $n \in N$*, definimos o conjunto de nós predecessores* $\blacksquare n = \{x \in N | (x, n) \in E\}$ *e o conjunto de nós sucessores* $n \blacksquare = \{x \in N | (n, x) \in E\}$.

DEFINIÇÃO 7 (GRAFO DE PROCESSO)

Um grafo de processo PG consiste em uma tupla (b, Z, S, L, E)*, em que:*

- b *– denota o nó inicial, de forma que* $|b \blacksquare| = 1$ *e* $|\blacksquare b| = 0$*, ou seja, o nó inicial possui apenas um nó sucessor e nenhum nó predecessor.*

- Z – *denota o conjunto de nós finais, de forma que* $|Z| \geq 1$ *e* $\forall z \in Z : |\blacksquare z| \geq 1$ *e* $|z \blacksquare| = 0$, *ou seja, cada nó final tem um ou mais nós predecessores e nenhum nó sucessor. Múltiplos nós finais são modelados para representar diferentes estados de finalização (sucesso, falha etc.).*
- S – *denota o conjunto de serviços, de forma que* $\forall s \in S : |\blacksquare s| \geq 1$ *e* $|s \blacksquare| \geq 1$, *ou seja, cada serviço tem um ou mais nós predecessores e um ou mais nós sucessores.*
- L – *denota o conjunto de conectores, de forma que* $|L| \equiv 0$ *(mod 2), e pode ser particionado em conjuntos disjuntos* L^s = *{AND*^{s} *(split de conjunção), OR*^{s} *(split de disjunção), XOR*^{s} *(split de disjunção mutuamente exclusiva)}* L^j = *e {AND*^{j} *(join de conjunção), OR*^{j} *(join de disjunção), XOR*^{j} *(join de disjunção mutuamente exclusiva)}, de forma que* $\forall l^s \in L^s : \left(|\blacksquare l^s| = 1 e\ |l^s \blacksquare| > 1\right), \forall l^j \in L^j : \left(|\blacksquare l^j| > 1 e\ |l^j \blacksquare| = 1\right), \forall l^s \in L^s \rightarrow \exists l^j \in L^j$, *ou seja, o tamanho do conjunto* L *é um número par, conectores split são usados para separar o fluxo do processo enquanto conectores join são usados para agregar partes do fluxo do processo, e para cada conector split há um conector join equivalente. Para cada* l^s = $\{XOR^s, OR^s\}$ $\in L^s$, $l^s\blacksquare = \{G_1,...,G_n\}$, *ou seja, uma requisição pode ser encaminhada para um subgrafo* $G_1, \ldots, G_n$. *É estabelecido que há uma probabilidade* p *para cada uma das possibilidades. Para um OR*^{s}, *em adição à probabilidade de cada subgrafo, há uma probabilidade* p *da requisição ser encaminhada simultaneamente para todos os subgrafos em um dado subconjunto da combinação de* $G_1, \ldots, G_n$ *considerando* k *elementos, onde* $2 \leq k \leq n$. *Como exemplo, com dois subgrafos como sucessores para um conector OR*^{s}, *uma requisição pode ser encaminhada para os subgrafos* G_1, G_2 *ou simultaneamente para ambos, com probabilidade* p_1, p_2 ep_{12}, *respectivamente.*

- E – *é um conjunto de arestas que definem o fluxo como um grafo direcionado. Cada aresta* $e \in E$ *é uma tupla* $(\underrightarrow{e}, \vec{e}, o)$, *onde* $\underrightarrow{e} \subseteq (b \cup S \cup L)$ *é a origem da aresta,* $\vec{e} \subseteq (Z \cup S \cup L)$ *é o destino da aresta, e* o *é a operação sendo requisitada no serviço modelado como destino. Se* $\vec{e} \in \{Z \cup L\}$, *então* o *é nula.*

Como forma de facilitar a apresentação, é adicionada uma representação gráfica desses elementos, conforme apresentado no Quadro 7.

QUADRO 7
Representação gráfica da notação do grafo de processo

Elemento	Notação *PG*	Elemento	Notação *PG*
Evento inicial	b	Evento final	z
Serviço	S_i		
Aresta quando $\vec{e} \subseteq S$	Operação →	Aresta quando $\vec{e} \subseteq (Z \cup L)$	→
Conector *fork* de conjunção	AND^s	Conector *join* de conjunção	AND^j
Conector *fork* de disjunção	OR^s	Conector *join* de disjunção	OR^j
Conector *fork* de disjunção mutuamente exclusiva	XOR^s	Conector *join* de disjunção mutuamente exclusiva	XOR^j

Fonte: Elaboração própria.

Nessa definição, o conjunto de possíveis padrões de composição é restringido a um subconjunto dos conectores usados com mais frequência na modelagem de processos de negócios (sequência, conjunção, disjunção e disjunção mutuamente exclusivas). Essa limitação foi assumida porque construções mais complexas podem,

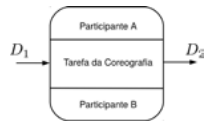
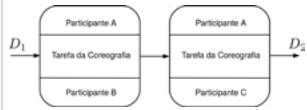
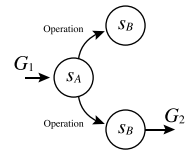
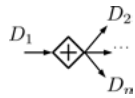
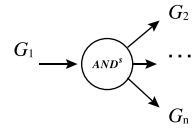
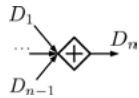
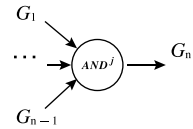
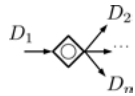
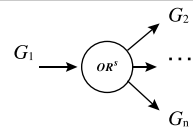
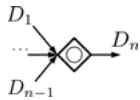
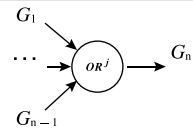
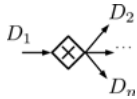
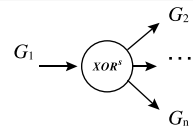
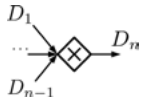
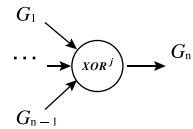
a princípio, ser obtidas a partir da combinação das construções desse subconjunto. Contudo, a comprovação da representatividade da notação proposta é tida como trabalho futuro. Para tal, é proposto uma análise baseada nos padrões de interação entre serviços (*Service Interaction Patterns*) (Barros; Dumas; Hofstede, 2005), um conjunto de padrões de projeto que são considerados como um *benchmark* para avaliação de linguagens de modelagem de coreografias de serviços. Eles capturam os cenários típicos de interação entre dois ou mais participantes.

Um grafo de processo representa uma única coreografia de serviços, destacando as tarefas executadas por serviço para alcançar o objetivo da coreografia. A principal meta no uso dessa notação é facilitar a identificação de dependências entre os serviços e a análise da carga neles agregada.

Ao traduzir modelos de coreografias descritas em BPMN para a notação de grafo de processo, a estratégia de mapeamento entre modelos foi limitada a um subconjunto de elementos de BPMN, considerando aqueles elementos que são mais comumente utilizados na representação de coreografias. Além disso, a associação de conectores com valores de probabilidade foi inclusa no modelo, uma vez que elas são quantificações do comportamento do processo de negócio.

O mapeamento dos elementos BPMN em componentes do grafo de processo é compactamente apresentado no Quadro 8, em que $D_1,\ldots,D_n$ são subdiagramas, $G_1,\ldots,G_n$ são subgrafos, $p_i, i \in \{1,\ldots,n\}$ é a probabilidade de uma requisição ser encaminhada ao subgrafo G_i e p_{12} é a probabilidade da requisição ser encaminhada aos subgrafos G_1 e G_2, simultaneamente. Usando as regras de mapeamento propostas é possível implementar um adaptador que gera modelos na notação PG a partir de modelos em BPMN.

QUADRO 8
Tradução dos principais elementos BPMN para a notação do grafo de processo

Elemento	Descrição	Notação BPMN	Notação PG
Evento inicial	Evento sem conotação especial que indica o ponto de partida		
Evento final	Evento sem conotação especial que indica o ponto de encerramento		
Tarefa	Representa a troca de mensagens entre dois participantes		
Sequência de tarefas	Representa a sequência da troca de mensagens		
Conector *fork* de conjunção	Divide o fluxo, sendo todos os fluxos de saída acionados simultaneamente		
Conector *join* de conjunção	Combina fluxos paralelos aguardando que todos os fluxos de entrada se completem antes de acionar o fluxo de saída		
Conector *fork* de disjunção	Divide o fluxo, e ativa um ou mais fluxos de saída		
Conector *join* de disjunção	Aguarda que todos os fluxos de entrada ativos se completem antes de acionar o fluxo de saída		
Conector *fork* de disjunção mutuamente exclusiva	Divide o fluxo encaminhando-o para exatamente um fluxo de saída		
Conector *join* de disjunção mutuamente exclusiva	Aguarda que o fluxo de entrada ativo se complete antes de acionar o fluxo de saída		

Fonte: Elaboração própria.

Conforme descrito, alguns elementos da BPMN não são representados na notação PG. Contudo, essa representação interna tem como objetivo fornecer uma visão estrutural da coreografia, evidenciando as dependências entre os serviços e a carga agregada em cada um deles.

O uso do grafo de processo é restrito às atividades relacionadas à implantação da coreografia, propósito para o qual os elementos representados são suficientes. A encenação da coreografia ou outras atividades para as quais a notação do grafo de processo não é suficiente, podem ser realizadas usando o modelo original em BPMN (ou outra linguagem).

As figuras 14 e 15 apresentam os grafos de processo gerados a partir dos modelos em BPMN das coreografias do exemplo de cenário. A notação em BPMN é novamente apresentada para facilitar a correspondência entre os dois modelos.

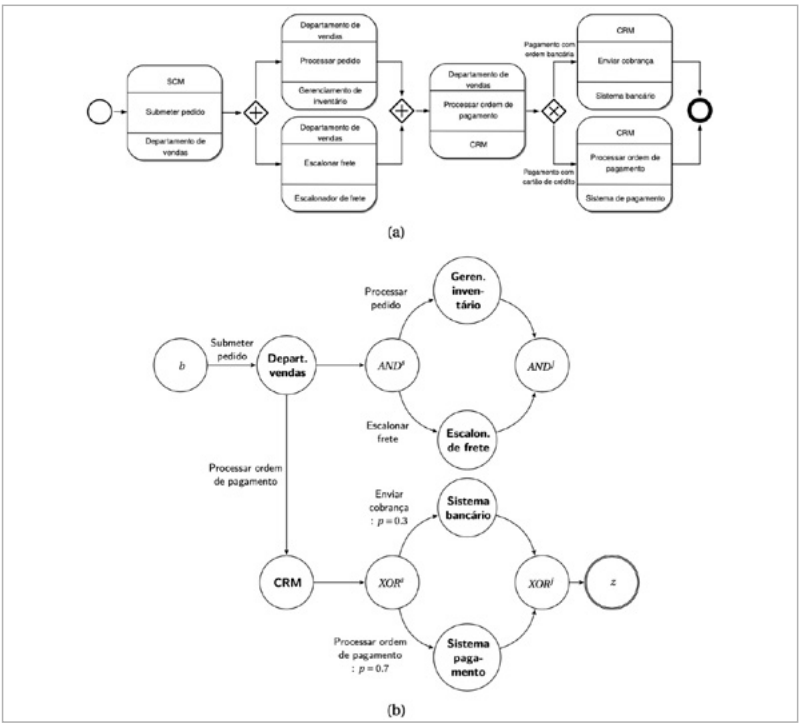


Figura 14 – Coreografia para cumprimento de pedido
(a) Usando a notação BPMN e (b) usando a notação PG.
Fonte: Elaboração própria.

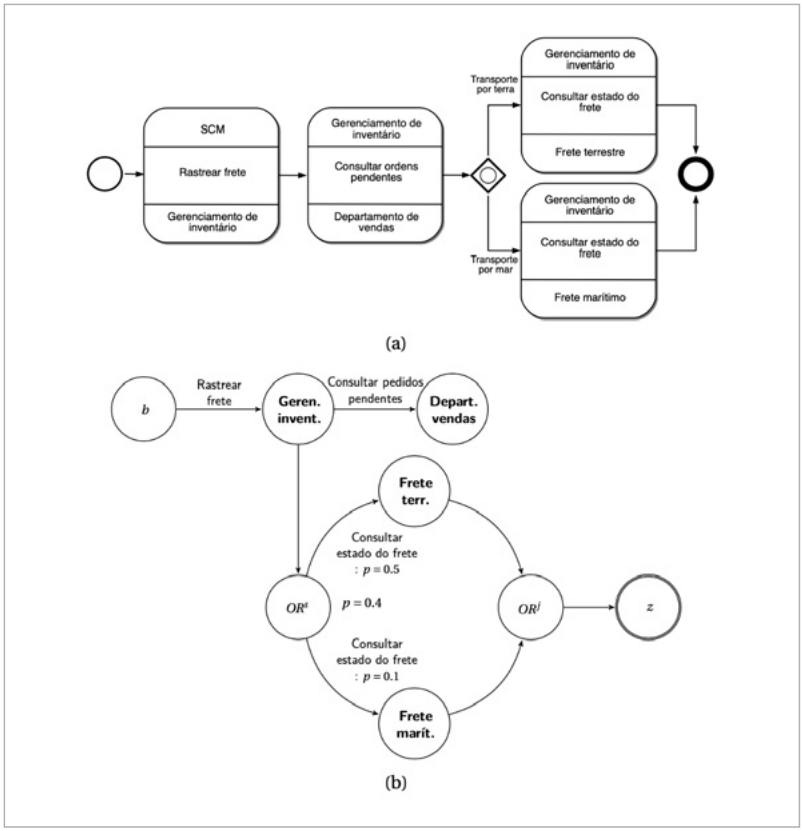


Figura 15 – Coreografia para rastreamento de frete
(a) Usando a notação BPMN.
(b) Usando a notação PG.
Fonte: Elaboração própria.

Em um modelo de coreografia usando a notação do grafo de processo é possível obter mais facilmente a carga agregada sobre as operações requisitadas em cada serviço, assim como os padrões de interação entre eles. No entanto, é necessário estabelecer alguma forma de representação das restrições sobre as operações. Para isso, foi proposto um arcabouço para representação dos principais atributos das definições de restrições. Além disso, o arcabouço facilita a inclusão de novos tipos de restrições na modelagem de coreografias.

Arcabouço para especificação de restrições não funcionais sobre serviços

No arcabouço proposto para representação de restrições não funcionais, é pressuposto que as informações sobre serviços estão disponíveis em um repositório global compartilhado, sendo acessadas por meio de um identificador do serviço.

Ao propor um modelo para a representação de restrições de QoS, é preciso generalizar a representação de todos os seus elementos. Um dos maiores desafios para tal é abstrair a forma pela qual os valores da função de utilidade são obtidos. Visando focar no problema do gerenciamento de recursos, o tratamento da estimativa de QoS na abordagem foi tratado como um problema isolado. Dessa forma, é assumido que para cada métrica de QoS há um *estimador de QoS* que fornece a estimativa para quando um serviço for implantado em um determinado recurso.

Os valores de QoS podem ser aferidos usando atributos que variam para cada métrica de QoS. Por exemplo, as estimativas da latência e da vazão são baseadas na carga das operações requisitadas, assim como na demanda de recurso para processar cada requisição. Em virtude disso, foi proposto que a implementação do estimador da métrica utilize um outro componente definido como *construtor de atributos*, o qual é responsável por obter os dados necessários para a estimativa da QoS. Esses dados podem ser obtidos usando atributos do serviço e do recurso sobre os quais a estimativa está sendo realizada, assim como atributos da carga sobre o serviço e do seu relacionamento com os demais serviços representados no grafo de dependências.

A relação dos componentes propostos para a generalização de estimativas de QoS é ilustrada na Figura 16. Consequentemente, para incluir uma restrição de QoS na modelagem de uma coreografia o usuário deve especificar uma implementação a ser usada para cada um desses componentes.

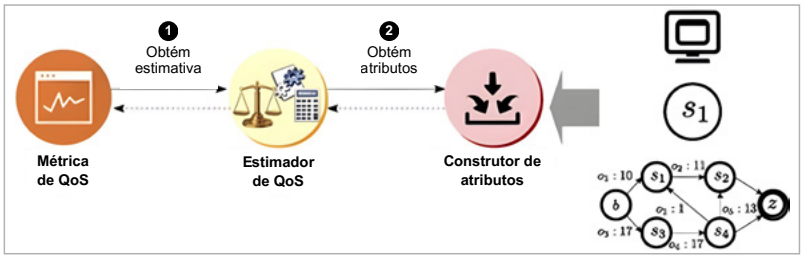


Figura 16 – Componentes para a generalização de estimativas de QoS
Fonte: Elaboração própria.

A representação de restrições eliminatórias e classificatórias pode ser implementada mais facilmente, porque elas são descritas usando um conjunto predefinido de elementos e avaliadas usando operadores predeterminados. Em virtude disso, para representar os principais elementos na definição de restrições em relação a atributos do recurso, foi proposta uma linguagem cuja gramática[3] é descrita na Figura 17 na forma de Backus-Naur:

```
⟨expression⟩      ::= ⟨term⟩ | ⟨term⟩ ⟨or⟩ ⟨term⟩
⟨term⟩            ::= ⟨factor⟩ | ⟨factor⟩ ⟨and⟩ ⟨factor⟩
⟨factor⟩          ::= ⟨resource att⟩ ⟨relational op⟩ ⟨comparable⟩
                    | ⟨rank order⟩ ⟨resource att⟩
                    | '(' ⟨expression⟩ ')'
⟨comparable⟩      ::= ⟨number⟩ | ⟨string⟩
⟨number⟩          ::= [⟨sign⟩] {⟨digit⟩}* ['.'] {⟨digit⟩}+
⟨sign⟩            ::= '+' | '-'
⟨digit⟩           ::= '0-9'
⟨string⟩          ::= {'A-Z'}+ ⟨string⟩ | {'a-z'}+ ⟨string⟩ | ϵ
⟨or⟩              ::= '|'
⟨and⟩             ::= '&'
⟨resource att⟩    ::= 'LOCATION' | 'COST' | 'CLOUD_PROVIDER' | ...
⟨rank order⟩      ::= 'ASCENDANT' | 'DESCENDANT'
⟨relational op⟩   ::= '<' | '<=' | '=' | '==' | '!=' | '<>' | '>' | '>='
```

Figura 17 – Gramática da linguagem usada na especificação de restrições em relação a atributos do recurso
Fonte: Elaboração própria.

3 Nesta gramática, o conjunto de atributos que descrevem o recurso e que, consequentemente, podem ser usados para descrever restrições não funcionais não é apresentado completamente.

De acordo com a Definição 4, que descreve restrições eliminatórias, a linguagem proposta é usada para especificar Φ (o atributo do recurso sobre o qual a restrição deve ser aplicada), $\boxdot$ (um operador relacional), e τ (o valor-alvo da restrição, que pode ser uma sequência de caracteres ou um número). Por simplicidade, não foi incluído na linguagem a definição da unidade de medida. Por exemplo, para especificar que um certo serviço deve ser implantado em recursos localizados no Brasil, a restrição deve ser especificada como:

```
(LOCATION = Brazil)
```

Adicionalmente, de acordo com a Definição 5 de restrição classificatória, a linguagem proposta é usada para especificar Φ (o atributo do recurso sobre o qual a restrição deve ser aplicada) e $\boxminus$ (operador de classificação, {ASCENDANT, DESCENDANT}). Por exemplo, para selecionar recursos em ordem ascendente de custo financeiro, isto é, dando preferência a recursos menos onerosos, a restrição deve ser representada com:

```
(ASCENDANT COST)
```

A linguagem proposta descreve restrições relativas a atributos do recurso usando operações lógicas na forma normal disjuntiva. O objetivo dessa representação é permitir as seguintes características:

- O usuário pode especificar múltiplas restrições a serem aplicadas a um mesmo objeto. Exemplo: a restrição a seguir especifica que a implantação deve ocorrer usando recursos localizados no Brasil, e cujo custo financeiro deve ser até US$3.5.

```
(LOCATION = Brazil) & (COST <= 3.5)
```

- O usuário pode especificar condições alternativas a serem aplicadas a um mesmo objeto. Exemplo: a restrição a seguir especifica que a implantação deve ocorrer usando recursos localizados no Brasil ou usando recursos nos Estados Unidos, mas que tenham custo financeiro de até US$1.0.

```
(LOCATION = Brazil) | ((LOCATION = USA) & (COST <= 1.0))
```

Representação de múltiplas coreografias e restrições

A satisfação de restrições em coreografias de serviços é ainda mais difícil se considerarmos o compartilhamento de serviços entre elas e os diferentes papéis que eles implementam em cada uma das coreografias. Por exemplo, um serviço de mapas pode ser usado em aplicações como guias de rotas de condução e marcação de localização em imagens. Para cada uma delas, o serviço pode ter restrições não funcionais diferentes. Este cenário é equivalente a um dançarino participante em uma coreografia de um *mashup*[4] (Gomes *et al.*, 2016): o profissional deve ser capaz de realizar a coreografia corretamente e com qualidade, lidando com prováveis diferentes ritmos de dança. No mesmo sentido, para uma dada aplicação (o que é análogo ao *mashup* em nossa metáfora) pode haver várias coreografias (ritmos em nossa metáfora), com serviços compartilhados (análogos aos dançarinos) e sujeitos a diferentes restrições. Portanto, não é possível fazer um gerenciamento eficiente dos recursos sem considerar, para cada serviço, todas as coreografias em que ele participa.

4 Um *mashup* é uma canção ou composição criada a partir da mistura de duas ou mais canções pré-existentes, normalmente pela fusão do vocal de uma canção ao instrumental de outra, de forma que se combinam (Mashup [...], 2022).

Na solução proposta, cada coreografia de serviços é analisada usando um modelo de Rede de Filas (*Queuing Network model*) no estado estável (Lazowska *et al.*, 1984). Além disso, é suposto que o gerenciamento de recursos é realizado considerando uma determinada fatia de tempo. Fazendo isso, é possível confiar em alguns resultados bem estabelecidos e assumir que o teorema de Burke (1956) é válido nesse cenário. De acordo com este teorema, as chegadas de requisições seguem um processo de *Poisson* com taxa λ ; e a taxa com que essas requisições são atendidas também é um processo de *Poisson* com taxa λ .

Considerando as suposições apresentadas, é proposto a criação de um modelo para a representação de múltiplas coreografias de serviços. Esse modelo é gerado a partir da representação das coreografias usando a notação do grafo de processo. O objetivo em propô-lo é desenvolver uma visão conjunta de todas as coreografias, sem considerar a lógica de coreografias específicas, mas somente a dependência entre os serviços e os fatores que afetam cada operação executada.

O primeiro passo no desenvolvimento do modelo conjunto de coreografias é remover os conectores entre os serviços em cada coreografia, pois eles representam a lógica de cada uma. Esse passo é nomeado como *redução do grafo de processo*. Para remover um conector, é preciso avaliar como ele influencia na avaliação de restrições não funcionais. Como restrições relativas a atributos do recurso e de QoS sobre serviços específicos são avaliadas considerando cada serviço isoladamente, os conectores só têm influência na avaliação de restrições de QoS fim-a-fim. Nesses casos, a avaliação de cada conector é realizada usando o operador de agregação e a função de utilidade das métricas de QoS em questão, de acordo com as regras apresentadas no Quadro 9, onde $G_1, G_2, \ldots, G_g$ são subgrafos do grafo de processo que representa a coreografia.

QUADRO 9
Regras de composição de valores de QoS fim-a-fim

Padrão	Regra de composição
Sequência	$U(G_1) \oplus U(G_2)$
XOR^s / XOR^j	$\oplus_{i=1}^{g} p_i U(G_i)$
AND^s / AND^j	$\vee\{U(G_1), U(G_2), \ldots, U(G_g)\}$
OR^s /OR^j	$\oplus_{i=1}^{g} p_i U(G_i) \oplus \oplus_{k=2}^{n} p_c \vee\{U(G_1), U(G_2), \ldots, U(G_k)\}$, onde $c = \binom{g}{k}$

Fonte: Elaboração própria.

O operador $\oplus$ define como a QoS é incrementada por cada novo evento/operação (para uma coreografia). O valor composto da função de utilidade para subgrafos conectados sequencialmente é obtido agregando o valor de utilidade de cada subgrafo usando $\oplus$. Para o padrão de disjunção mutuamente exclusivo, é utilizada uma estratégia semelhante, mas aplicando a probabilidade de execução de cada subgrafo no seu valor de utilidade. Para o padrão de conjunção, o paralelismo deve ser levado em conta. Nesse caso, somente a pior QoS é tomada entre os ramos executados em paralelo usando o operador $\vee$ definido na métrica. Para o padrão de disjunção, o valor composto da função de utilidade é calculado agregando os valores da função de utilidade, considerando cada subgrafo multiplicado pela probabilidade de executar apenas este subgrafo. Além do valor de utilidade na execução de cada subgrafo isolado, a execução paralela de subgrafos também deve ser incluída nesse caso. Isso é realizado restringindo o pior valor de QoS com a probabilidade de executar os subgrafos em paralelo. Esse valor é calculado para cada combinação de todos os subgrafos tomados k a k ($k = \{2, \ldots, g\}$, em que g é o número de subgrafos).

Com base nas regras de composição apresentadas, ao fazer a redução de um grafo de processo, considerando o padrão de sequência, é suficiente especificar a carga esperada para cada operação. O mesmo é verdadeiro para o padrão de disjunção mutuamente exclusiva se for aplicado sobre a carga a probabilidade. Contudo, para os padrões de conjunção e disjunção é preciso preservar a informação sobre a execução paralela, já que apenas indicar a carga não é suficiente para compor a QoS final.

Como exemplo, considere o fragmento da coreografia de rastreamento de frete, apresentado na Figura 18(a). Supondo que a carga seja constante de 100 requisições por unidade de tempo, ao reduzir esse subgrafo para eliminar os conectores OR^s / OR^j, levando em conta apenas a aplicação da probabilidade sobre esta carga, o resultado será o grafo apresentado na Figura 18(b), uma vez que a carga sobre o serviço "Frete terrestre" será

$$p_{\text{Frete terrestre}} \times \lambda + p_{\text{ambos}} \times \lambda = 0,5 \times 100 + 0,4 \times 100 = 90$$

e sobre o serviço "Frete marítimo" será

$$p_{\text{Frete marítimo}} \times \lambda + p_{\text{ambos}} \times \lambda = 0,1 \times 100 + 0,4 \times 100 = 50$$

Contudo, a informação sobre a quantidade real de carga executada em paralelo não é apresentada no subgrafo gerado, levando à interpretação errônea de que 50 requisições são executadas em paralelo. Isso faria com que a QoS composta fosse computada como[5]:

$$0,5 \vee \left\{U_{\text{Frete terrestre}}, U_{\text{Frete marítimo}}\right\} \oplus 0,4 U_{\text{Frete terrestre}}$$

em vez de

$$0,4 \vee \left\{U_{\text{Frete terrestre}}, U_{\text{Frete marítimo}}\right\} \oplus 0,5 U_{\text{Frete terrestre}} \oplus 0,1 U_{\text{Frete marítimo}}$$

como deveria.

5 Neste exemplo, para facilitar a leitura, os índices j e k foram omitidos na definição da função de utilidade U_{ijk}.

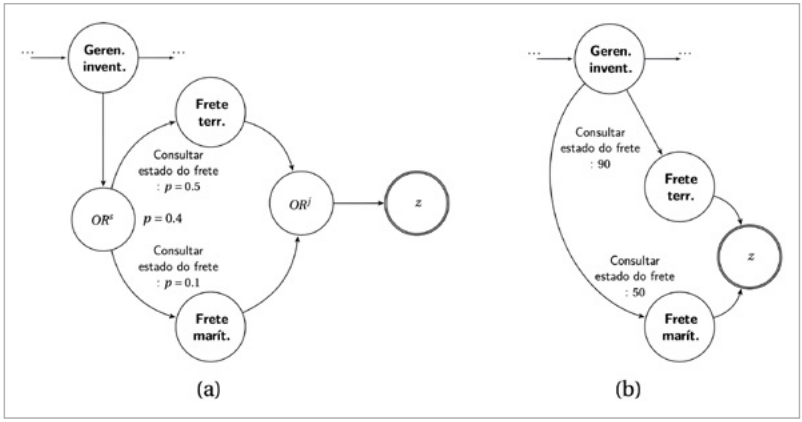


Figura 18 – Exemplo para ilustrar a necessidade de representação de paralelismo
(a) Fragmento da coreografia de rastreamento de frete.
(b) Redução da coreografia por meio da eliminação dos conectores.
Fonte: Elaboração própria.

Para contornar o problema apresentado, é definida outra estrutura para auxiliar na representação das requisições que devem ser avaliadas em paralelo. Essa estrutura é denominada como árvore de paralelismo.

DEFINIÇÃO 8 (ÁRVORE DE PARALELISMO)

Uma árvore de paralelio PT é um grafo acíclico conectado representado por uma tupla (b, L, S', E)*, em que:*

- b *– denota o nó raiz,* $|b\blacksquare| \geq 2$ e $|\blacksquare b| = 0$*, ou seja, o nó raiz tem dois ou mais nós sucessores e nenhum nó predecessor.*
- L *– denota o conjunto de descritores de carga,* $b \in L, |L| \geq 1$ *e* $\forall l \in L : l = c$*, onde* c *é uma constante e* $\forall l \in L : l = b \rightarrow |\blacksquare l| = 0$ e $l \neq b \rightarrow |\blacksquare l| = 1$ e $|l\blacksquare| \geq 2$*, ou seja, descritores de carga têm dois ou mais nós sucessores; não têm nó predecessor se este for o nó raiz e têm necessariamente um nó predecessor nos outros casos.*
- S' *– denota o conjunto de serviços, em que* $\forall s \in S' : s \in S$ e $|\blacksquare s| = 1$ e $0 \leq |s\blacksquare| \leq 1$,

ou seja, serviços têm um nó predecessor e têm um ou nenhum nó sucessor (neste último caso são os nós folha).

- E *– é um conjunto de arestas. Cada aresta* $e \in E$ *é uma tupla* $(\underset{\rightarrow}{e}, \vec{e}, o)$*, onde* $\underset{\rightarrow}{e} \subseteq (L \cup S')$ *é o nó origem da aresta,* $\vec{e} \subseteq (L \cup S')$ *é o nó destino da aresta, e* o *é a operação sendo requisitada quando* $\vec{e}$ *é um serviço ou nulo quando* $\vec{e}$ *é uma constante e* $\forall e \in E : \underset{\rightarrow}{e} \in L \rightarrow \vec{e} \ S'$ *e* $\vec{e} \ L \rightarrow \underset{\rightarrow}{e} \in S'$*, ou seja, descritores de carga não podem ser conectados com outros descritores de carga. O fato de ser um grafo simples implica que* $\forall v \in L \cup S' : (v, v) \notin E$ *(sem arestas reflexivas) e que* $\forall x, y \in L \cup S' : \left| \{(x, y) | (x, y) \in E\} \right| = 1$ *(sem múltiplas arestas para um mesmo par).*

A Figura 19 ilustra fragmentos de grafos de processo e a árvore de paralelismo equivalente usando uma notação gráfica. Como pode ser visto, cada subgrafo que contém operações executadas em paralelo é representado como uma árvore cujo nó raiz é a carga e os ramos são os subgrafos ligados pelo conector de conjunção.

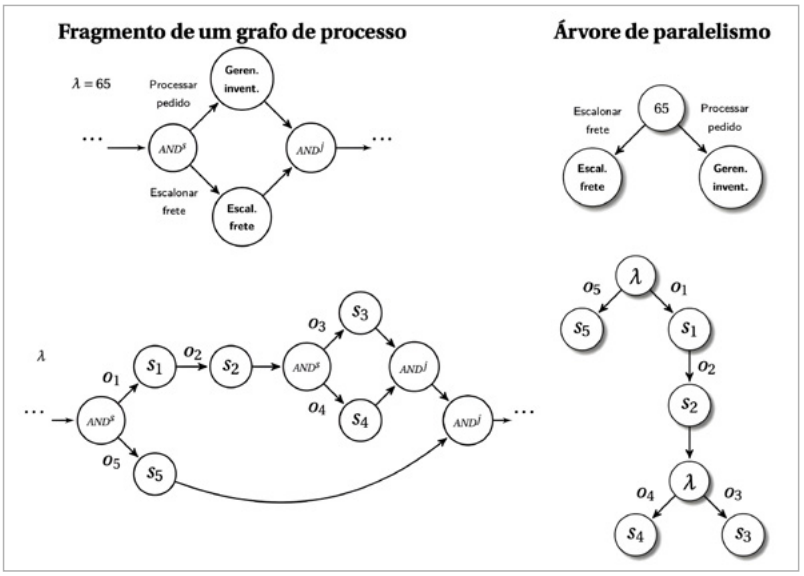
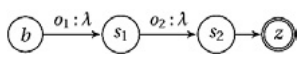
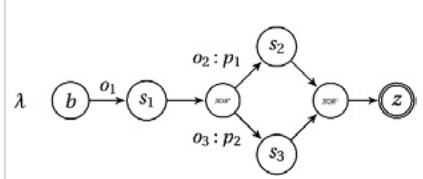
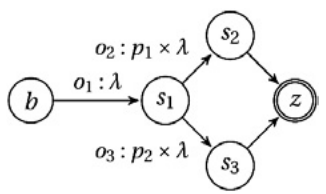
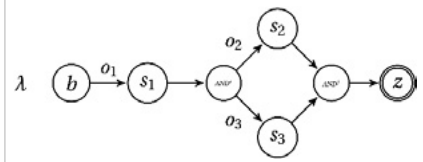
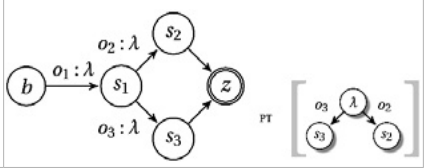
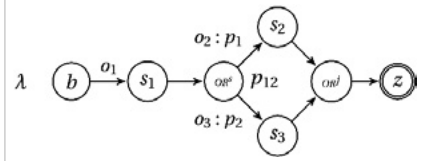
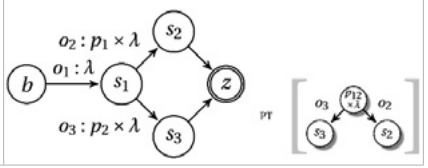


Figura 19 – Exemplos de subgrafos de processo e equivalentes árvores de paralelismo
Fonte: Elaboração própria.

Usando a estrutura de árvore de paralelismo para preservar a informação sobre operações executadas em paralelo, é possível reduzir um grafo de processo de forma a eliminar todos os conectores. O Quadro 10 ilustra como isso é realizado para cada padrão de composição, em que λ é a carga de entrada e PT é a árvore de paralelismo gerada na redução.

QUADRO 10
Redução de grafos de processo de acordo com o padrão de composição

Padrão	Exemplo de grafo de processo	Redução
Sequência		
Disjunção mutuamente exclusiva		
Conjunção		
Disjunção		

Fonte: Elaboração própria.

Embora árvores de paralelismo sejam usadas para especificar quais operações devem ser executadas em paralelo, a simples existência dessa estrutura não é suficiente para representar essa informação de maneira adequada. O principal motivo é que os serviços descritos em uma árvore de paralelismo podem ser usados em outras coreografias (ou inclusive na mesma coreografia em outro ponto da composição) por meio de outros padrões de composição e/ou

sujeitos a outra carga. Dessa forma, representar árvores de paralelismo isoladas torna impossível saber em qual restrição ela deve ser aplicada, uma vez que os conectores são eliminados na redução do grafo de processo. Devido a isso, as árvores de paralelismo devem estar relacionadas com a restrição para a qual o paralelismo deve ser aplicado.

Outro conceito, denominado *Grupo em Restrição*, é especificado para representar a restrição com as informações de paralelismo necessárias para avaliar restrições de QoS fim-a-fim.

DEFINIÇÃO 9 (GRUPO EM RESTRIÇÃO)

Um grupo em restrição K *é representado por um par* (k, Λ)*, em que:*

- k *– é uma restrição.*
- Λ *– é um conjunto de árvores de paralelismo relacionadas à restrição.* $\Lambda = \phi$ *; quando* k *é uma restrição de QoS cujos serviços não devem ser avaliados em paralelo, ou quando* k *é uma restrição em relação a atributos do recurso.*

A descrição dos grupos em restrição é representada internamente incluindo, quando necessário, a informação sobre as árvores de paralelismo na descrição da restrição. Isso é realizado automaticamente, levando em conta a carga esperada e a topologia da coreografia descrita no grafo de processo.

Para agregar todas as coreografias de serviços de entrada em uma estrutura única, é necessário decidir como tratar os grupos em restrição gerados. A junção de dois deles apenas pode ser realizada se forem grupos equivalentes, isto é, possuírem exatamente os mesmos atributos, exceto pelo valor-alvo. Caso essa condição não seja verificada, ambos os grupos em restrição devem ser mantidos na estrutura conjunta, uma vez que eles têm escopos diferentes.

É assumido que o tratamento de conflito está fora do escopo deste trabalho. Quando restrições conflitantes são detectadas, uma mensagem de erro é enviada ao usuário, sendo ele(a) responsável por resolver o conflito optando por uma das restrições conflitantes. Como exemplo, suponha que o usuário submeta duas coreografias.

Suponha também que, devido à localidade dos clientes, o usuário requisite que os serviços da primeira coreografia sejam implantados usando recursos localizados no Brasil, e que os da segunda coreografia sejam implantados a partir de recursos localizados nos Estados Unidos. Caso haja compartilhamento de serviços entre essas duas coreografias, haverá um conflito, tendo o usuário que escolher qual das duas localidades será usada na implantação dos serviços compartilhados.

A junção de grupos envolvendo restrições de QoS deve considerar como o valor-alvo é tratado nas métricas equivalentes. Ao considerar duas restrições de QoS definidas pela mesma métrica, a junção destas restrições pode considerar o valor-alvo mais restritivo (como para latência), ou a composição dos valores-alvo (como para vazão), ou ainda a intersecção de conjuntos quando o valor-alvo é definido usando grupos de valores (como para segurança, quando são definidos diferentes níveis aceitáveis). Dessa forma, uma vez que a abordagem propõe a abstração da representação da métrica de QoS, é preciso abstrair, também, a forma como é feita a junção dos valores-alvo. Para tal, é definido a função a seguir.

DEFINIÇÃO 10 (FUNÇÃO DE JUNÇÃO DE VALORES-ALVO)

É uma função $\odot : D_m \times D_m \rightarrow D_m$, *definida de maneira específica para a métrica* m, *que obtém o valor resultante da junção de dois valores-alvo estabelecidos sobre o domínio* D *desta métrica.*

A função $\odot$ deve ser modelada para cada métrica de QoS e usada no processo de junção de grupos equivalentes. Por exemplo, no caso da métrica se referir à latência, $\odot = \wedge$, significando que, para juntar dois grupos equivalentes restritos pela métrica de latência, é suficiente manter o melhor valor-alvo de latência, dado que o outro objetivo é obtido como consequência. Também, para a métrica vazão, $\odot = +$, significando que, para juntar dois grupos equivalentes restritos pela métrica vazão, é necessário estabelecer como novo valor-alvo a soma dos valores-alvo anteriores.

Para restrições eliminatórias, também ocorre junção de grupos equivalentes quando o atributo representa uma variável numérica.

Nesse caso, a junção também é realizada conservando o valor-alvo mais restrito, que é determinado pelo operador relacional usado.

Nos casos em que o valor-alvo é uma sequência de caracteres ou quando a restrição é classificatória, a junção só ocorre para grupos exatamente iguais (grupos equivalentes com mesmo valor-alvo). Assim como em restrições de QoS, é assumido que não há tratamento de conflito para restrições em relação a atributos do recurso.

Aplicando as técnicas de redução do grafo de processo e junção de restrições, a estrutura para representação de um conjunto de coreografias é gerada. Essa estrutura é chamada de *grafo de dependências sujeito a restrições*, ou simplesmente *grafo de dependências*.

DEFINIÇÃO 11 (GRAFO DE DEPENDÊNCIAS SUJEITO A RESTRIÇÕES) *um grafo de dependências sujeito a restrições DG é um grafo direcionado representado por uma tupla* (b, z, S, E, K)*, em que:*

- b *– denota o nó inicial,* $|b\blacksquare| = 1$ e $|\blacksquare b| = 0$*, ou seja, o nó inicial possui apenas um nó sucessor e nenhum nó predecessor.*
- z *– denota o nó final,* $|\blacksquare z| \geq 1$ e $|z\blacksquare| = 0$*, ou seja, o nó final tem um ou mais nós predecessores e nenhum nó sucessor.*
- S *– denota o conjunto de serviços,* $\forall s \in S : |\blacksquare s| \geq 1$ e $|s\blacksquare| \geq 1$*, ou seja, serviços tem um ou mais nós predecessores e um ou mais nós sucessores.*
- E *– é um conjunto de arestas que definem as dependências como um grafo direcionado. Cada aresta* $e \in E$ *é uma tupla* $(\underset{\sim}{e}, \vec{e}, o, \lambda)$*, onde* $\underset{\sim}{e} \subseteq (b \cup S)$ *é a origem da aresta,* $\vec{e} \subseteq (z \cup S)$ *é o destino da aresta,* o *é a operação sendo requisitada, e* λ *é a carga de requisições para a operação. Têm-se que* $\vec{e} = z \rightarrow \{o \text{ é nulo e } \lambda \text{ é nulo}\}$*,* $\vec{e} \in S \rightarrow \{o \in O_{\vec{e}} \text{ e } \lambda \text{ é a carga estimada de acordo com a redução do grafo de processo}\}$*.*
- K *– é um conjunto de grupos em restrição.*

A Figura 20 ilustra a implementação da geração de um grafo de dependências a partir de um conjunto de grafos de processo. O funcionamento deste algoritmo se baseia em analisar cada coreografia

(ou seja, o grafo de processo que a representa) por vez, adicionando novos vértices e arestas no grafo de dependências de maneira incremental e fazendo a junção das restrições quando possível.

Entrada: Conjunto de grafos de processo $PG[]$.
Saída: Grafo de dependências DG.

```
1  dg.addNode(b)
2  dg.addNode(z)
3  para cada pg_i em PG[] faça
4      K_i ← reduce(pg_i)
5      para cada e ∈ E_{pg_i} faça
6          se e = b_{pg_i} então
7              e ← b
8          fim
9          se ē ⊆ Z_{pg_i} então
10             ē ← z
11         fim
12         se DG não contém vértice e então
13             DG.addNode(e)
14         fim
15         se DG não contém vértice ē então
16             DG.addNode(ē)
17         fim
18         se DG não contém aresta (e, ē, o) então
19             DG.addEdge(e, ē, o, λ)
20         senão
21             Incremente a carga da aresta com λ (se não nulo)
22         fim
23         se conflict(DG.K, K_i) então
24             retorna CONFLICT_ERROR
25         fim
26         mergeConstraintGroups(DG.K, K_i)
27     fim
28 fim
29 retorna dg
```

Figura 20 – Algoritmo para geração de um grafo de dependências a partir de um conjunto de grafos de processo
Fonte: Elaboração própria.

O algoritmo inicia com a eliminação de conectores por meio da redução do grafo de processo (linha 4). A operação reduce é responsável por eliminar conectores no grafo de processo e, se necessário, construir árvores de paralelismo a serem incluídas na descrição das restrições de QoS especificadas para a coreografia modelada. O resultado é um conjunto de grupos em restrição. Cada aresta no grafo de processo reduzido é então avaliada para verificar se ela contém algum vértice ainda não incluído no grafo de dependências resultante, de forma a haver apenas um único nó inicial e um único nó final.

Caso a aresta contenha um desses nós para o grafo de processo, ele é rotulado como o nó equivalente no grafo de dependências (linhas 6 e 9). Em seguida, caso não exista aresta no grafo de dependências equivalente, a aresta que está sendo avaliada é adicionada com a carga propagada para a operação em questão. Caso já exista uma aresta equivalente no grafo de dependências, à carga nela contida é adicionada a carga propagada da aresta sendo avaliada. No grafo de dependências gerado, a carga isolada de cada coreografia não é mantida, uma vez que ela é transposta para as arestas do grafo.

A ação seguinte no algoritmo é avaliar se há conflito entre os grupos em restrição gerados com aqueles já existentes no grafo de dependências (linha 23). Caso algum conflito seja detectado, uma mensagem de erro é enviada ao usuário, para que ele possa solucionar o conflito. A última etapa no algoritmo é realizar a junção das restrições conforme discutido anteriormente (linha 26). A saída da operação mergeConstraintGroups são as restrições combinadas com aquelas que permaneceram isoladas.

A Figura 21 apresenta o grafo de dependências gerado por meio desse algoritmo ao considerar as coreografias do estudo de caso. Nesse caso, é assumido que a carga (λ) é constante de 30 requisições na coreografia para cumprimento de pedido e de 80 requisições na coreografia para rastreamento de frete. Para tornar o exemplo compacto, a representação das restrições não é apresentada.

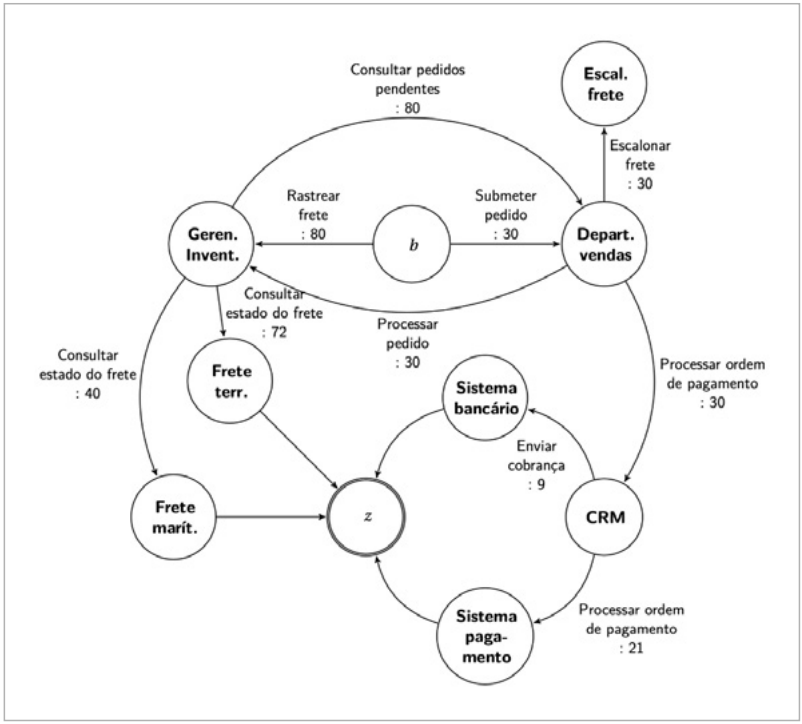


Figura 21 – Grafo de dependências gerado a partir das coreografias (grafos de processo) da Figura 14(b) e da Figura 15(b)
Fonte: Elaboração própria.

As principais dificuldades encontradas na implantação de composições de serviços estão relacionadas à estimativa e à seleção de recursos. Isso se dá porque as restrições não funcionais requeridas têm que ser analisadas face a um conjunto vasto de opções de recursos. Essa tarefa é ainda mais complexa ao considerar múltiplas composições com compartilhamento de serviços, visto que a contribuição dos serviços compartilhados deve ser analisada considerando todas as composições. Além disso, as restrições fim-a-fim têm que ser eficientemente balanceadas entre os serviços a que elas se referem. Essas dificuldades motivaram o tratamento do problema discutido neste livro, cujos principais elementos foram formalizados nesse capítulo.

Inicialmente, foram apresentadas a definição de serviços e de recursos e uma categorização dos tipos de restrições não funcionais que são considerados na especificação de coreografias. A formalização desses elementos teve como objetivo definir de maneira precisa os atributos incluídos no escopo do problema e generalizar a solução que foi desenvolvida, permitindo, assim, o tratamento do problema de forma mais ampla.

Apesar da existência de várias linguagens para modelagem de coreografias de serviço, nenhuma delas consegue expressar restrições não funcionais no nível proposto. Em virtude desse e de outros motivos, como independência e tratabilidade do modelo da coreografia, foi proposta uma abordagem para representação de coreografias de serviços sujeitas a restrições não funcionais.

A notação proposta também foi apresentada de maneira formal neste capítulo, comparando-a com a linguagem BPMN. Em complemento, foi desenvolvido um arcabouço que abstrai a especificação das restrições não funcionais.

Tendo como base a notação proposta, foi apresentada uma estratégia para a representação conjunta de múltiplas coreografias. A estrutura foi descrita com o procedimento para obter tal representação. A geração de um modelo conjunto das coreografias tem como objetivo estabelecer uma visão singular dos serviços a serem implantados e, dessa forma, tornar explícitas as dependências entre eles, facilitando a identificação dos efeitos de seu provável compartilhamento.

A Figura 22 resume as etapas realizadas para gerar a representação conjunta proposta. Em cada etapa são apresentadas as notações utilizadas na representação do modelo das coreografias e das restrições não funcionais, bem como as tarefas desempenhadas para gerar a representação de saída. Como pode ser visto, as coreografias são inicialmente fornecidas em BPMN, ou outra linguagem com este propósito, e transformadas gradativamente até que o modelo conjunto, representado pelo grafo de dependências, seja gerado.

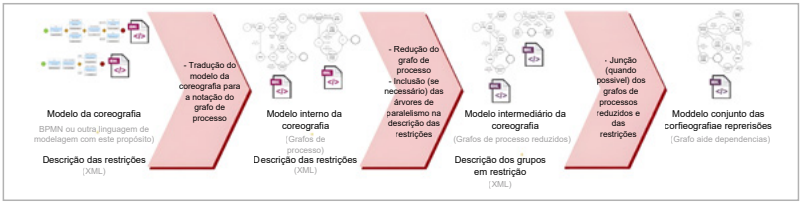


Figura 22 – Etapas na geração da representação combinada de múltiplas coreografias
Fonte: Elaboração própria.

Além da representação conjunta das coreografias e das restrições, o provisionamento de recursos se baseia em um modelo que caracteriza os recursos disponíveis. Dessa forma, a representação conjunta das coreografias e restrições não funcionais, proposta neste capítulo e o modelo de recursos formam a base para o processo de síntese, que consiste em estimar a capacidade necessária e selecionar o recurso ideal para implantar cada serviço. No próximo capítulo, é apresentada nossa proposta para criação do modelo de recursos.

4

Modelagem de recursos

Uma estratégia para auxiliar na seleção de recursos ao implantar uma composição de serviços é decompor restrições fim-a-fim em restrições locais. Nesse caso, as decisões são tomadas de forma isolada para cada serviço, sem considerar os demais. No entanto, essa estratégia restringe excessivamente as decisões sobre a seleção de recursos, uma vez que soluções possíveis com outras decomposições serão desconsideradas, embora elas possam representar escolhas mais eficientes. Para superar esta limitação, é proposto uma abordagem holística de decomposição das restrições fim-a-fim, baseada numa estratégia de emparelhamento de serviços e recursos. Em virtude disso, é necessário que os tipos de recurso disponíveis sejam explicitamente representados.

A representação de recursos proposta nesta abordagem considera a intersecção existente entre os tipos de VM disponíveis em provedores de nuvem pública e o fato de que diferentes tipos podem satisfazer uma dada demanda. Dessa forma, a estratégia é composta de uma modelagem de recursos que utiliza uma representação auxiliar, em que são incluídas somente instâncias representativas dos tipos disponíveis. Neste capítulo é descrita a motivação para adotar essa estratégia e é apresentado o procedimento para gerar a representação proposta. Como forma de ilustrar a técnica apresentada, é gerado o modelo de recursos usando os tipos disponíveis em alguns provedores de nuvem relevantes.

Tipos de recurso

A representação dos tipos de recurso é utilizada como entrada nas atividades de estimativa e seleção de recursos. A sua estimativa depende somente do poder computacional expresso pela capacidade de hardware, ao passo que a seleção pode envolver outros atributos, como custo, localidade, provedor de nuvem que disponibiliza o tipo, entre outros.

A caracterização de tipos de recurso adotada em provedores de nuvem pública especifica cada um a partir de seus atributos de hardware. Com base na especificação do tipo de recurso, a cobrança é realizada de acordo com a região escolhida para instanciá-lo, sendo assim custo e localidade variáveis.

Em virtude dessas características descritas, a modelagem de recursos proposta na abordagem considera dois níveis de representação. No primeiro nível, são descritos todos os atributos que caracterizam um tipo de recurso na modelagem, ao passo que o segundo considera apenas os atributos de hardware. Por simplicidade, estes dois níveis são tratados unicamente como *modelo concreto de recursos*, sendo que instâncias desse nível são chamadas de *tipos concretos de recurso* ou simplesmente *tipos de VM*. A adoção desses dois níveis de representação visa fazer com que a estimativa de recursos seja realizada de maneira mais eficiente, pois, dessa forma, não necessitará levar em consideração múltiplas representações de uma mesma capacidade de hardware, ocasionadas pela repetição de um tipo de recurso nas regiões disponibilizadas por um mesmo provedor.

Uma vez que cada provedor de nuvem usa atributos de hardware diferentes para caracterizar os tipos de recurso disponibilizados, a representação é generalizada usando apenas os atributos mais comumente encontrados: CPU (número de núcleos e velocidade do *clock*), tamanho da RAM, dimensão do disco e desempenho de rede.

Cada um desses atributos é padronizado usando uma nomenclatura única para todos os provedores de nuvem considerados, estabelecida de maneira *ad hoc* para os provedores considerados neste livro. Como trabalho futuro, é proposto o uso de uma ontologia para descrição dos atributos que definem tipos de recurso, como em Cavalcante *et al*. (2012), Ngan e Kanagasabai (2012) e Quinton, Romero e Duchien (2014).

O modelo concreto de recursos também inclui informações sobre recursos instanciados em um ambiente privado. Nesse caso, os tipos de recurso são criados de acordo com a capacidade disponível nos recursos físicos, sendo o custo estabelecido como uma proporção do gasto em energia para seu funcionamento.

Ao analisar o segundo nível da representação proposta, mesmo ao considerar apenas atributos de hardware para descrever os tipos, a estimativa de recursos pode se tornar complexa devido à possível grande quantidade de tipos concretos disponíveis. Além disso, esses tipos possuem intersecção entre eles. Por exemplo, considerando a oferta de VMs em dezembro de 2021, foi analisada a intersecção entre os tipos disponíveis nos provedores AWS, Microsoft Azure e Alibaba.[6] Ao considerar apenas o segundo nível da representação, havia, no total, 1.098 tipos disponibilizados nesses provedores, dos quais para apenas 45,92% não havia duplicidade, ou seja, são tipos únicos.

Apesar da quantidade significativa de tipos idênticos, a intersecção, na prática, é bem maior, pois, ao implantar um serviço deve-se também considerar os tipos que não são idênticos, embora sejam aceitáveis, ou seja, que possuem atributos de hardware com maior capacidade. Apesar de significar desperdício de recurso,

6 Google Cloud e Oracle não foram considerados nesta análise porque nesses provedores a cobrança é realizada individualmente por núcleo de CPU e GB de memória RAM, não fazendo sentido a análise de tipos nesse caso.

pois consideram capacidade adicional na alocação, eles farão parte da solução nos casos em que tipos, com exatamente a capacidade de hardware necessária, não podem ser usados em virtude das demais restrições não funcionais estabelecidas sobre o serviço. A Tabela 1 apresenta o percentual de tipos nesses provedores de acordo com a quantidade de tipos aceitáveis. Por exemplo, para 100% dos tipos considerados, há ao menos um outro aceitável, ao passo que, para 68,71%, há 100 ou mais tipos aceitáveis, e assim suscetivelmente.

TABELA 1
Relação entre a quantidade de tipos aceitáveis e o percentual de tipos concretos para os quais essa quantidade existe nos provedores considerados

Tipos aceitáveis	% de tipos
≥ 1	100
≥ 100	68,71
≥ 200	55,78
≥ 300	46,59
≥ 400	41,83
≥ 500	31,29
≥ 600	22,45
≥ 700	19,73
≥ 800	13,94
≥ 900	6,12

Fonte: Elaboração própria.

Como pode ser visto na tabela, a relação entre a quantidade de tipos aceitáveis e o percentual de tipos concretos que apresentam essa quantidade é significativa. Por exemplo, dos 1.098 tipos concretos considerados, para 55,78% deles é possível obter 200 outros que possuem a mesma capacidade de hardware ou mais. Em virtude dessas intersecções, não é preciso considerar todos os tipos concretos para realizar a estimativa de recursos. Por esse motivo,

é proposta a inclusão de um nível auxiliar na representação do modelo de recursos, que consiste em uma representação sintetizada do segundo nível dos tipos concretos, denominado *modelo canônico de recursos*. As instâncias que compõem esse modelo são denominadas *tipos canônicos* de recurso. A quantificação de atributos de hardware nessas instâncias é especificada de acordo com os tipos concretos considerados em um procedimento que será descrito na próxima seção.

A Figura 23 ilustra os níveis de representação propostos na modelagem de recursos, e como eles se relacionam. A nomenclatura utilizada para referir a tipos concretos no primeiro nível da representação consiste no identificador usado pelos provedores que os disponibilizam, como *t2.nano* e *Standard_A0*, seguido pelo símbolo *@* e pelo identificador da região, por exemplo, *t2.nano@us-west-1* e *Standard_A0@central-us*. Também, para referir ao mesmo tipo concreto, mas no segundo nível da representação, apenas o identificador do tipo é usado. Quanto aos tipos canônicos, identificadores genéricos relacionados à capacidade de hardware provida pelo tipo são usados, por exemplo, *SMALL*, *LARGE* etc.

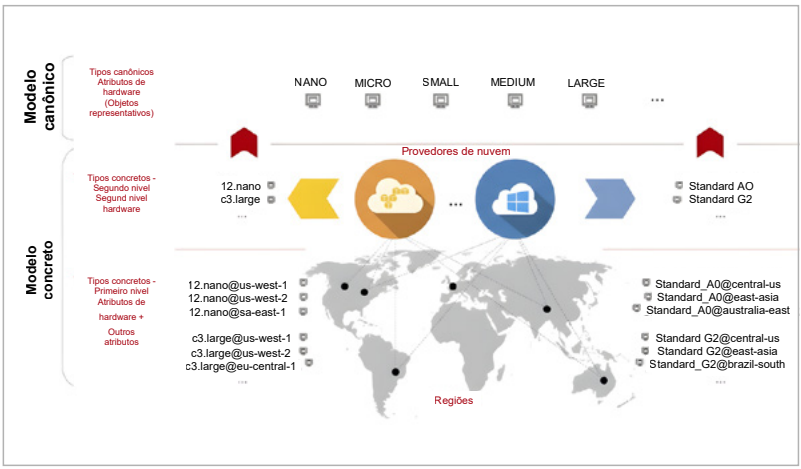


Figura 23 – Modelo de recursos
Fonte: Elaboração própria.

Geração de tipos canônicos de recurso

Ao definir a estratégia para gerar o modelo canônico de recursos, inicialmente foi investigado o uso de clusterização (Berkhin, 2006). A ideia inicial era usar o K-means (Hartigan; Wong, 1979), um algoritmo de particionamento computacionalmente eficiente para agrupar conjuntos de dados n-dimensionais em k *clusters* por meio da redução da variância dentro da classe; e o X-means (Pelleg; Moore, 2000), a versão estendida de K-means, usada para estimar o número de grupos.

Cada atributo de hardware foi definido como uma dimensão, sendo variáveis linguísticas mapeadas para intervalos numéricos. Considerando como parâmetro as dimensões modeladas, o software de mineração de dados Weka (Hall *et al*., 2009) foi usado para gerar os centroides dos *clusters*. Dessa forma, cada centroide seria definido como um tipo canônico. No entanto, esta estratégia apresentou alguns problemas.

O primeiro problema encontrado foi decidir qual medida estatística seria usada para gerar o centroide de cada cluster, uma vez que qualquer categoria de média descartaria tipos cuja capacidade de hardware estivesse abaixo do centroide resultante. Este descarte deve ser adotado porque a estimativa de recursos seria realizada com base nos valores dos centroides, e qualquer capacidade inferior considerada insuficiente. Uma forma de lidar com esse problema seria usar o valor mínimo em cada *cluster* para gerar o centroide, mas os *clusters* gerados nas avaliações realizadas apresentaram intersecção entre eles, ou seja, o valor mínimo em cada *cluster* era menor que o valor máximo do inferior. Por causa disso, os valores mínimos representariam instâncias de múltiplos *clusters*.

Diante dos problemas encontrados, a estratégia adotada como alternativa para a geração do modelo canônico de recursos inicia com a normalização dos atributos de hardware dos tipos

concretos para o intervalo [1-10], usando os valores mínimo e máximo em cada atributo. Em seguida, é realizada a junção dos valores normalizados para cada tipo, resultando em um vetor que quantifica os atributos de hardware do tipo concreto. Os vetores são ordenados, sendo os tipos canônicos gerados a partir deles. Apesar dos problemas descritos anteriormente, o algoritmo X-means é usado como uma heurística para determinar o número de tipos canônicos a serem gerados. Cada tipo canônico é gerado usando percentis e tipos extremos na série ordenada, de acordo com equação a seguir, sendo x o número de clusters retornado pelo algoritmo X-means e $1 \leq i \leq x$:

$$\text{cluster}(i) = \begin{cases} \text{Tipo concreto menos robusto} & , \text{se } i = 1 \\ \dfrac{100}{x-1} \times (i-1) \text{ percentil} & , \text{se } 1 < i < x \\ \text{Tipo concreto mais robusto} & , \text{se } i = x \end{cases}$$

Por exemplo, supondo que o algoritmo X-means estabeleça a existência de seis *clusters*, os tipos canônicos seriam estabelecidos como:

- *Cluster* 1 (NANO): tipo concreto menos robusto (tipo com menor capacidade de hardware entre todos os outros).
- *Cluster* 2 (MICRO): 20º percentil.
- *Cluster* 3 (SMALL): 40º percentil.
- *Cluster* 4 (MEDIUM): 60º percentil.
- *Cluster* 5 (LARGE): 80º percentil.
- *Cluster* 6 (XLARGE): tipo concreto mais robusto (tipo com maior capacidade de hardware entre todos os outros).

Nesse caso, a identificação dos *clusters* (NANO, MICRO etc.) é estabelecida apenas para facilitar a leitura. A Figura 24 ilustra o procedimento utilizado na geração do modelo canônico de recursos.

As hastes indicam cada uma das tarefas realizadas: normalização dos valores dos atributos, ordenação dos tipos concretos considerando o valor total dos atributos normalizados e clusterização usando as propriedades descritas acima. Cada cubo simboliza o valor total da normalização da capacidade de *hardware*, que é usada na geração dos tipos canônicos. Para ser considerado como pertencente ao conjunto representado por um tipo canônico, um tipo concreto deve ter a soma de seus atributos de hardware normalizados, sendo maior ou igual à soma dos atributos estabelecidos para ele. Além disso, essa soma deve ser menor que a soma do tipo canônico subsequente.

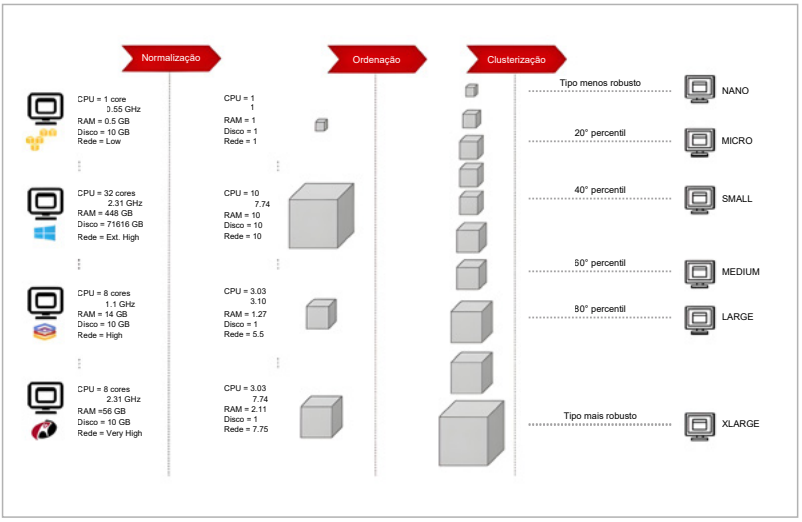


Figura 24 – Estratégia utilizada na geração do modelo canônico de recursos
Fonte: Elaboração própria.

Usando esse procedimento, os problemas descritos anteriormente são solucionados. Outra vantagem é que a quantidade de tipos em cada conjunto se torna balanceada. A única exceção é para o último conjunto (XLARGE, no exemplo), que contém apenas tipos concretos cuja capacidade de hardware é a especificada pelo tipo concreto mais robusto.

Exemplo de geração do modelo canônico de recursos

Para ilustrar o procedimento descrito na seção anterior, o modelo canônico de recursos foi gerado usando os tipos concretos disponibilizados pelos provedores AWS, Microsoft Azure, Google Cloud e Rackspace em maio de 2016. A Figura 25 descreve a localização das regiões disponíveis para cada provedor considerado. Para reduzir a complexidade, zonas de disponibilidade foram combinadas em regiões únicas.

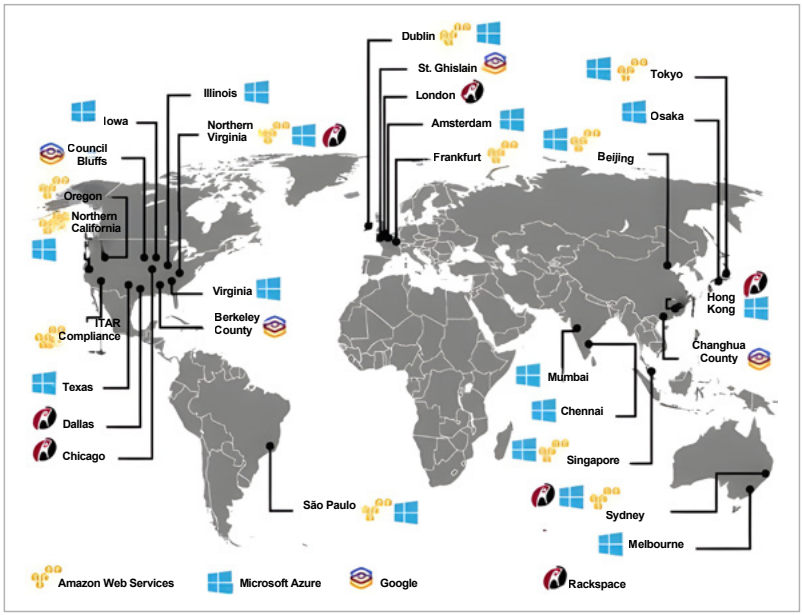


Figura 25 – Regiões utilizadas no exemplo de geração do modelo canônico de recursos
Fonte: Elaboração própria.

A Tabela 2 resume a disponibilidade de recursos para cada provedor de nuvem na data considerada. A segunda e a terceira coluna mostram a quantidade de tipos concretos disponíveis em cada provedor considerando o primeiro e o segundo nível da representação, respectivamente. As colunas seguintes apresentam o valor mínimo e máximo para cada atributo de hardware para tipos disponibilizados por este provedor. Alguns atributos, como desempenho de rede, não foram especificados por todos os provedores considerados.

TABELA 2
Atributos dos tipos concretos disponíveis nos provedores de nuvem pública no período considerado

Provedor de nuvem	# tipos concretos (1° nível)	# tipos concretos (2° nível)	# cores de CPU	clock speed da CPU (GHz)	RAM (GB)	Disco (GB)	Desempenho de rede
Amazon Web Services	352	42	1-40	2,3-3,3	0,5-244	10-48.000	Low-Extremely high
Microsoft Azure	420	61	1-32	0,55-2,537	0,75-488	1.043-71.616	Low-Extremely high
Google Cloud	36	21	1-32	2,4-2,4	0,6-208	3.072-10.240	-
Rackspace	19	9	1-32	-	1-240	20-1.268	-
Total	827	133	1-40	0,55-3,3	0,5-448	10-71.616	Low-Extremely high

Fonte: Elaboração própria.

A Tabela 3 apresenta os tipos canônicos gerados por meio do procedimento descrito ao considerar os tipos concretos disponibilizados pelos provedores. A primeira coluna é o nome usado para identificar o tipo canônico. As colunas seguintes são os valores usados para quantificar cada atributo de hardware. Abaixo de cada valor, entre parênteses, estão os valores mínimo e máximo para esse atributo ao considerar os tipos concretos representados por este tipo canônico. Como pode ser visto, cada atributo de hardware do tipo canônico é definido como o valor mínimo do atributo equivalente nos tipos concretos representados. O uso de valores extremos nesta estratégia pode ser considerado porque os conjuntos gerados não têm intersecção entre eles.

7 Microsoft Azure usava, no período considerado, a medida *Azure Compute Unit* (ACU) para especificar a velocidade de *clock*. A ACU era padronizada em uma VM do tipo *Small* (Standard_A1) sendo 100. Para mapear a ACU para a velocidade de *clock*, foi usada a mesma estratégia de mapeamento utilizada pela Amazon para especificar o desempenho por meio de sua medida *Elastic Compute Unit* (ECU): uma ECU é a potência equivalente de uma CPU com processador 1.0-1.2 GHz 2007 Opteron ou Xeon (Ostermann *et al*., 2010).

TABELA 3
Atributos que caracterizam os tipos canônicos gerados no exemplo

Tipo canônico	# cores de CPU	clock speed da CPU (GHz)	RAM (GB)	Disco (GB)	Desempenho de rede
NANO	1 (1-8)	0,55 (0,55-2,4)	0,5 (0,5-30)	10 (10-10.240)	Low (Low-Moderate)
MICRO	2 (2-16)	1,1 (1,1-2,4)	3,5 (3,5-60)	10 (10-10.240)	Moderate (Moderate-High)
SMALL	4 (4-16)	1,1 (1,1-2,4)	7 (7-104)	10 (10-10.240)	High (High-High)
MEDIUM	8 (8-32)	1,1 (1,1-2,4)	14 (14-208)	10 (10-16.973)	High (High-Very High)
LARGE	8 (8-40)	1,76 (1,76-2,9)	56 (56-244)	10 (10-64.000)	Very High (Very High-Extremely High
XLARGE	32 (32-32)	2,31 (2,31-2,31)	488 (488-488)	76.616 (76.616-76.616)	Extremely High (Extremely High-Extremely High)

Fonte: Elaboração própria.

Nesse capítulo, foi descrita a representação utilizada na modelagem de recursos. Foi proposto um modelo que representa os atributos mais comumente encontrados nos tipos de VM que estão disponíveis em provedores públicos, e diferentes níveis de abstração desses tipos foram criados. Como forma de lidar com a variabilidade de tipos, levando em consideração a intersecção entre eles, além de reduzir o tempo necessário para a estimativa de recursos, foi proposta uma técnica para gerar tipos de recurso representativos, nomeados como tipos canônicos.

Apesar do modelo proposto para representação de recursos não considerar todos os atributos que caracterizam uma VM, argumenta-se que ele é suficiente para guiar a implantação de composições. Os tipos de recursos propostos nesse capítulo são usados como entrada na abordagem de síntese de recursos, que será descrita no próximo capítulo.

5

Síntese de recursos

A principal contribuição deste livro é propor uma abordagem de estimativa e seleção de recursos visando a satisfação de restrições não funcionais na implantação de múltiplas coreografias de serviços. A realização dessas atividades é definida como síntese de recursos e neste capítulo é descrito como elas são realizadas com base na representação conjunta das coreografias e na modelagem de recursos descritos nos capítulos anteriores. Como forma de elucidar os principais elementos desse problema e definir de maneira precisa seu escopo, inicialmente é apresentado sua formalização, tendo como base os conceitos formalizados anteriormente.

Na abordagem de síntese, assume-se que os recursos são instanciados sob demanda, ou seja, não há um conjunto de instâncias de VM pré-criadas. Por essa razão, apenas as atividades relacionadas à estimativa, à descoberta, à seleção e à alocação de recursos entram no escopo a ser discutido.

A estimativa de recursos é realizada com base nas restrições de QoS, pois elas têm influência direta sobre a capacidade de hardware estabelecida para a implantação de cada serviço. As restrições fim-a-fim dessa categoria permitem variar, especialmente, o balanceamento da contribuição de cada serviço relacionado a essas restrições e, assim, obter ganhos maiores na alocação de recursos.

A qualidade geral na estimativa de recursos é medida como uma função global, calculada a partir do valor de utilidade das métricas de QoS consideradas e do ganho ao selecionar um determinado recurso.

Dessa forma, o objetivo da abordagem de estimativa de recursos é indicar uma capacidade apropriada para cada serviço coreografado que satisfaça as restrições impostas, ao mesmo tempo em que maximiza o valor dessa função.

Por outro lado, restrições em relação a atributos do recurso devem ser avaliadas na seleção de recursos, uma vez que elas podem eliminar tipos de recurso candidatos e classificar os tipos de recurso, de forma a definir quais têm maior preferência na decisão sendo tomada.

Este capítulo tem como objetivo apenas apresentar as técnicas propostas para realizar a síntese de recursos. Detalhes de como essas técnicas foram implementadas serão discutidos no próximo capítulo, no qual será apresentada a arquitetura geral da abordagem.

O problema de síntese de recursos

O problema de síntese de recursos sujeita a restrições para múltiplas coreografias de serviços (*Constraint-aware resource Synthesis for multiple service Choreography Problem* – CSDP) consiste em, dado um conjunto de coreografias de serviços, com possível compartilhamento de serviços entre elas, estimar e selecionar os recursos necessários para implantar os serviços pertencentes às coreografias, com o objetivo de satisfazer um conjunto de restrições não funcionais especificadas como parte da entrada.

O conjunto de restrições a serem satisfeitas é definido como K, o qual possui ρ restrições $\{k_1, k_2, \ldots, k_\rho\}$. É definido ρ^q, ρ^e e ρ^r como o número de restrições de QoS, restrições eliminatórias e restrições classificatórias, respectivamente, onde $\rho^q + \rho^e + \rho^r = \rho$. O resultado do CSDP é representado como $F = \{f_1, f_2, \ldots, f_n\}$, onde $f_i, 1 \leq i \leq n$, é um par $(s_i, v_j), s_i \in S, v_j \in V$ que representa o ma-

peamento de um serviço para um recurso que será usado para sua implantação. Consequentemente, $f_i.s$ e $f_i.v$ representam o serviço e o recurso no mapeamento, respectivamente.

Para comparar duas alternativas de mapeamento, é definido φ_{ij}, uma função que representa o ganho oferecido ao alocar o recurso j para implantar o serviço i. Assim, é definido $\varphi(F)$ como o ganho composto de todos os recursos em F. Dessa forma, a solução ótima para o problema é aquela que satisfaz todas as restrições e apresenta o melhor valor de $\varphi(F)$. Esta definição permite que diferentes critérios possam ser utilizados para representar o ganho. Na abordagem, considera-se que o custo de utilização dos recursos é usado como critério para determinar o ganho na seleção.

Como ilustrado na Figura 26, CSDP consiste em decidir o melhor emparelhamento de acordo com as restrições não funcionais especificadas. Essa decisão é tomada avaliando a satisfação das restrições ao se basear nos serviços que compõem as coreografias e no padrão de conexão entre eles. Apesar de supor que essas informações são representadas no grafo de dependências, na resolução do problema outra notação poderia ser utilizada como entrada.

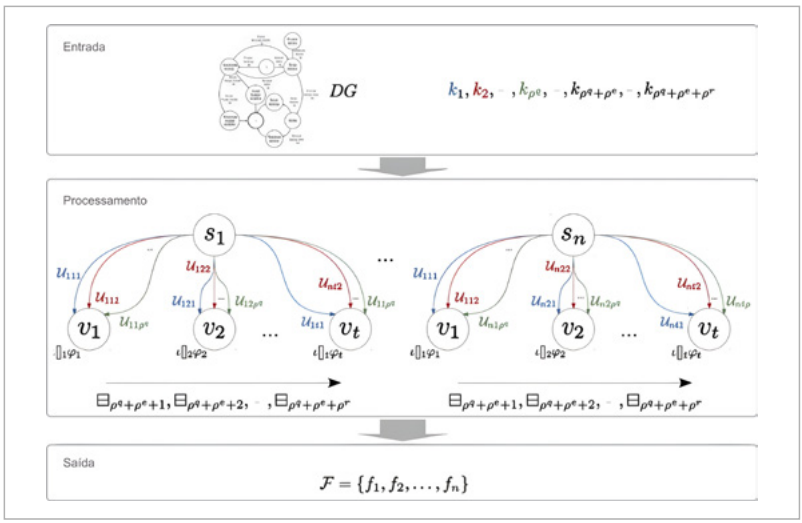


Figura 26 – Elementos do problema CSDP
Fonte: Elaboração própria.

De maneira mais detalhada, dado o conjunto de serviços $\{s_1, s_2, \ldots, s_n\}$ pertencentes às coreografias especificadas, assim como as interligações entre eles; e o conjunto de recursos $\{v_1, v_2, \ldots, v_t\}$ disponíveis para implantar esses serviços, deve-se encontrar um mapeamento de cada serviço para um recurso de forma que todas as restrições não funcionais sejam satisfeitas. A fim de simplificar, não é representado na figura a interligação entre os serviços.

O mapeamento de um serviço para um determinado recurso gera um valor para cada restrição de QoS. Quando o serviço $i, 1 \leq i \leq n$ é mapeado para o recurso $j, 1 \leq j \leq t$, o valor para a restrição de QoS $k, 1 \leq k \leq \rho^q$ é representado como U_{ijk}, onde U é a função de utilidade especificada para essa restrição. A agregação dos valores de QoS, considerando todos os serviços incluídos na restrição em questão, deve estar dentro do limite definido pelo valor-alvo $\tau_k, 1 \leq k \leq \rho^q$ para que ela seja satisfeita. Esse mapeamento também considera o ganho $\varphi_j, 1 \leq j \leq t$ obtido ao selecionar cada recurso.

Restrições de QoS, assim como restrições referentes a atributos do recurso, são especificadas para coreografias isoladas. Assim, um determinado serviço pode ser incluído em diversas restrições, das quais algumas podem ser baseadas na mesma métrica de QoS ou estar relacionadas ao mesmo atributo do recurso, nos casos em que o serviço é compartilhado entre múltiplas coreografias. Para lidar com essa duplicidade, foi proposto o procedimento para a geração da representação conjunta das coreografias, que analisa a possibilidade de junção de restrições. Dessa forma, nos casos em que não é possível juntá-las, cada restrição será analisada separadamente, mesmo que estas restrições sejam baseadas na mesma métrica de QoS ou no mesmo atributo do recurso. Por exemplo, na notação apresentada na Figura 27, U_{121} e U_{122}, representam os valores estimados para as restrições de QoS 1 e 2 quando o serviço 1 é mapeado para o recurso 2. As restrições 1 e 2, por sua vez, podem ser baseadas na mesma métrica de QoS, significando que não foi possível realizar a junção dessas restrições, ou cada uma ser baseada em uma métrica distinta.

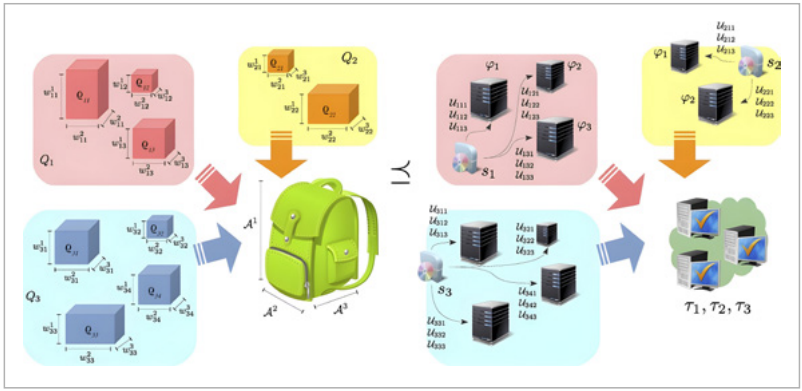


Figura 27 – Redução de uma instância do MMKP para uma instância do CSDP
Fonte: Elaboração própria.

Além das restrições de QoS, o mapeamento deve ser realizado considerando as restrições eliminatórias estabelecidas em função dos atributos $\iota\left[\square\right]_j, 1 \le j \le t$ que descrevem cada recurso. A avaliação dessas restrições é realizada considerando os valores-alvo $\tau_k, \rho^q + 1 \le k \le \rho^q + \rho^e$ definidos para elas.

Adicionalmente, os recursos selecionados devem ser ordenados considerando as restrições classificatórias especificadas. A ordem será obtida de acordo com os operadores de classificação $\boxminus_k, \rho^q + \rho^e + 1 \le k \le \rho^q + \rho^e + \rho^r$ definidos para essas restrições.

CSDP pode ser mais formalmente expresso como:

$$\begin{aligned} \text{Maximizar} \quad & \varphi(F) = \sum_{i=1}^{n} \sum_{j=1}^{t} \varphi_{ij} y_{ij}; \\ & y_{ij} \in \{0,1\}, \\ \text{sujeito a} \quad & \sum_{k=1}^{\rho} x_k^{\circ} = \rho; \quad \sum_{j=1}^{t} y_{ij} = 1, \quad i \in \{1,\ldots,n\}, \end{aligned}$$

A variável y_{ij} é igual a 0, implicando que o recurso j não é selecionado para implantar o serviço i, ou igual a 1 implicando que o recurso j é selecionado para o serviço i. A notação x_k° é usada como uma generalização das variáveis de satisfatibilidade das restrições.

CSDP é claramente um problema de otimização e supostamente não é possível encontrar em tempo polinomial uma solução ótima para uma instância desse problema. Para confirmar isso, é demonstrado que a versão de decisão do CSDP pertence à classe NP-Completo de problemas.

NP-Completude de CSDP

Teorema 1 *CSDP pertence à classe NP-Completo.*

Demonstração. Mesmo CSDP sendo um problema de otimização, por simplicidade, é apresentado a demonstração tratando este como um problema de decisão, isto é, há um limite b para $\varphi(F)$. Esta equivalência é válida, uma vez que ao fazer uma busca binária ao limite b, pode-se transformar uma solução de tempo polinomial para a versão de decisão em um algoritmo de tempo polinomial para o problema de otimização correspondente. A seguir, é apresentada a definição do problema de decisão considerado, usando o formato proposto por Garey e Johnson (1979):

- *INSTÂNCIA:* um conjunto S de serviços; um conjunto V de recursos; para cada $v_j \in V, 1 \leq j \leq t$, um ganho $\varphi(F) \in R^+$; um conjunto K de ρ restrições não funcionais; um valor $b \in R^+$.
- *PERGUNTA:* há uma seleção de exatamente um recurso para cada serviço $s_i \in S, 1 \leq i \leq n$ tal que $\sum_{k=1}^{\rho} x_k^{\circ} = \rho$ e $\sum_{i=1}^{n} \sum_{j=1}^{t} \varphi_{ij} y_{ij} \geq b$?

Em primeiro lugar, se demonstra que CSDP $\in$ NP. É evidente que, dado um certificado representado por um conjunto de n pares $(s_i, v_j), s_i \in S, 1 \leq i \leq n, v_j \in V, 1 \leq j \leq t$ e um limite b, é simples apresentar um algoritmo que verifica se $\sum_{k=1}^{\rho} x_k^{\circ} = \rho$ e $\sum_{i=1}^{n} \sum_{j=1}^{t} \varphi_{ij} y_{ij} \geq b$ em tempo $O(n\rho)$. Esse algoritmo deverá

simplesmente avaliar todas as restrições, considerando cada um dos pares especificados.

A seguir, é demonstrado que CSDP pertence à classe NP-Difícil por meio de uma redução do problema da mochila multidimensional de múltipla escolha (*Multiple-choice Multi-dimension Knapsack Problem* – MMKP) (Moser; Jokanovic; Shiratori, 1997).

MMKP é uma das variantes do Problema da Mochila 0-1 (Sahni, 1975). Ele é um problema de otimização que pode ser definido como: dado um conjunto H de itens divididos em h categorias Q_q, onde cada categoria $Q_q, 1 \le q \le h$, possui $H_q = \left|Q_q\right|$ itens tal que $\forall i, j, 1 \le i \le h, 1 \le j \le h$ e $i \ne j, Q_i \cap Q_j = \varnothing$ e $\cup_{i=1}^{h} Q_i = H$. Cada item $o, o = 1, \ldots, K_q$, da categoria Q_q oferece um valor não negativo como lucro ϱ_{qo}, e possui dimensões dadas por um vetor de peso $W_{qo} = \left\{w_{qo}^1, w_{qo}^2, \ldots, w_{qo}^l\right\}$, em que cada elemento $w_{qo}^a, 1 \le a \le l$ também é um valor positivo. As dimensões da mochila são dadas pelo vetor $A = \left\{A^1, A^2, \ldots, A^l\right\}$. O objetivo do MMKP é selecionar exatamente um item de cada categoria de forma a maximizar o lucro total, sujeito às dimensões da mochila.

Formalmente, o MMKP pode ser definido como:

$$\begin{aligned} \text{Maximizar} \quad & Z(Y) = \sum_{q=1}^{h} \sum_{o=1 ij}^{K_q} y_{ij}; \\ & y_{ij} \in \{0,1\}, \\ \text{sujeito a} \quad & \sum_{k=1}^{\rho} x_k^{\circ} = \rho; \quad \sum_{j=1}^{t} y_{ij} = 1, \quad i \in \{1, \ldots, n\}, \end{aligned}$$

A variável y_{qo} é igual a 0, implicando que o item o da categoria Q_q não é selecionado, ou igual a 1 implicando que o item o da categoria Q_q é selecionado.

A versão de decisão para o problema MMKP é definida a seguir.

- *INSTÂNCIA:* um conjunto H de itens divididos em h categorias Q_q, onde cada categoria $Q_q, 1 \le q \le h$, possui $K_q = \left|Q_q\right|$ itens tal que

  $\forall i, j, 1 \le i \le h, 1 \le j \le h$ e $i \ne j, Q_i \cap Q_j = \varnothing$ e $\cup_{i=1}^{h} Q_i = H$ para cada item $o \in Q_q$ um lucro $\varrho_{qo} \in R^+$, e dimensões dadas por um vetor $W_{qo} = \left\{w_{qo}^1, w_{qo}^2, \ldots, w_{qo}^l\right\}$, em que cada elemento $w_{qo}^a \in R^{+, 1 \le a \le l}$; as dimensões da mochila dadas pelo vetor $A = \left\{A^1, A^2, \ldots, A^l\right\}$, no qual cada elemento $A^a \in R^{+, 1 \le a \le l}$; um valor $b \in R^+$.

- *PERGUNTA:* há uma seleção de exatamente um elemento de cada categoria $Q_q, 1 \le q \le h$ tal que $\sum_{q=1}^{h} \sum_{o=1}^{K_q} w_{qo}^a y_{qo} \le A^a, 1 \le a \le l$ e $\sum_{q=1}^{h} \sum_{o=1}^{K_q} \varrho_{qo} y_{qo} \ge b$?

Dado que MMKP pertence à classe NP-Completo (Moser; Jokanovic; Shiratori, 1997), será demonstrado que MMKP ∞_{P} CSDP.

Dada uma instância do MMKP, ou seja,

$$H, h, \left\{Q_1, \ldots, Q_q\right\}, K_q, \left\{\varrho_{11}, \ldots, \varrho_{1K_1}, \ldots, \varrho_{q1}, \ldots, \varrho_{qK_q}\right\}, \left\{W_{11}, \ldots, W_{1K_1}, \ldots, W_{q1}, \ldots, W_{qK_q}\right\}, A, b$$

será construído uma instância do CSDP. Essa redução é ilustrada na Figura 27. Seja $n = h$. Embora no CSDP todos os elementos no conjunto de recursos são analisados para cada serviço, é assumido, sem perda de generalidade, que para cada serviço há um conjunto diferente de recursos disponíveis para sua implantação. Essa suposição restringe ainda mais o problema, mas não altera sua definição. Ao fazer isso, é possível mapear H para V e supor cada Q_q como sendo o conjunto de recursos para um determinado serviço. Seja W_{qo} a função de utilidade empregada na avaliação de uma restrição de QoS, de forma que $\left(w_{qo}^1, w_{qo}^2, \ldots, w_{qo}^l\right)$ é mapeado para $\left(U_{ij1}, U_{ij2}, \ldots, U_{ij\rho}\right)$ e ϱ_{qo} é mapeado para φ_j. Nesse caso, é assumido que na instância criada para o CSDP, $\rho^e = 0$ e $\rho^r = 0$.

Consequentemente, cada dimensão $A^a, a = 1, \ldots, l$ é mapeada para o valor-alvo τ_k da restrição. Considera-se que o limite b é o mesmo para as instâncias dos dois problemas. Claramente, essa construção pode ser feita em tempo polinomial. Em seguida, é necessário demonstrar que os itens selecionados na instância original do MMKP satisfazem todas as restrições e possuem lucro $\geq b$ se e somente se os recursos selecionados na instância criada do CSDP também satisfazem todas as restrições e oferecem um ganho $\geq b$.

Inicialmente, suponha que há uma seleção de um único recurso para cada serviço (considerando todas as coreografias) que satisfaça todas as restrições e ofereça um ganho $\varphi(F) = b$, ou seja, $U_{11k} \oplus \ldots \oplus U_{1tk} \oplus U_{21k} \oplus \ldots \oplus U_{2tk} \oplus \ldots \oplus U_{ntk} \leq \tau_k, 1 \leq k \leq \rho$ onde $U_{ijk}, 1 \leq i \leq n, 1 \leq j \leq t$ é o elemento neutro da métrica de QoS referente à restrição k quando não existir par (s_i, v_j); $\sum_{k=1}^{\rho} x_k^q = \rho$ e $\sum_{i=1}^{n} \sum_{j=1}^{t} \varphi_{ij} y_{ij} = b$. Pela construção realizada, isso significa que $\sum_{q=1}^{h} \sum_{o=1}^{K_q} w_{qo}^a y_{qo} \leq A^a, a \in \{1, \ldots, l\}$, ou seja, os itens equivalentes aos recursos selecionados não ultrapassarão os limites da mochila. Tendo a instância do CSDP solução, para a instância equivalente do MMKP, $\sum_{o=1}^{H_q} y_{qo} = 1, q \in \{1, \ldots, h\}$, isto é, somente um item de cada categoria será selecionado. Como havia sido suposto que $\varphi(F) = b$, para a instância do MMKP a relação $Z(Y) = b$ também será verdadeira.

Por outro lado, suponha que a instância do MMKP tem solução tal que $Z(Y) = b$. Dessa forma, $\sum_{q=1}^{h} \sum_{o=1}^{K_q} w_{qo}^a y_{qo} \leq A^a, a \in \{1, \ldots, l\}$. Similarmente, pela construção realizada, isso significa que $U_{11k} \oplus \ldots \oplus U_{1tk} \oplus U_{21k} \oplus \ldots \oplus U_{2tk} \oplus \ldots \oplus U_{ntk} \leq \tau_k, 1 \leq k \leq \rho$ ou seja, todas as restrições são satisfeitas (considerando todas as coreografias): $\sum_{k=1}^{\rho} x_k^q = \rho$. Tendo a instância do MMKP solução, $\sum_{o=1}^{K_q} y_{qo} = 1, 1 \leq q \leq h$, isto é, somente um item de cada categoria é selecionado, o que significa que $|F| = n$, implicando que há um par com cada serviço que o mapeia a um recurso.

Como havia sido suposto que $Z(Y) = b, \varphi(F) = b$ e a instância do CSDP criada também possui solução. Inversamente, suponha que a instância do MMKP não tenha solução. Assim sendo, para satisfazer a condição $\sum_{o=1}^{K_q} y_{qo} = 1, 1 \leq q \leq h$, tem-se que $\exists a \in \{1, \ldots, l\} \vee \sum_{q=1}^{h} \sum_{o=1}^{K_q} w_{qo}^{a} y_{qo} > A^{a}$. Pela construção realizada, a restrição correspondente $k, 1 \leq k \leq \rho$ sobre a é também avaliada como $U_{11k} \oplus \ldots \oplus U_{1tk} \oplus U_{21k} \oplus \ldots \oplus U_{2tk} \oplus \ldots \oplus U_{ntk} > \tau_k$, isto é, não é possível satisfazer essa restrição e, consequentemente, a instância do CSDP não tem solução.

Portanto, foi demonstrado que MMKP $\preccurlyeq_{\mathrm{P}}$ CSDP. Assim, CSDP pertence à classe NP-difícil, e uma vez que foi demonstrado que CSDP pertence à classe NP, CSDP pertence à classe NP-completo.

Estimativa de recursos

A estimativa de recursos está diretamente relacionada a propriedades não funcionais esperadas para os serviços coreografados. Mais precisamente, essa atividade é guiada pelas restrições de QoS que devem ser satisfeitas, tornando a estimativa de QoS um dos aspectos mais importantes na definição da capacidade dos recursos.

Conforme discutido anteriormente, uma restrição de QoS pode ser especificada para um único serviço/operação ou pode restringir valores fim-a-fim na encenação de uma coreografia. No primeiro caso, basta analisar a função de utilidade especificada em cada restrição de QoS e verificar se sua estimativa está dentro do valor-alvo. Por outro lado, a avaliação de QoS fim-a-fim é influenciada pela topologia da coreografia e, nesse caso, a estimativa de QoS é obtida de forma diferente para cada padrão de conexão.

Na abordagem, a influência de cada padrão de conexão na QoS fim-a-fim é avaliada, de forma indireta, com base na representação conjunta das coreografias (grafo de dependências). Usando essa estrutura, é possível avaliar restrições fim-a-fim mesmo sem a representa-

ção explícita dos conectores das composições originais, considerando que:

1. O padrão de sequência é explicitamente modelado por meio das arestas do grafo.
2. Quando há um conector de disjunção mutuamente exclusiva, a probabilidade de execução de cada opção no fluxo é refletida na carga quando o grafo de dependências é gerado. Dessa forma, quando a função de utilidade é calculada para serviços incluídos em fluxos com este padrão, a probabilidade é aplicada indiretamente.
3. Quando há conectores de conjunção e disjunção, as informações sobre fluxos executados em paralelo são preservadas nas estruturas de árvore de paralelismo incluídas na representação dos grupos em restrição. Conforme será detalhado no próximo capítulo, é considerada a estimativa de QoS para cada um dos fluxos executados em paralelo sendo estabelecida aquela mais restritiva como estimativa final. Ao adotar esse procedimento, a estimativa de QoS obtida será a mesma que aquela resultante da avaliação do conector original.

A estimativa de recursos é realizada com base no modelo canônico de recursos. A utilização desse modelo em detrimento do uso do modelo concreto se justifica pelo fato do modelo canônico oferecer uma visão geral dos níveis de capacidade oferecidos nos tipos disponíveis, ao mesmo tempo que permite reduzir significativamente o espaço de busca.

Um dos principais desafios para a estimativa de recursos no problema considerado é lidar com interferências sobre a satisfação de restrições de QoS devido ao compartilhamento de serviços. Para oferecer uma solução eficiente para a estimativa de recursos sem o custo associado a uma estratégia de busca exaustiva, é utilizado uma heurística para a solução do problema MMKP. Usando em ordem inversa a redução apresentada para demonstrar a NP-com-

pletude de CSDP, é construído uma instância do problema MMKP a partir de uma instância do problema CSDP. Então é utilizada a heurística WS-HEU (Yu; Zhang; Lin, 2007), que resolve instâncias do MMKP e gera soluções que chegam a 96% do valor da solução ótima. Esta heurística possui complexidade de tempo no pior caso de $O\left(\rho^q n^2 \left(\psi - 1\right)^2\right)$, em que ρ^q é número de restrições de QoS, n é o número de serviços, e ψ é o número de tipos canônicos.

WS-HEU é uma versão modificada da heurística HEU (Khan, 1998), baseada no conceito de consumo agregado ao escolher um item candidato em um grupo para resolver o MMKP. A original encontra uma solução atualizando os itens selecionados de cada grupo. A WS-HEU começa com um pré-processamento para encontrar uma solução viável inicial, isto é, uma combinação de itens que satisfaz todas as restrições, mas que não necessariamente é a melhor. Uma etapa de pós-processamento melhora o valor de utilidade total da solução, com uma atualização, seguida por um ou mais retrocessos.

Na versão original do problema MMKP e, consequentemente na heurística WS-HEU, as restrições são aplicadas a todos os grupos em que itens devem ser selecionados. Dessa forma, a heurística foi modificada para que as restrições possam ser aplicadas apenas a um subconjunto dos grupos, que no caso em questão equivalem aos serviços. A necessidade dessa modificação ocorre porque no problema tratado as restrições são especificadas para coreografias isoladas e, assim, não serão sempre aplicadas a todos os serviços. Apesar da falta de demonstração formal, argumenta-se que ela não altera o problema original de forma a invalidar a heurística. Esse argumento é embasado no fato das restrições serem avaliadas de acordo com o valor agregado da contribuição dos serviços (grupos) a que elas se referem, sendo que informações sobre esses serviços não são usadas de forma direta na decisão. Com isso, considerar

todos os serviços ou somente um subconjunto deles não interfere no funcionamento da heurística.

Na implementação da estimativa de recursos, o procedimento começa pela avaliação das restrições de QoS especificadas para serviços isolados (restrições locais). Cada restrição com esse perfil é avaliada considerando os tipos canônicos de recurso disponíveis. Dessa forma, os tipos canônicos que não possuem capacidade suficiente para satisfazer alguma das restrições locais para um determinado serviço são descartados na subsequente avaliação de restrições fim-a-fim especificadas sobre coreografias que contém este serviço. Caso todos os tipos canônicos sejam descartados nessa etapa para um determinado serviço, é adicionada uma nova instância como forma de reduzir a demanda de recurso sobre instâncias individuais desse serviço. Em seguida, o procedimento é reiniciado. Esse processo é iterado até que haja ao menos um tipo canônico para cada serviço que possa satisfazer todas as restrições de QoS locais sobre ele.

A avaliação de restrições de QoS fim-a-fim inicia com o cálculo do peso de todos os pares de serviço e recurso, a partir da contribuição agregada desse serviço em todas as restrições de QoS fim-a-fim especificadas sobre coreografias que o contêm. Para calcular essa agregação, os valores da contribuição em cada restrição de QoS são mapeados para um valor numérico que representa o peso. Para métricas cujo domínio já é um valor numérico (como latência), isso é direto. Contudo, para métricas cujo domínio é formado por valores nominais, deve ser definido um valor numérico equivalente a cada um dos valores nominais.

Para se obter a contribuição de um serviço em uma restrição, é verificado se este serviço faz parte de alguma árvore de paralelismo relacionada a ela. Nos casos em que o serviço não está inserido em uma árvore de paralelismo, a contribuição é estabelecida como o valor da função de utilidade estimado para a métrica. Quando o servi-

ço está inserido em uma árvore de paralelismo, a contribuição desse serviço deve ser calculada levando em conta a contribuição (nesta restrição) dos demais serviços inseridos na árvore. Nesse caso, o procedimento para obter a contribuição consiste em verificar o ramo da árvore que oferece o pior valor de QoS. Caso o serviço esteja inserido nesse ramo a contribuição é estabelecida como o valor da função de utilidade para esse serviço. Caso contrário, sua contribuição é estabelecida como zero, uma vez que a contribuição que prevalece é aquela definida pelos serviços que formam o ramo com pior QoS, devendo, então, a contribuição de serviços fora desse ramo ser ignorada.

Após o cálculo da contribuição de QoS, os pares são ordenados pela razão entre o valor do recurso e o peso calculado. O valor do recurso, nesse caso, refere-se ao ganho obtido ao selecioná-lo. A estratégia imediata para estabelecer esse atributo seria considerar o inverso do custo monetário, de forma que recursos com menor custo devem ser privilegiados na seleção. Contudo, uma vez que não faz sentido incluir a representação do custo monetário como atributo nos tipos canônicos, pois eles representam uma generalização dos tipos concretos, o valor do recurso é definido como

$$\frac{1}{r}$$

em que r é a agregação dos atributos de hardware normalizados. Como consequência, os tipos canônicos com maior capacidade de hardware apresentam menor valor.

Esta estratégia foi adotada como uma heurística para influenciar a seleção de tipos concretos com menor custo, uma vez que nos provedores de nuvem pública o custo de um tipo concreto é diretamente proporcional à capacidade de hardware. Essa heurística não é ideal em todos os casos, visto que esse comportamento não é verificado para tipos alocados em nuvens privadas. Contudo, o

estabelecimento de uma estratégia mais precisa para a definição do valor dos tipos canônicos é tido como trabalho futuro.

Tendo como entrada os pares ordenados, uma solução é construída escolhendo para cada serviço o par com maior razão. Após isso, as restrições fim-a-fim são verificadas e, se não forem satisfeitas, novas soluções são criadas considerando os pares subsequentes até que todas as restrições sejam satisfeitas. Quando, mesmo considerando todos os pares nesta etapa, nenhuma solução viável é encontrada, novas instâncias dos serviços devem ser incluídas. A estratégia implementada para realizar a adição de instâncias de serviços será discutida no próximo capítulo.

Após a inclusão de novas instâncias dos serviços, nos casos em que isso é necessário, o procedimento é reiniciado, refazendo a avaliação das restrições locais para os serviços que tiveram instâncias adicionadas. Em seguida, o peso das contribuições é novamente calculado e os pares de serviço e recurso são ordenados pela razão entre o valor do recurso e o peso calculado, buscando-se uma solução viável. Caso ainda assim nenhuma solução seja encontrada, todo o processo de estimativa é repetido novamente. Caso contrário, a heurística tenta melhorar a solução encontrada localmente para cada serviço usando a técnica *simulated annealing* (Aarts; Korst, 1991).

Esta é uma meta-heurística para otimização que realiza busca local probabilística e se fundamenta numa analogia com a termodinâmica. *Annealing* é o processo utilizado para fundir um metal, no qual é aquecido a uma temperatura elevada e em seguida é resfriado lentamente, de modo que o produto final seja uma massa homogênea. Neste contexto, o processo de otimização simula os níveis de temperatura no resfriamento. Em cada nível, dado um ponto no espaço de solução, vários pontos na vizinhança são gerados e o correspondente valor da função sendo otimizada é calculado. Cada ponto gerado é aceito ou rejeitado de acordo com uma certa probabilidade. Esta probabilidade de aceitação decresce

de acordo com o nível do processo, ou equivalentemente, de acordo com a temperatura (Haeser; Ruggiero, 2008).

A saída da heurística WS-HEU é um mapeamento de cada serviço na estrutura do grafo de dependências (e, consequentemente, suas prováveis instâncias adicionais) para o tipo canônico selecionado. Ao utilizar essa estratégia, a capacidade de hardware mais adequada é selecionada a partir dos tipos canônicos de recurso disponíveis, ao passo que a contribuição nas restrições fim-a-fim é balanceada entre os serviços do conjunto de coreografias. A estimativa de recursos é, portanto, obtida como uma consequência do emparelhamento. Os tipos canônicos selecionados são usados na próxima etapa da síntese, que realiza a seleção de recursos.

Seleção de recursos

A segunda etapa consiste na seleção de recursos. Essa atividade é implementada na abordagem por meio do mapeamento dos tipos canônicos selecionados na primeira etapa da síntese para tipos concretos de recurso. As restrições relacionadas a atributos do recurso são avaliadas nessa etapa, pois elas determinam os tipos concretos que são aceitáveis para cada serviço.

Na seleção de recursos, além dos tipos concretos inicialmente descobertos, considera-se a inclusão de novos tipos criados como uma aproximação da capacidade de hardware estabelecida em cada tipo canônico selecionado. Eles são incluídos quando se considera o uso de uma nuvem privada com capacidade disponível ou quando a funcionalidade de criação de tipos de VM customizáveis é oferecida pelos provedores de nuvem pública considerados. A criação de tipos customizáveis é uma funcionalidade que passou a ser oferecida por alguns provedores, por exemplo, Google, como uma alternativa nos cenários em que os tipos predefinidos não satisfazem de maneira eficiente as necessidades da aplicação. Contudo, a inclusão desses

tipos na abordagem não descarta a utilização de tipos concretos predefinidos, considerando que:

1. Ainda é limitado o número de provedores de nuvem que permitem a criação de tipos de VM customizáveis. Além disso, não há total liberdade na configuração desses tipos, visto que alguns provedores impõem regras sobre sua criação, como por exemplo, limitar o número de núcleos de CPU a números pares ou o total de memória a múltiplos de uma determinada constante. Outro problema é que essas regras podem variar para cada região. Dessa forma, ao considerar apenas o uso de tipos de VM customizáveis seria difícil satisfazer todas as restrições não funcionais requeridas. O uso dos tipos concretos preestabelecidos faz com que as opções na seleção de recursos sejam aumentadas significativamente, favorecendo o processo de implantação de serviços.
2. Os tipos concretos predefinidos são necessários para a geração dos tipos canônicos. Outra estratégia para a geração dos tipos canônicos poderia ser a geração de tipos com capacidade estabelecida de maneira empírica, ou seja, pela variação da capacidade considerando faixas de valores, como por exemplo, entre 1 GB e 100 GB para memória, ou usando uma estratégia equivalente. Contudo, a alocação de tipos concretos considerando as capacidades estabelecidas seria um desafio, pelos motivos citados acima.

Após a inclusão dos novos tipos, a seleção de recursos é realizada pela filtragem e classificação dos tipos concretos representados por tipo canônico. Visando uma seleção de recursos ainda mais eficiente, é investigada a possibilidade de alocar um único recurso para implantar múltiplos serviços, ou seja, consolidar a seleção inicialmente realizada para um conjunto de serviços em um único recurso. Essa etapa é definida como *consolidação de recursos*, que con-

siste na última etapa realizada na seleção de recursos. A seguir, são discutidos maiores detalhes sobre cada etapa.

Filtragem e classificação de recursos

Em virtude da estratégia utilizada para geração dos tipos canônicos, pode-se garantir que todos os tipos concretos representados por um determinado tipo canônico têm, no mínimo, a mesma capacidade de hardware descrita por ele. Dessa forma, essa etapa da seleção de recursos pode ser realizada com base apenas nas restrições relativas a atributos do recurso. Para tal, a filtragem e a classificação podem ser aplicadas para cada serviço separadamente, uma vez que as restrições em relação a atributos do recurso são especificadas para serviços isolados. Para permitir seu processamento, é adicionado na taxonomia originalmente apresentada na Figura 13 um nível de categorização dessas restrições, conforme pode ser visto na Figura 28.

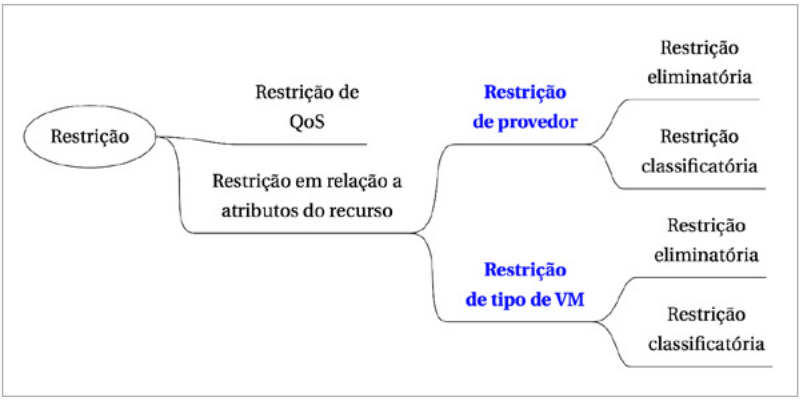


Figura 28 – Taxonomia de restrições atualizada
Fonte: Elaboração própria.

Restrições de provedor referem-se a atributos que caracterizam o provedor e, consequentemente, aplicam-se a todos os tipos de VM

disponibilizados por ele. Exemplos de restrições nessa categoria são: evitar provedores que requerem cobrança mínima dos recursos utilizados (restrição eliminatória) e privilegiar a seleção de recursos de provedores com maior reputação (restrição classificatória).

Por outro lado, *restrições de tipos de VM* dizem respeito ao tipo concreto propriamente dito e não têm relação direta com atributos que caracterizam o provedor. Exemplos de restrições nessa categoria são: selecionar recursos que serão alocados em um determinado país (restrição eliminatória) e usar o custo de utilização como critério de escolha entre os tipos de VM disponíveis (restrição classificatória).

De acordo com a definição em que as restrições são especificadas por meio da linguagem proposta, geram-se filtros e classificadores que são aplicados sobre os tipos concretos. A ordem de aplicação é definida de acordo com a taxonomia atualizada. Visando diminuir o tempo necessário para obter o resultado da seleção de recursos, restrições eliminatórias são avaliadas primeiro. Pelo mesmo motivo, restrições de provedor têm preferência sobre restrições de tipos de VM, uma vez que elas permitem uma redução mais imediata no espaço de busca. Com base nisso, as restrições em relação a atributos do recurso são avaliadas na seguinte sequência:

1. restrições eliminatórias de provedor;
2. restrições eliminatórias de tipo de VM;
3. restrições classificatórias de provedor;
4. restrições classificatórias de tipo de VM.

A ordem estabelecida dos tipos concretos de recurso depende da sequência com que as restrições classificatórias são avaliadas, uma vez que a ordem parcial estabelecida por uma restrição é mantida ao avaliar a seguinte. Mais precisamente, a sequência de avaliação das restrições classificatórias é estabelecida privilegiando restrições de provedor e, em seguida, de tipo de VM, conforme esta-

belecido anteriormente. No caso de haver restrições em uma mesma categoria, a sequência é estabelecida usando a de antes. Contudo, ela pode ser definida de forma diferente por meio da especificação (pelo usuário) da prioridade de cada restrição classificatória.

Tendo como base a sequência estabelecida, cada restrição classificatória só é avaliada nos casos em que não é possível estabelecer a ordem apenas considerando a restrição precedente, ou seja, quando os tipos de recurso são iguais de acordo com o critério usado na ordenação. Se esta estratégia não fosse adotada, a única ordem garantida seria aquela da última restrição avaliada, sendo que a ordem estabelecida por restrições de maior prioridade poderia ser desfeita em alguns casos. Porém, é importante ressaltar que esse procedimento desconsidera a avaliação de algumas restrições classificatórias estabelecidas, mas é justificável porque, em alguns casos, não é possível realizar a ordenação de um conjunto considerando muitos critérios.

Como forma de simplificar a implementação, os algoritmos propostos para a filtragem e a classificação dos recursos são baseados na análise individual de cada tipo concreto disponível. Como trabalho futuro, é sugerido o uso de alguma heurística para tornar essa etapa mais eficiente, como restringir o conjunto de tipos que devem ser analisados por meio de algum critério probabilístico.

Resultante da filtragem e da classificação dos recursos, tem-se um conjunto de tipos concretos aceitáveis para cada serviço, ordenados de acordo com as restrições classificatórias. Alternativamente, para alguns serviços pode-se ter um conjunto vazio, significando que não há nenhum tipo concreto que satisfaça todas as restrições em relação a atributos do recurso estabelecidas. Nesse caso, a estimativa de recursos deve ser refeita de forma que outros tipos canônicos sejam selecionados para os serviços sem sucesso.

Quando o resultado da seleção de recursos obtém ao menos um tipo concreto para cada serviço, a próxima etapa consiste em verificar a possibilidade de consolidar os recursos selecionados.

Consolidação de recursos

O uso do modelo canônico de recursos permite que os tipos concretos sejam agregados em conjuntos representados pelos tipos canônicos. Porém, como cada um deles é definido como a capacidade de hardware mais restrita no conjunto, pode ocorrer situações em que o tipo concreto selecionado tenha capacidade excedente àquela realmente necessária para o serviço. Além disso, mesmo com a aplicação das restrições em relação a atributos do recurso, o conjunto de tipos concretos aceitáveis para um determinado serviço pode ser grande.

Como forma de se beneficiar da variedade de tipos aceitáveis e aproveitar a provável capacidade excedente, evitando desperdícios na alocação de recursos, é proposta uma estratégia de consolidação de recursos.

A consolidação de recursos consiste em implantar dois ou mais serviços em um único recurso em vez de alocá-lo isoladamente a cada serviço. A consolidação pode envolver a troca de um tipo concreto por outro que também satisfaça as restrições estabelecidas e que permita a consolidação. Este pode ser considerado uma variação do problema de compartimento (*bin packing*) heterogêneo ou de tamanho variável (*Variable Size Bin Packing Problem* – VSBPP) (Correia; Gouveia; Saldanha-Da-Gama, 2008; Friesen; Langston, 1986; Kang; Park, 2003).

No VSBPP, dado um conjunto de itens com um certo peso cada e um conjunto de compartimentos particionados em várias classes, em que cada uma corresponde a compartimentos de uma dada capacidade e um certo custo, o objetivo é empacotar todos os itens nos compartimentos e minimizar o custo total associado aos utilizados. Para tal, é assumido que cada classe tem um número ilimitado de compartimentos. Este problema pertence à classe NP-Completo (Rhee; Talagrand, 1987), o que evidencia sua complexidade.

A estratégia de consolidação de recursos pode ser realizada quando há capacidade suficiente para satisfazer a demanda de todos os serviços envolvidos na consolidação e ao custo dos recursos selecionados. Nesse caso, o custo é tido como satisfatório quando

se mantém o mesmo que o custo obtido ao alocar cada serviço separadamente, ou apresenta um acréscimo adicional suportado, de acordo com a configuração do usuário.

A Figura 29 apresenta a situação em que a consolidação pode ocorrer, com as regras que devem ser satisfeitas para que ela ocorra.

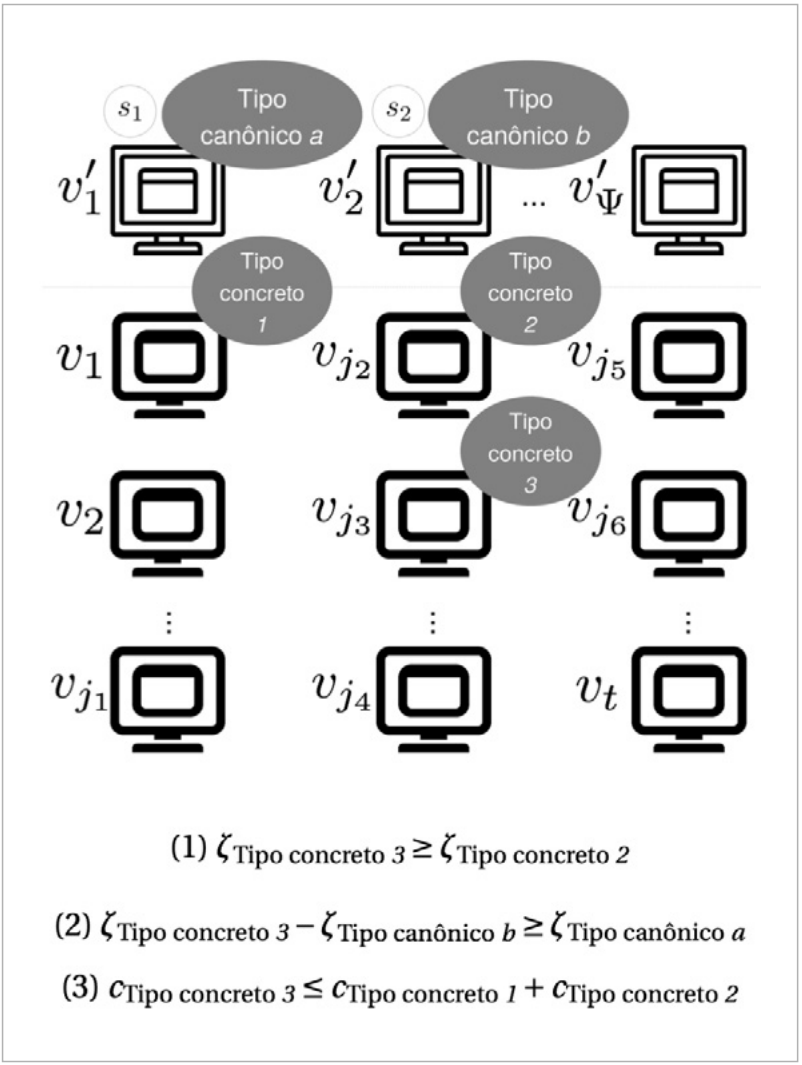


Figura 29 – Regras para consolidação de recursos
Fonte: Elaboração própria.

Nessa figura, s_1 e s_2 são os serviços para os quais os recursos estão sendo analisados, ζ_v e c_v. especificam a capacidade e o custo de utilização, respectivamente, do recurso v,o passo que $\left\{v_1', v_2', \ldots, v_\psi'\right\}$ representam os tipos canônicos mapeados na estimativa de recursos, $\left\{v_1, v_2, \ldots, v_{j_1}, \ldots, v_{j_2}, \ldots, v_t\right\}$ representam os tipos concretos representados pelos tipos canônicos considerados, e "Tipo canônico *a*", "Tipo canônico *b*", "Tipo concreto *1*", "Tipo concreto *2*", "Tipo concreto *3*" são papéis assumidos por esses tipos nas regras apresentadas, durante uma busca por consolidação. Essas regras representam condições gerais que devem ser atendidas para que a consolidação de recursos ocorra. Elas são detalhadas a seguir:

1. A capacidade do novo tipo concreto selecionado deve ser igual ou superior à capacidade do tipo concreto atualmente selecionado.
2. A capacidade excedente do novo tipo concreto selecionado, isto é, a capacidade adicional à capacidade especificada pelo tipo canônico que representa este tipo concreto, deve ser igual ou superior à capacidade dos tipos canônicos selecionados para os demais serviços envolvidos na consolidação.
3. O custo do novo tipo concreto selecionado deve ser igual ou inferior à soma dos tipos concretos atualmente selecionados para os serviços envolvidos na consolidação. Alternativamente, pode-se estabelecer um limite aceitável de gasto adicional.

Para verificar se essas regras realmente podem ser satisfeitas, analisa-se os tipos concretos usados no exemplo apresentado anteriormente. Considera-se todos os tipos concretos disponíveis nos provedores utilizados no exemplo, de forma que nenhuma restrição em relação a atributos do recurso é aplicada sobre eles.

Para cada tipo concreto, foi verificado se as regras (1), (2) e (3), descritas anteriormente, podem ser satisfeitas considerando tipos concretos em conjuntos representados por tipos canônicos com menor capacidade. Essa verificação foi realizada de forma isolada para cada par de conjuntos referentes aos tipos canônicos. Por exemplo,

os tipos concretos referentes ao tipo canônico SMALL foram verificados inicialmente considerando os tipos concretos relacionados ao tipo canônico MICRO e, em seguida, aos tipos concretos referentes ao tipo canônico NANO. Foi contabilizado, então, para cada conjunto, a quantidade de tipos para os quais as três regras foram satisfeitas para ao menos um tipo do conjunto comparado.

A Tabela 4 apresenta o resultado dessa análise. Cada linha/coluna equivale ao conjunto representado pelo tipo canônico descrito. Cada célula $[i][j]$ (na qual i é a linha e j é a coluna) indica o percentual de tipos concretos no conjunto i para os quais as três regras são satisfeitas ao comparar com tipos concretos do conjunto j. Por exemplo, para 52,53% dos tipos concretos no conjunto MICRO é possível consolidar ao menos um recurso do conjunto NANO.

TABELA 4
Percentual de tipos concretos que satisfazem as regras de consolidação comparando-os com tipos em conjuntos mais restritos

	NANO	MICRO	SMALL	MEDIUM	LARGE
MICRO	52,53%				
SMALL	31,46%	0,00%			
MEDIUM	37,46%	5,28%	0,00%		
LARGE	79,94%	73,35%	70,13%	54,19%	
XLARGE	0,00%	0,00%	0,00%	0,00%	0,00%

Fonte: Elaboração própria.

Com exceção de tipos XLARGE, para os quais não há capacidade adicional uma vez que eles representam um tipo concreto específico, e de alguns casos em que tipos de um conjunto são comparados com tipos no conjunto diretamente inferior (SMALL com MICRO e MEDIUM com SMALL), a consolidação de recursos seria possível para grande parte dos tipos concretos. Neste exemplo, essa quantidade chegou a quase 80% dos casos (ao se comparar tipos de LARGE com tipos de NANO).

Além de reduzir o número de VMs alocadas, a consolidação de recursos tem como objetivo reduzir o atraso de comunicação entre os serviços. Para isso, foi usada a demanda de comunicação entre os serviços (representada pela carga λ sobre as operações) como critério para

guiar as decisões relacionadas à consolidação de recursos. Outro critério utilizado é o ganho obtido ao considerar um determinado tipo concreto. Esse ganho é estabelecido em função de uma variável chamada de *fator de comunicação*. Essa variável avalia a redução no atraso de comunicação ao considerar um serviço e o tipo concreto selecionado para ele, e os serviços adjacentes a ele, com os respectivos tipos concretos selecionados.

O fator de comunicação é calculado de acordo com a equação a seguir como sendo a média dos valores de ganho de comunicação ao considerar cada par formado pelo recurso selecionado para o serviço sendo analisado e os recursos selecionados para os serviços adjacentes a ele. Esse ganho, por sua vez, é estabelecido como a razão entre a carga de comunicação (λ) entre os serviços e a distância física entre os recursos selecionados, ou somente como a carga quando a distância é zero. Para reduzir a complexidade, não foi incluído nessa equação o volume de dados trocados entre os serviços. Na equação, n indica a quantidade de serviços adjacentes a um determinado serviço a. O fator de comunicação é usado como um indicador do ganho ao selecionar um tipo concreto em detrimento de outro. Quanto maior o fator de comunicação, maior será a redução no atraso de comunicação.

$$\text{Fator de comunicação } (a) = \frac{\sum_{i=1}^{n} \text{Ganho na comunicação}\left(\text{tipo}a, \text{tipo}i\right)}{n}$$

$$\text{Ganho na comunicação}\left(\text{tipo}a, \text{tipo}i\right) = \begin{cases} \dfrac{\lambda}{\text{distância}\left(a,i\right)} & \text{se distância} > 0; \\ \lambda & \text{caso contrário.} \end{cases}$$

A Figura 30 ilustra o procedimento para selecionar recursos a serem consolidados. Ele consiste em avaliar os tipos concretos remanescentes para cada serviço após a filtragem e a classificação de recursos. Inicialmente, cada serviço é mapeado para o primeiro tipo concreto disponível. Os demais tipos são avaliados (a partir da linha 8 no algoritmo) de forma a verificar a possibilidade de consolidação ou redução no atraso de comunicação pela troca por outros tipos concretos.

Essa avaliação ocorre em ordem decrescente do valor de corte do vértice equivalente no grafo de dependências. O corte de um vértice é definido como a soma da carga de comunicação do serviço, representada como parte do peso em cada aresta incidente ao vértice em questão, quando o vértice adjacente também representa um serviço.

Entrada: $\mathcal{G}$: grafo de dependências; $f : \mathcal{S} \rightarrow \mathcal{V}$: tipos canônicos selecionados; $\mathcal{V}[]$: um vetor que representa os tipos concretos remanescentes, em que $\mathcal{V}[j], 1 \leq j \leq \Psi$, especifica os tipos concretos remanescentes no conjunto representado pelo tipo canônico j.
Saída: $f' : \mathcal{S}' \subseteq \mathcal{S} \rightarrow \mathcal{V}$: tipos concretos selecionados.
Dados: $\mathcal{S}$: conjunto de serviços em $\mathcal{G}$, t: número de tipos canônicos.

```
1  Marque todos os serviços s ∈ S como não mapeados
2  enquanto existir serviço não mapeado faça
3      maxP ← índice do vértice p com maior corte em G | p ∈ S e p ainda não foi mapeado
4      tiposCandidatos ← V[maxP]
5      melhorTipo ← null, melhorRecursoAdic ← 0, melhorFatorCom ← 0
6      minDemandaRec ← capacidade do tipo canônico menos robusto
7      capacParaMelhorar ← 0
8      enquanto tiposCandidatos não está vazia faça
9          tipoCorrente ← tiposCandidatos.first()
10         recursoAdic ← Recurso excedente de tipoCorrente
11         fatorCom ← Fator de comunicação de tipoCorrente
12         se recursoAdic ≥ minDemandaRec e (recursoAdic ≥ melhorRecursoAdic +
             capacParaMelhorar ou fatorCom > melhorFatorCom) então
13             busca_consolidação(G, f, V, maxP, tipoCorrente)
14             se há redução no atraso de comunicação então
15                 melhorTipo ← tipoCorrente, melhorRecursoAdic ← recursoAdic,
                     melhorFatorCom ← fatorCom
16                 se número de serviços consolidados = |S| então
17                     Remove todos os tipos em tiposCandidatos
18                 senão
19                     Remove tipoCorrente de tiposCandidatos
20                 fim
21             fim
22         senão
23             Remove tipoCorrente de tiposCandidatos
24         fim
25         se tipoCorrente é o tipo mais robusto no conjunto então
26             Remove todos os tipos em tiposCandidatos
27         fim
28     fim
29     f'( serviços envolvidos na consolidação) ← melhorTipo
30     para cada serviço s ∈ grupo de consolidação faça
31         Marque s como mapeado
32     fim
33 fim
34 retorna f'
```

Figura 30 – Consolidação de recursos
Fonte: Elaboração própria.

A condição, na linha 12, verifica a possibilidade de realizar a busca por consolidação, que só ocorre nos casos em que a capacidade excedente no tipo concreto corrente é suficiente para consolidar outros recursos. Para tal, ela deve ser igual ou superior à demanda do tipo canônico menos robusto. Além disso, essa capacidade deve ser superior à melhor capacidade excedente tida até então, acrescida da capacidade necessária para melhorar a melhor solução até então encontrada.

Outro fator que gera a possibilidade de busca por consolidação ocorre quando o tipo concreto analisado tem possibilidade de reduzir o atraso de comunicação, o que é determinado pelo fator de comunicação.

Apesar da quantidade de tipos concretos a serem avaliados ser potencialmente grande, a busca por consolidação não ocorre sempre, pois ela será realizada apenas nos casos em que realmente há possibilidade de haver algum ganho.

A busca por consolidação é implementada pelo Algoritmo 6.2, invocado na linha 13. Ele recebe como entrada o serviço e o tipo concreto atualmente sendo avaliados. Seu resultado é o conjunto de serviços para os quais a consolidação nesse tipo concreto ocorreu, ou um conjunto unitário contendo apenas o serviço fornecido como entrada, significando que não é possível haver consolidação. Como resultado secundário, esse algoritmo indica a capacidade adicional de recurso necessária para obter um resultado melhor na consolidação, o qual é usado nas buscas seguintes, conforme descrito anteriormente.

A condição, na linha 14, determina os casos em que há a troca do tipo concreto até então selecionado pelo atualmente em análise. Isso ocorre quando há redução no atraso de comunicação, ou seja, quando o número de serviços envolvidos na consolidação é aumentado e/ou quando o fator de comunicação do tipo em análise é maior que o do tipo até então selecionado.

A busca pela consolidação é interrompida para um determinado serviço caso os demais serviços sejam envolvidos na consolidação para um dos recursos considerados (linha 17), uma vez que não é possível obter resultado mais satisfatório.

Outra condição para interromper a busca para um determinado serviço é quando o tipo concreto mais robusto para este serviço foi avaliado (linha 26), novamente porque nenhuma solução envolvendo os outros tipos será mais satisfatória que a encontrada.

O resultado da consolidação é estabelecido como um mapeamento de um conjunto de serviços para o tipo concreto que possibilitou o melhor resultado na consolidação (linha 29). Esse conjunto contém necessariamente o serviço com o qual a busca foi iniciada, além dos demais serviços envolvidos na consolidação. Nos casos em que não é possível realizar consolidação, o conjunto conterá apenas o serviço com o qual a busca foi iniciada, sendo o tipo concreto estabelecido como aquele que obteve melhor fator de comunicação ao considerar os serviços adjacentes.

O fato de o mapeamento ser estabelecido como resultado de uma busca não indica que os serviços permanecerão mapeados para o tipo selecionado, uma vez que grupos já mapeados também são considerados em buscas seguintes. Dessa forma, quando for possível, o grupo como um todo poderá se envolver em uma consolidação para outro tipo de recurso. Este processo ocorrerá até que todos os serviços sejam mapeados.

Para realizar a busca por consolidação é definida uma nova estrutura de grafo auxiliar, criada a partir do grafo de dependências e dos resultados de buscas anteriores. Nessa estrutura, os vértices representam grupos de serviços envolvidos em uma consolidação, mapeados para um determinado recurso no algoritmo principal. Inicialmente, é criado um grupo para cada serviço, que contém apenas esse serviço mapeado para o primeiro tipo concreto disponível para ele. Esses grupos podem ser alterados após a busca por consolidação ser realizada. As arestas nessa estrutura representam a carga de comunicação entre os serviços que compõem cada grupo. A Figura 31 ilustra como essa estrutura é criada e atualizada. Por simplicidade, não é apresentado todo o modelo de recursos na figura.

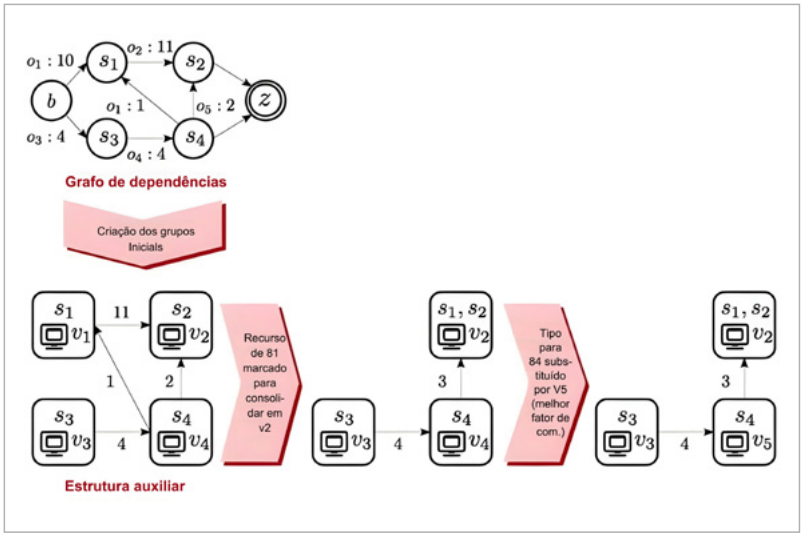
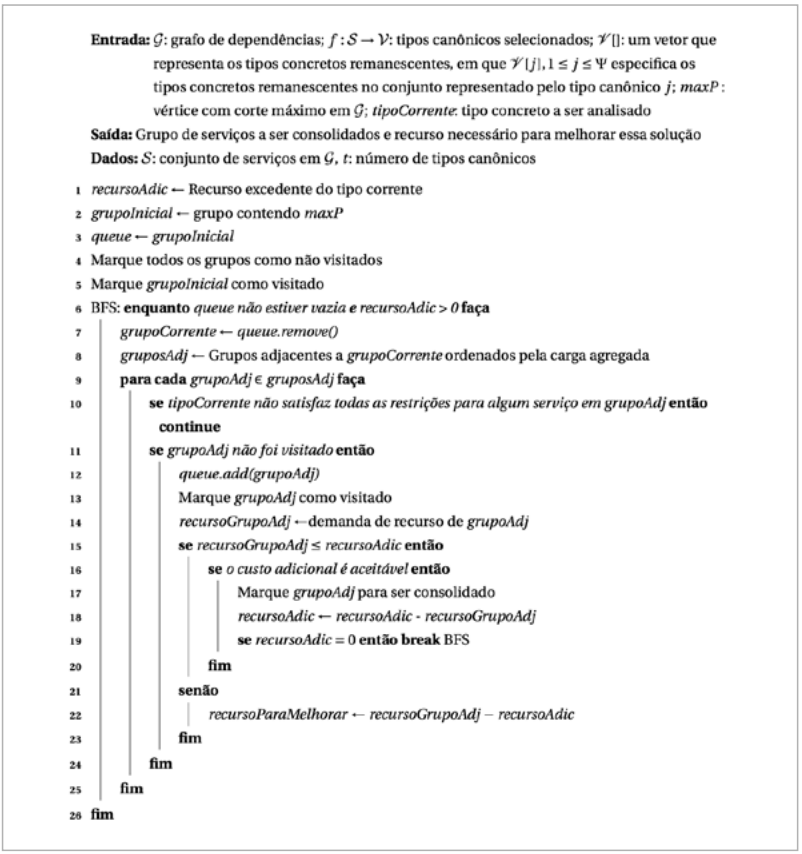


Figura 31 – Exemplo de criação e atualização da estrutura auxiliar utilizada na busca por consolidação
Fonte: Elaboração própria.

Inicialmente, é criado um grupo para cada um dos serviços (s_1, s_2, s_3, s_4) e o primeiro tipo concreto aceitável para estes serviços (v_1, v_2, v_3 e v_4, neste caso). Apenas as arestas que ligam pares de serviços são mantidas nessa estrutura. No exemplo, o recurso utilizado por s_1 foi marcado para consolidar com s_2. Dessa forma, o grupo original com s_1 deixa de existir, e esse serviço é adicionado ao grupo de s_2. A figura também ilustra outra transformação que pode ser realizada na estrutura, que ocorre quando não é possível haver consolidação, mas um novo tipo concreto é selecionado por oferecer melhor fator de comunicação. Nesse caso, o tipo concreto para o qual o serviço s_4 será mapeado, é trocado de v_4 para v_5.

O procedimento de busca por consolidação, descrito no algoritmo da Figura 32, baseia-se em uma busca em largura (*Breadth-First Search* – BFS) (Bundy; Wallen, 1984) no grafo de dependências. A busca é realizada enquanto houver grupos não visitados e recurso excedente (linha 6). A ordem de avaliação de grupos é estabelecida de acordo com a carga agregada dos serviços neles contidos, visando consolidar aqueles que têm maior demanda de comunicação.



Entrada: $\mathcal{G}$: grafo de dependências; $f : \mathcal{S} \rightarrow \mathcal{V}$: tipos canônicos selecionados; $\mathcal{V}[]$: um vetor que representa os tipos concretos remanescentes, em que $\mathcal{V}[j], 1 \leq j \leq \Psi$ especifica os tipos concretos remanescentes no conjunto representado pelo tipo canônico j; $maxP$: vértice com corte máximo em $\mathcal{G}$; *tipoCorrente*: tipo concreto a ser analisado
Saída: Grupo de serviços a ser consolidados e recurso necessário para melhorar essa solução
Dados: $\mathcal{S}$: conjunto de serviços em $\mathcal{G}$, t: número de tipos canônicos

```
1  recursoAdic ← Recurso excedente do tipo corrente
2  grupoInicial ← grupo contendo maxP
3  queue ← grupoInicial
4  Marque todos os grupos como não visitados
5  Marque grupoInicial como visitado
6  BFS: enquanto queue não estiver vazia e recursoAdic > 0 faça
7      grupoCorrente ← queue.remove()
8      gruposAdj ← Grupos adjacentes a grupoCorrente ordenados pela carga agregada
9      para cada grupoAdj ∈ gruposAdj faça
10         se tipoCorrente não satisfaz todas as restrições para algum serviço em grupoAdj então
             continue
11         se grupoAdj não foi visitado então
12             queue.add(grupoAdj)
13             Marque grupoAdj como visitado
14             recursoGrupoAdj ←demanda de recurso de grupoAdj
15             se recursoGrupoAdj ≤ recursoAdic então
16                 se o custo adicional é aceitável então
17                     Marque grupoAdj para ser consolidado
18                     recursoAdic ← recursoAdic - recursoGrupoAdj
19                     se recursoAdic = 0 então break BFS
20                 fim
21             senão
22                 recursoParaMelhorar ← recursoGrupoAdj − recursoAdic
23             fim
24         fim
25     fim
26 fim
```

Figura 32 – Busca por consolidação
Fonte: Elaboração própria.

Realiza-se uma consulta a cada grupo adjacente ao avaliado para ver a possibilidade de serviços desse grupo serem implantados no tipo concreto fornecido como parâmetro. Essa verificação é realizada na linha 10 por meio da aplicação das restrições em relação a atributos do recurso referentes aos serviços contidos no grupo. Caso algum serviço do grupo não possa ser implantado no tipo concreto em questão, o grupo é desconsiderado. Caso contrário (e se o grupo ainda não tiver sido visitado), a demanda de recurso desse grupo é somada usando o tipo canônico selecionado para cada serviço (linha 14). Essa demanda é comparada com a capacidade excedente no recurso usado na busca para checar se a consolidação é possível (linha 15).

Outra verificação realizada é se o custo adicional (caso ele exista) é aceitável (linha 16), o que é avaliado de acordo com a configuração de custo adicional admissível, estabelecida pelo usuário. Apenas se ambas as condições forem satisfeitas é que o grupo é marcado em uma consolidação. Nesse caso, a busca é interrompida se não há mais capacidade excedente (linha 19).

Nos casos em que não é possível consolidar porque a capacidade excedente não é suficiente, a capacidade necessária para melhorar a solução é atualizada (linha 22) como sendo a capacidade adicional que seria necessária para permitir a consolidação desse grupo, ou seja, a diferença entre a demanda de recurso para todos os serviços no grupo e a capacidade adicional nessa iteração.

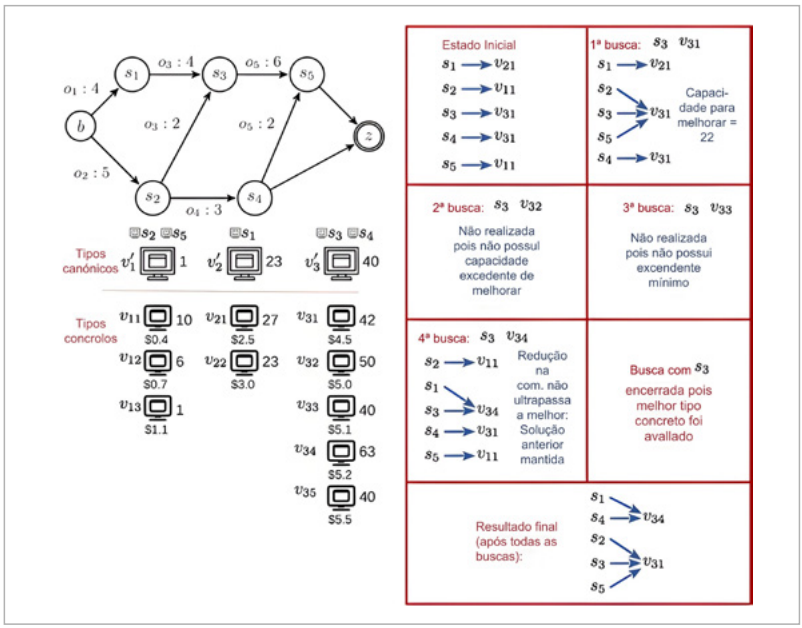


Figura 33 – Exemplo de execução do algoritmo de consolidação
Fonte: Elaboração própria.

A Figura 33 ilustra parte da execução do algoritmo de consolidação. À esquerda está a entrada, que consiste no grafo de dependências e o mapeamento de cada serviço para um tipo canônico, com os tipos concretos remanescentes após a aplicação das restrições

em relação a atributos do recurso. À esquerda de cada tipo, é apresentado seu identificador, ao passo que, à sua direita, é apresentado um valor numérico que indica sua capacidade. Para os tipos concretos, também é apresentado o custo.

Nesta entrada, por simplicidade, é assumido que o resultado da filtragem e da classificação dos tipos concretos é igual quando utilizado o mesmo tipo canônico. O quadro na parte direita apresenta parte do processo, considerando as buscas a partir do serviço s_3, que equivale ao vértice com maior corte, ou seja, o vértice que possui precedência na avaliação dos tipos concretos no algoritmo da Figura 30 (linha 3). A partir desse vértice, as buscas serão realizadas neste algoritmo considerando a ordem definida pela carga agregada. Por questão de espaço, a análise do fator de comunicação é desconsiderada nesse exemplo, e apenas a demanda expressa pela carga é usada.

O algoritmo inicia com cada serviço sendo mapeado para o primeiro tipo concreto disponível. No exemplo dado, a primeira chamada ao procedimento de busca é realizada com o tipo v_{31}. O primeiro grupo adjacente a ser avaliado é aquele que contém s_5 devido à demanda maior de comunicação. O recurso para o qual esse serviço foi mapeado é então marcado para consolidar, pois sua demanda de recursos não ultrapassa a capacidade adicional de v_{31} e não há custo adicional.

Como ainda há recurso adicional, a busca avalia o próximo serviço s_1, mas o recurso não pode ser consolidado, uma vez que sua demanda é maior que a capacidade remanescente: sua demanda é 23 e a capacidade remanescente é 1. Nesse ponto, a capacidade necessária para melhorar a solução é estabelecida como 22, por ser a capacidade adicional necessária para permitir a consolidação do recurso para o qual s_1 foi mapeado. Em seguida, s_2 é avaliado, sendo seu recurso também marcado para consolidação. Como não há mais capacidade excedente após a avaliação de s_2, a busca é encerrada com os recursos de s_5 e s_2 marcados para consolidar no recurso de s_3, e s_1 e s_4 mantidos nos recursos originais.

O próximo tipo, v_{32}, não é avaliado pois não possui capacidade excedente suficiente para melhorar a solução previamente encontrada. Para ser possível melhorar a solução, a capacidade deveria ser maior ou igual à soma da capacidade excedente do melhor tipo até então encontrado ($v_{31} = 2$) com a capacidade não disponível na busca anterior (22), ou seja, deveria ser no mínimo 24 (mas é somente 10). O mesmo acontece para o próximo tipo v_{33}, mas, nesse caso, ele não é avaliado porque não possui nem mesmo a capacidade excedente mínima para permitir que o recurso seja consolidado, que é igual à demanda do tipo canônico menos robusto (1 em nosso exemplo).

Continuando o procedimento, a busca usando o tipo v_{34} também é realizada de acordo com a ordem da demanda de comunicação. Ao avaliar o serviço s_5, seu recurso não é marcado para consolidar porque o custo adicional não é aceitável, supondo, nesse caso, que não há tolerância quanto ao custo adicional. Isso acontece porque a soma do custo dos recursos, inicialmente utilizados por s_3 (US$4.5) e s_5 (US$0.4), é menor que o custo de v_{34} (US$5.2), não justificando a troca, pois o custo será menor ao alocar os recursos sem a consolidação. Em virtude disso, a busca prossegue com s_1 e finaliza com o recurso desse serviço sendo consolidado pois não há mais capacidade excedente.

O resultado dessa busca é comparado com o melhor resultado até então obtido (que era usando v_{31}). Como a quantidade de serviços cujos recursos foram consolidados, e consequentemente a redução na demanda de comunicação anteriormente obtida é maior, o resultado anterior é mantido. Nesse momento, as buscas partindo do serviço s_3 são encerradas, pois o tipo de recurso concreto mais robusto foi avaliado. Somente nesse instante os grupos envolvendo serviços para os quais houve mudança no mapeamento são atualizados na estrutura auxiliar, sendo os serviços s_3, s_5 e s_2 marcados como mapeados. O procedimento se repete novamente considerando os serviços que ainda não foram mapeados, s_1 e s_4. O resultado final da consolidação é apresentado no final do quadro,

no qual, apesar de não terem comunicação entre si, o recurso de s_1 foi marcado para consolidar no recurso de s_4 devido à capacidade excedente, diminuindo o gasto com os recursos selecionados.

O exemplo apresentado ilustra os dois tipos de redução de custos proporcionados pela técnica de consolidação: redução no atraso de comunicação e no custo dos recursos alocados.

Casos de insucesso da síntese de recursos

A síntese de recursos está sujeita a alguns casos em que não é possível obter os resultados das atividades de estimativa e seleção de recursos. Como a estimativa é realizada considerando apenas as restrições de QoS sem, portanto, avaliar as restrições em relação a atributos do recurso, pode ocorrer que nenhum tipo concreto representado pelo tipo canônico satisfaça todas as restrições impostas sobre o serviço em questão. Com isso, ocorrerá insucesso na seleção pois não será possível implantar o serviço em um recurso com capacidade estabelecida pelo tipo canônico selecionado. Esse problema ocorre principalmente porque não é possível garantir que os tipos concretos representados por um determinado tipo canônico possuam alguma propriedade além daquela relacionada a sua capacidade. Ou seja, não é possível garantir, por exemplo, que haverá tipos concretos que são disponibilizados por um determinado provedor, ou que haverá alguns que podem ser instanciados em uma certa localidade.

Para realizar a seleção de recursos de forma satisfatória, a estimativa de recursos deve desconsiderar os tipos canônicos para os quais não haverá na seleção ao menos um tipo concreto que satisfaça todas as restrições. Uma forma de garantir que isso ocorra é avaliar essas restrições antes de realizar a estimativa de recursos, como forma de desconsiderar para um serviço os tipos canônicos para os quais todos os tipos concretos foram eliminados. Ao fazer

isso, a seleção de recursos passaria a ser imediata, pois o primeiro tipo concreto disponível no conjunto seria considerado factível.

Contudo, para implementar essa funcionalidade, é preciso avaliar as restrições eliminatórias para todos os pares formados por serviços e tipos concretos disponíveis, o que não é uma solução eficiente, dado que a complexidade dessa tarefa é $O\left(\rho^e nt\right)$, em que ρ^e é o número de restrições eliminatórias, n é o número de serviços e t é o número de tipos concretos de recurso. Apesar de ser uma tarefa com complexidade polinomial, o número de tipos concretos pode ser alto se considerarmos múltiplos provedores. Por exemplo, considerando os provedores utilizados para demonstrar a geração dos tipos canônicos, foram utilizados 827 tipos concretos.

Em virtude disso, é proposta outra abordagem mais simples, que consiste em detectar insucesso na seleção de recursos e refazer a etapa de estimativa, desconsiderando os tipos canônicos para os serviços em que o insucesso foi detectado. Apesar de parecer ineficiente, essa abordagem obtém um índice menor de tempo de processamento que a proposta anterior, porque a estimativa se baseia nos tipos canônicos, cuja quantidade é necessariamente ordens de magnitude menor que a quantidade de tipos concretos. Dessa forma, mesmo que a estimativa tenha que ser refeita (o que não acontecerá com tanta frequência) o resultado será obtido em menos tempo do que usando a solução anterior.

Outro cenário de insucesso possível na síntese é não haver tipos canônicos com capacidade suficiente para satisfazer todas as restrições de QoS. Isso ocorre quando a carga e/ou a demanda de recursos para processar determinadas operações é muito elevada. Nesses casos, mesmo selecionando o tipo canônico mais robusto para todos os serviços, não é possível satisfazer as restrições impostas. Isso acontece porque todos os tipos concretos representados pelo tipo canônico mais robusto têm exatamente a capacidade de

hardware expressa nele, o que significa que não há disponível nenhum tipo de capacidade superior.

Para lidar com esse tipo de problema, é proposta a adição de instâncias dos serviços. Para tal, assume-se que há um balanceador de carga com sobrecarga negligenciável, que escalona as requisições recebidas de maneira igual entre as instâncias de um determinado serviço. Uma vez que essas instâncias não são entidades de primeira classe na modelagem dos serviços e visando reduzir o tempo de processamento da síntese, as instâncias adicionais não são explicitamente representadas no grafo de dependências. Em vez disso, apenas a carga de entrada é alterada como forma de simular a inclusão de novas instâncias.

A Figura 34 mostra como o uso de novas instâncias é considerado no modelo, em que se é o serviço sobrecarregado para o qual uma nova instância foi adicionada. Apesar das novas instâncias existirem logicamente, como mostrado no esquema apresentado no lado superior direito da figura, no grafo de dependências apenas a carga sobre o serviço que teve novas instâncias criadas é alterada. Isso faz com que a síntese de recursos seja realizada de forma mais eficiente, pois não é preciso estimar e selecionar recursos para cada instância separadamente. Essas tarefas são realizadas apenas uma vez e o resultado é reproduzido para cada instância do serviço. A informação sobre a quantidade de instâncias é mantida no descritor do serviço.

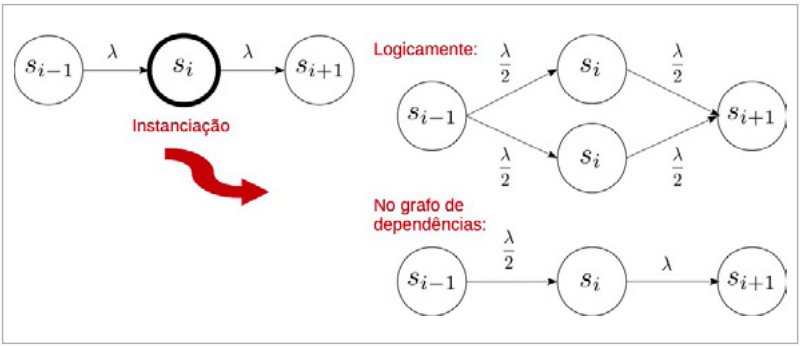


Figura 34 – Forma de inclusão de novas instâncias na modelagem de serviços
Fonte: Elaboração própria.

A estratégia usada para identificar os serviços que devem ter instâncias adicionadas será discutida no próximo capítulo, em que serão apresentados detalhes sobre a implementação das estratégias aqui discutidas.

A síntese de recursos consiste em determinar a capacidade de recurso necessária para implantar um conjunto de serviços coreografados e selecionar o tipo de recurso mais apropriado para oferecer essa capacidade. Nesse capítulo, foi descrita uma estratégia para solucionar este problema.

A Figura 35 apresenta as etapas da estratégia de síntese de recursos. Para cada etapa, são apresentadas as entradas (acima) e as saídas obtidas (abaixo).

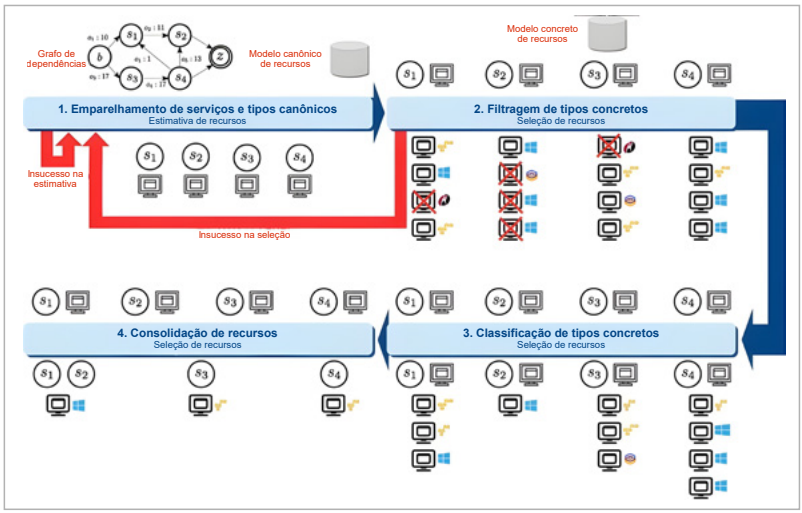


Figura 35 – Etapas da síntese de recursos
Fonte: Elaboração própria.

A primeira etapa realizada é a estimativa de recursos, que consiste em determinar, com base na representação do grafo de dependências e no modelo canônico de recursos, a capacidade de recurso necessária para satisfazer as restrições de QoS. É proposto que essa atividade seja realizada por meio do emparelhamento dos serviços

com tipos canônicos, usando uma estratégia baseada no problema MMKP. Nessa etapa, também é avaliado se instâncias únicas dos serviços serão suficientes para atender a demanda e, caso não seja (o que é caracterizado como insucesso), novas instâncias são criadas para os serviços. A saída da estimativa de recursos é o mapeamento de cada serviço (com possíveis instâncias adicionais) para um tipo canônico.

A próxima etapa da síntese consiste na primeira tarefa da seleção de recursos, que é responsável por filtrar os tipos concretos disponíveis para cada tipo canônico. Essa filtragem é realizada de acordo com as restrições eliminatórias estabelecidas sobre os serviços. De maneira semelhante, as restrições classificatórias são avaliadas para classificar esses tipos na etapa seguinte. A saída de ambas as etapas é o mapeamento de cada serviço para um conjunto de tipos concretos ou insucesso na síntese por não haver satisfação a todas as restrições eliminatórias. No caso de insucesso, a estimativa de recursos é refeita ignorando o tipo canônico que causou o insucesso para o serviço.

Após a avaliação das restrições em relação a atributos do recurso, a última etapa da síntese consiste em verificar a possibilidade de realizar a consolidação de recursos e avaliar a redução no atraso de comunicação proporcionada pela troca de tipos concretos. A saída dessa etapa, e consequentemente da síntese de recursos, é o mapeamento de um serviço (ou conjunto de serviços se houve consolidação) para o tipo concreto que será usado na implantação.

A apresentação da síntese de recursos consiste na principal contribuição deste livro. Ela permite que um conjunto de coreografias seja implantado de maneira eficiente, satisfazendo um conjunto de restrições não funcionais e levando em consideração um vasto conjunto de tipos de recursos. Para agregar as técnicas apresentadas neste capítulo (e nos capítulos anteriores) é proposta uma arquitetura para implantação de coreografias, que será descrita no próximo capítulo.

6

Arquitetura e implementação

Anteriormente, foram discutidos os problemas tratados neste livro e apresentadas as técnicas propostas para resolvê-los. Mais precisamente, foram propostas estratégias para a representação de coreografias de serviços e restrições não funcionais associadas e para a modelagem de recursos disponibilizados em um ambiente com múltiplos provedores de nuvem. Com base nessa representação, foram indicadas abordagens para a estimativa e a seleção de recursos, visando a implantação de um conjunto de coreografias, com provável compartilhamento de serviços entre elas. Contudo, para que os objetivos estabelecidos nesta pesquisa sejam de fato alcançados, é preciso que o uso das abordagens propostas seja realizado de maneira conjunta e coordenada. Diante disso, é apresentado neste capítulo uma arquitetura proposta com esse objetivo.

A arquitetura considera o gerenciamento de recursos tanto em tempo de implantação quanto em tempo de execução dos serviços que compõem as coreografias. Dessa forma, ela pode ser usada para estimar e selecionar os recursos que serão usados na implantação dos serviços e para garantir que as restrições não funcionais sejam satisfeitas durante a encenação das coreografias. Para ilustrar o uso da arquitetura, são discutidos os cenários em que ela pode ser aplicada.

Neste capítulo, também se discute a implementação de um protótipo de parte da arquitetura proposta que foi implementado para demonstrar e avaliar a abordagem.

Arquitetura

É proposta uma arquitetura que fornece abstração no gerenciamento de recursos para múltiplas coreografias de serviços. Esta se baseia no processamento de modelos por camadas de funcionalidades que transformam os elementos do modelo até que comandos sejam gerados sobre a infraestrutura subjacente. Dessa forma, como ilustrado na Figura 36, a arquitetura proposta é estruturada em três camadas: síntese de coreografias, síntese de recursos e *broker* de recursos.

A primeira camada da arquitetura – *síntese de coreografias* – constitui o ponto de interação com o usuário, por intermédio do componente *interface do usuário*. A partir dele, a especificação da(s) coreografia(s) de serviços, com restrições não funcionais associadas, é submetida pelo usuário. Nossa abordagem é melhor aplicada se as coreografias forem submetidas em lote, já que o compartilhamento de serviços pode ser melhor analisado nesse caso. No entanto, a implantação de coreografias também pode ser realizada com a submissão de cada uma de forma isolada, mas possivelmente exigindo adaptação em serviços previamente implantados.

É proposto o uso de adaptadores para a representação interna. Dessa forma, desde que o adaptador equivalente esteja implementado como parte da arquitetura (componente *adaptador para grafo de processo*), a especificação das coreografias pode ser realizada em qualquer linguagem usada para este propósito (como BPMN 2.0). Dado que a seleção de serviços não faz parte do escopo considerado neste trabalho, assume-se que a modelagem das coreografias é realizada indicando implementações concretas dos serviços.

Além de interagir com o usuário, a camada de síntese de coreografias também é responsável por gerar a estrutura do grafo de dependências, que consiste na representação conjunta das coreografias e das restrições não funcionais associadas e submetidas como entrada. Essa tarefa é realizada pelo componente *gerador de grafo de dependências*. Sua estrutura é usada na próxima camada para guiar as atividades de estimativa e seleção de recursos.

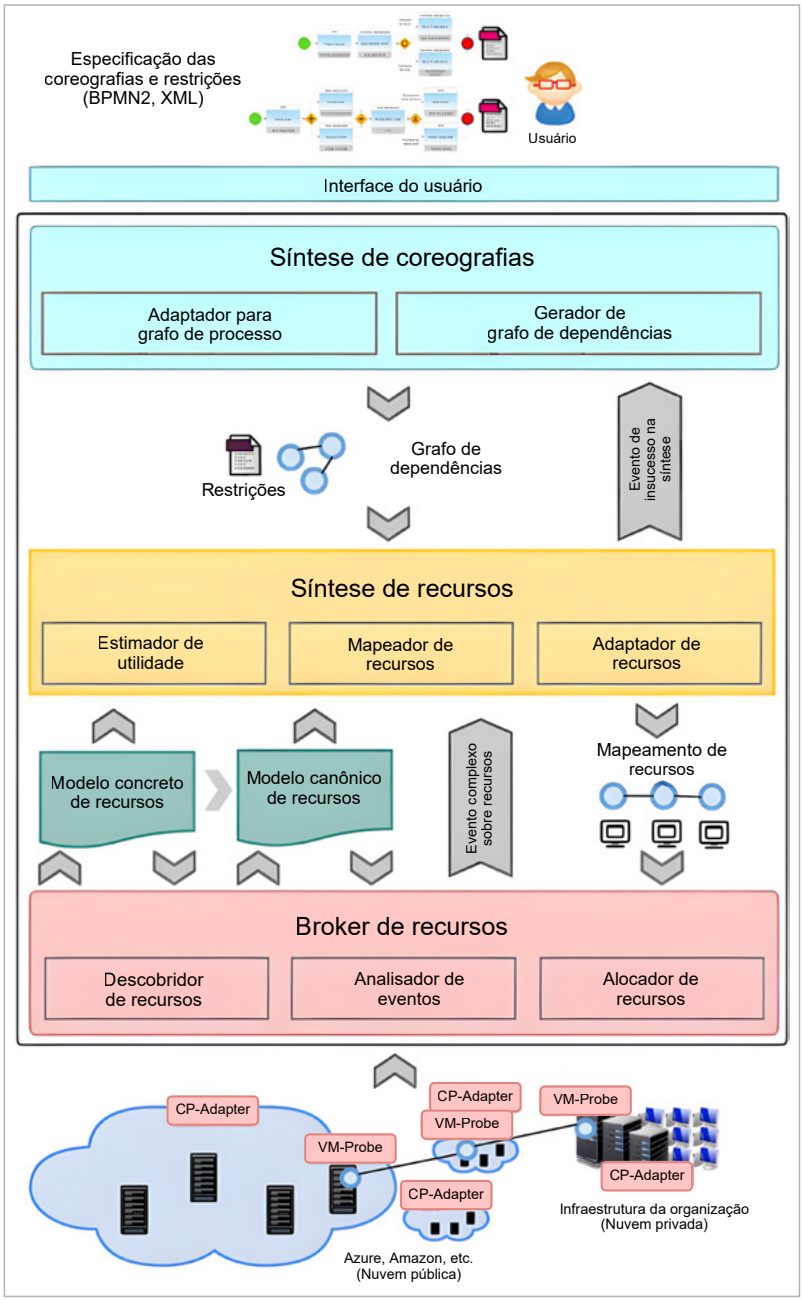


Figura 36 – Arquitetura geral proposta
Fonte: Elaboração própria.

A *síntese de recursos* é a principal camada da arquitetura. Ela é responsável por estimar e selecionar recursos usando as abordagens discutidas no capítulo anterior. Para isso, além do grafo de dependências, ela utiliza os modelos de recurso (concreto e canônico). O principal componente na síntese é o *mapeador de recursos*, que implementa tanto a estimativa quanto a seleção de recursos. A primeira tarefa é realizada usando as estimativas de contribuição de QoS de acordo com os tipos canônicos.

A estimativa da função de utilidade associada às métricas de QoS é abstraída no componente *estimador de utilidade*. É assumido que esta tarefa pode ser implementada independentemente do mapeamento de recursos. Dessa forma, o problema da estimativa de QoS é isolado e o trabalho é focado nas especificidades do mapeamento de recursos.

A saída da síntese de recursos é o mapeamento de conjuntos de serviços coreografados para o tipo concreto selecionado para cada conjunto. O tamanho de cada conjunto dependerá da possibilidade de realizar consolidação de recursos. Nos casos em que isso não é possível, os conjuntos corresponderão a conjuntos unitários contendo serviços isolados.

Nos casos em que não é possível satisfazer todas as restrições não funcionais com os recursos disponíveis, um evento de insucesso é gerado para a camada superior que notifica o usuário. Nesse caso, ele deve selecionar outra implementação de serviço ou relaxar as restrições especificadas para o serviço ou coreografia para a qual ocorreu insucesso.

Outra tarefa executada na camada de síntese de recursos, por meio do componente *adaptador de recursos*, é a adaptação da alocação de recursos frente às mudanças em tempo de execução, como aumento ou diminuição da carga de trabalho.

A implantação dos serviços é realizada pela camada inferior – *broker de recursos* –, pelo componente *alocador de recursos*, usando como entrada o modelo de recursos gerado pela síntese. Essa camada gerencia a infraestrutura privada e interage com provedores de nuvem pública por meio do envio de comandos por adaptadores.

Outra tarefa implementada nesta camada é a geração dos modelos concreto e canônico de recursos. Para tal, o componente *descobridor de recursos* obtém a capacidade disponível em máquinas físicas, no caso

de nuvens privadas, e os tipos de VM disponibilizados por provedores de nuvem pública. O gerenciamento de recursos em nuvens públicas é realizado agregando todos os provedores em um mesmo canal por meio de um *endpoint* gerenciável (que interage com adaptadores para cada provedor – *CP-Adapter*). Somente são considerados provedores previamente registrados nesta camada e para os quais há um adaptador.

Apesar de não ter sido implementado no protótipo desenvolvido, a arquitetura prevê que durante a encenação das coreografias, o componente *VM-Probe* execute, em cada VM alocada, o monitoramento contínuo para detectar possíveis violações da satisfação nas restrições não funcionais especificadas. Dessa forma, quando ocorresse mudanças, como aumento no número de requisições, o modelo de recursos seria verificado pelo componente *analisador de eventos* para avaliar se a satisfação de restrições ainda é possível, dado o novo cenário. É proposto que esta verificação se baseie no uso de Processamento de Eventos Complexos (*Complex Event Processing* – CEP) (Luckham; Frasca, 1998), uma técnica para monitorar e executar tarefas reativas em sistemas distribuídos, por intermédio da análise de correlação entre eventos. A partir dos dados monitorados, seria realizada uma análise lógico-temporal para correlacionar eventos capturados, gerando eventos complexos que seriam usados para inferir causas e possíveis soluções frente a violações das restrições. A implementação desses mecanismos é tida como trabalho futuro.

Caso as mudanças verificadas sejam admissíveis com a atual configuração de recursos, a encenação da coreografia não é interrompida. Senão, quando for difícil satisfazer todas as restrições para o novo cenário (por exemplo, quando os recursos selecionados são insuficientes), uma adaptação é acionada com o envio de um evento complexo sobre recursos. Com base neste evento, o componente *adaptador de recursos* decide qual a melhor estratégia para solucionar as violações detectadas, o que provavelmente resultará na execução de uma nova síntese, com o redimensionamento ou aumento/diminuição no número de VMs previamente alocadas; ou, alternativamente, no envio de um evento de insucesso na síntese,

para os casos em que não é possível reparar as violações (por exemplo, quando não há recursos que satisfaçam todas as restrições).

Cenário de uso da arquitetura

Conforme ilustrado na Figura 37, a arquitetura proposta pode ser classificada como uma solução de plataforma como serviço (PaaS) para o gerenciamento de recursos, que interage com provedores na categoria de infraestrutura como serviço (IaaS) para permitir a implantação de composições gerenciadas por provedores de software como serviço (SaaS).

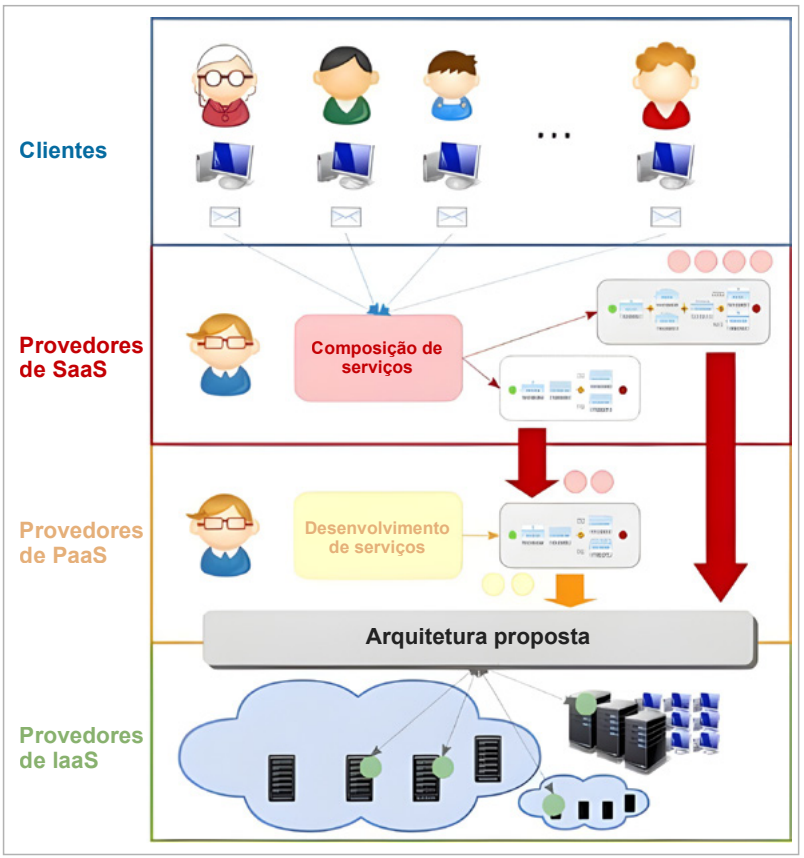


Figura 37 – Cenário de uso da arquitetura
Fonte: Elaboração própria.

Adicionalmente, os provedores de serviço podem desenvolver suas soluções usando funcionalidades oferecidas por provedores de PaaS que, dessa forma, podem utilizar a arquitetura proposta para implantar os serviços. Porém, é importante destacar que a classificação da arquitetura como uma solução de PaaS não inclui todos os elementos tipicamente encontrados em soluções dessa categoria (Lawton, 2008). As facilidades oferecidas pela arquitetura como plataforma são limitadas ao gerenciamento de recursos, não tendo relação com as outras atividades refrentes ao desenvolvimento das aplicações, como codificação e teste.

Nessa figura, as três camadas (SaaS, PaaS e IaaS) podem constituir um provedor único ou cada uma delas representar um provedor diferente. Os provedores de SaaS desenvolvem aplicações direcionadas a um conjunto de clientes. Embora esse desenvolvimento envolva, também, a criação de aplicações usando serviços isolados, são considerados os casos em que as aplicações constituem composições de serviços. Nos casos em que há separação entre as categorias de provedores, o desenvolvimento das composições de serviços geralmente é realizado tendo como base facilidades oferecidas por provedores de PaaS, que proveem não somente o ambiente de desenvolvimento do software, mas também um conjunto de serviços previamente desenvolvidos e que podem ser integrados à composição. A composição de serviços precisa ser implantada em um conjunto de VMs na camada de IaaS antes que elas possam atender as requisições dos clientes.

Para provedores de SaaS que interagem diretamente com provedores de IaaS, a implantação dos serviços deve ser realizada por eles mesmos. Por outro lado, se provedores de SaaS desenvolvem suas aplicações usando plataformas de provedores de PaaS, este é o responsável pela implantação dos serviços. Em ambos os casos, a arquitetura proposta pode ser utilizada para facilitar as atividades relacionadas ao gerenciamento de recursos. Dessa forma,

ela constitui uma interface entre a camada de IaaS e as demais camadas, podendo o usuário, neste cenário, ser tanto provedor de SaaS quanto de PaaS. Nas duas situações, as restrições não funcionais estabelecidas são guiadas pelas necessidades dos clientes que utilizam os serviços, havendo a preocupação de que essas restrições sejam atendidas de forma a reduzir o custo associado à utilização dos recursos.

Apesar das atividades automatizadas pela arquitetura terem como alvo a experiência oferecida aos clientes, eles não interagem diretamente com ela. Apenas o usuário (em nível SaaS ou PaaS) faz submissões a ela e tem influência sobre as decisões tomadas durante sua utilização.

Da maneira que foi proposta, a arquitetura pode ser implementada como um sistema dedicado, que é utilizado em contexto intra-institucional, ou como uma plataforma compartilhada, que é utilizada, por exemplo, por meio de uma API de acesso remoto.

Implementação da arquitetura

Para demonstrar e validar as abordagens propostas, foi implementado um protótipo de parte da arquitetura apresentada. Esta foi realizada usando as linguagens Java 8 e XML, além de um conjunto de arcabouços e bibliotecas auxiliares. Foram empregados diversos princípios de engenharia de software, como padrões de projeto, para tornar o código modular e de fácil extensão, visando sua reutilização em trabalhos futuros.

Implementação da síntese de coreografias

Como primeiro elemento na camada de síntese de coreografias, foi implementado um mecanismo para a representação dos grafos de processo e das restrições não funcionais.

Na implementação foi considerado que dados sobre os serviços estão disponíveis em um catálogo implementado usando o padrão *Registry* (Fowler, 2002). Com isso, usando um identificador do serviço é possível obter os dados disponíveis para este serviço. De maneira semelhante, foi considerado que existe um catálogo para cada um dos modelos de recurso gerados, cujo conteúdo também é acessado usando um identificador.

A especificação de restrições de QoS é realizada por meio da implementação das classes abstratas apresentadas no diagrama de classes da Figura 38. Consequentemente, para incluir uma restrição de QoS associada a uma coreografia de serviços, o usuário deve especificar uma implementação a ser usada para cada uma dessas classes.

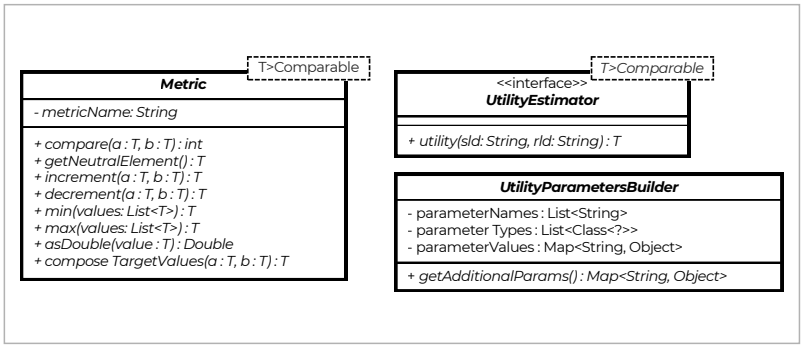


Figura 38 – Classes que definem uma restrição de QoS no arcabouço proposto
Fonte: Elaboração própria.

Como pode ser visto, a representação de métricas de QoS (definida pela classe Metric) segue a definição proposta anteriormente. Os valores da função de utilidade são obtidos pela implementação da classe UtilityEstimator, usando como parâmetros os identificadores do serviço e do recurso, assim como uma possível adição de parâmetros providos pelo construtor de atributos da métrica equivalente, definido pela implementação da classe UtilityParametersBuilder.

Como o objetivo da implementação desse protótipo era demonstrar a abordagem proposta, foi assumido que a especificação da coreografia é dada diretamente na notação do grafo de processo, ou seja, não foi implementado um adaptador que traduz coreografias especificadas em BMPN 2.0 (ou outra linguagem). Dessa forma, a implementação do componente Adaptador para grafo de processo foi estabelecida como trabalho futuro.

Visando permitir a submissão de coreografias usando protocolos padrão da Web, a especificação de cada grafo de processo é realizada usando um arquivo XML. Apesar dessa decisão, por simplicidade, a interface do usuário foi disponibilizada na forma de uma biblioteca na linguagem Java. A implementação de outras alternativas mais adequadas ao cenário de uso da arquitetura, como o oferecimento de suas funcionalidades por meio de uma API REST (Fielding, 2000), é tida, também, como trabalho futuro.

Na especificação de restrições de QoS, além de indicar a implementação das classes abstratas que permitem a estimativa dos valores de utilidade da métrica de QoS, o usuário deve especificar as operações que são restringidas por ela (ou omitir essa informação nos casos em que a restrição se aplica a toda a coreografia), o operador relacional considerado e o valor-alvo da restrição.

Para distinguir as restrições eliminatórias das classificatórias que devem ser aplicadas a todos os serviços que compõem a coreografia, é usada uma *tag* XML própria. Quando a restrição diz respeito a serviços específicos, o usuário deve especificar quais são esses serviços. A descrição dos demais componentes dessas restrições é realizada a partir da linguagem integrante do arcabouço.

Para ilustrar a representação proposta, a Figura 39 apresenta o arquivo XML de especificação da coreografia de rastreamento de frete e as restrições não funcionais associadas. Por simplicidade, é apresentado apenas a representação dos serviços Gerenciador de Inventário (GEREN_INVENT) e Departamento de Vendas (DEP_VENDAS),

uma vez que a especificação dos demais é semelhante. As anotações incluídas indicam os componentes principais da representação.

```
<?xml version="1.0" encoding="UTF-8"?>
<processGraph>
    <!-- Vértices do grafo -->
    <vertices>
        <!-- Vértice inicial -->
        <vertex id="START" type="START" />
        <vertex id="GEREN_INVENT" type="SERVICE">
            <!-- Descrição de um serviço -->
            <serviceDescriptor>
                <name>GEREN_INVENT</name>
                <dependencies>
                    <dependency
serviceSpecRole="consultarPedPen" />
                    <dependency
serviceSpecRole="consultarFrete" />
                    <dependency
serviceSpecRole="consultarFrete" />
                </dependencies>
                <logicalOperations>
                    <!-- Descrição de uma operação -->
                    <logicalOperation>
                        <name>rastrearFrete</name>
                        <numberOfInstructions>3</
numberOfInstructions>
                        <uri>/rastrearFrete/{id}</uri>
                        <roles>rastrearFrete</roles>
                    </logicalOperation>
                </logicalOperations>
                <serviceType>SOAP</serviceType>
                <roles>rastrearFrete</roles>
                <category>NEW_SERVICE</category>
                <packageURI>http://www.inf.ufg.br/~raphael/
services/inventManager.jar</packageURI>
                <packageType>COMMAND_LINE</packageType>
                <endpointName>inventManager</endpointName>
<port>1234</port>
                <version>0.1</version>
            </serviceDescriptor>
        </vertex>
        <vertex id="DEP_VENDAS" type="SERVICE">
            <serviceDescriptor>
                <name>DEP_VENDAS</name>
                <logicalOperations>
```

```
                    <logicalOperation>
                        <name>consultarPedPen</name>
                        <numberOfInstructions>29</
numberOfInstructions>
                        <uri>/consultarPedPen/{id}</uri>
                        <roles>consultarPedPen</roles>
                    </logicalOperation>
                </logicalOperations>
                <serviceType>SOAP</serviceType>
                <roles>consultarPedPen</roles>
                <category>NEW_SERVICE</category>
                <packageURI>http://www.inf.ufg.br/~raphael/
services/salesDep.jar</packageURI>
                <packageType>COMMAND_LINE</packageType>
                <endpointName>salesDep</endpointName>
                <port>1234</port>
                <version>0.1</version>
            </serviceDescriptor>
        </vertex>
...
        <!-- Conectores entre serviços -->
        <vertex id=»ORS1» type=»ORS» />
        <vertex id=»ORJ1» type=»ORJ» />
        <!-- Vértice final -->
        <vertex id=»END» type=»END» />
    </vertices>
    <!-- Arestas do grafo -->
    <edges>
        <!-- Requisição a uma operação -->
        <edge vertexFrom="START" vertexTo="GEREN_INVENT"
op="rastrearFrete" />
        <edge vertexFrom="GEREN_INVENT" vertexTo="DEP_
VENDAS" op="consultarPedPen" />
        <edge vertexFrom="GEREN_INVENT" vertexTo="ORS1" />
        <!-- Requisição a uma operação com probabilidade
-->
        <edge vertexFrom="ORS1" vertexTo="FRETE_TERR"
op="consultarFrete">
            <p>0.5</p>
            <pConj id="12">0.4</pConj>
        </edge>
        <edge vertexFrom="ORS1" vertexTo="FRETE_MAR"
op="consultarFrete">
            <p>0.1</p>
            <pConj id="12">0.4</pConj>
        </edge>
        <edge vertexFrom="FRETE_TERR" vertexTo="ORJ1" />
        <edge vertexFrom="FRETE_MAR" vertexTo="ORJ1" />
        <edge vertexFrom="ORJ1" vertexTo="END" />
```

```
    </edges>
    <!-- Carga esperada sobre a coreografia -->
    <load>80</load>
    <constraints>
        <!-- Especificação de restrições de QoS -->
        <qosConstraints>
            <qosConstraint>
                <targetOps>
                    <targetOp service="GEREN_INVENT"
op="rastrearFrete" />
                    <targetOp service="DEP_VENDAS"
op="consultarPedPen" />
                    <targetOp service="FRETE_TERR"
op="consultarFrete" />
                    <targetOp service="FRETE_MAR"
op="consultarFrete" />
                </targetOps>
                <metricClass>br.ufg.inf.utility.
ThroughputMetric</metricClass>
                <utilityEstimator>br.ufg.inf.utility.
ThroughputEstimator</utilityEstimator>
                <utilityParamBuilder>br.ufg.inf.utility.
ThroughputParamBuilder</utilityParamBuilder>
                <relationalOp>>=</relationalOp>
                <targetValue>10.0</targetValue>
            </qosConstraint>
        </qosConstraints>
        <!-- Especificação de restrições em relação a
atributos do recurso -->
        <resourceConstraints>
            <generalConstraintsDescriptor>
                 (MINIMUM_RESOURCE_USE = 0) & (ASCENDENT
COST) & ( DESCENDENT CLOUD_PROVIDER_REPUTATION)
            </generalConstraintsDescriptor>
            <!-- Apenas 12 horas de indisponibilidade por
ano -->
            <serviceConstraintDescriptor service="GEREN_
INVENT">
                 (AVAILABILITY > 99.9987443)
            </serviceConstraintDescriptor>
        </resourceConstraints>
    </constraints>
</processGraph>
```

Figura 39 – Especificação do grafo de processo e das restrições não funcionais referentes à coreografia para rastreamento de frete
Fonte: Elaboração própria.

Como pode ser visto, a representação proposta inclui atributos relacionados à descrição da coreografia (como papéis e conectores) e aos serviços que a compõem.

A topologia da coreografia é especificada por meio dos vértices e das arestas do grafo. Cada vértice é definido por um tipo que estabelece os atributos que obrigatoriamente devem ser especificados. Os tipos possíveis são:

- START: define o vértice inicial do grafo;
- END: define um vértice final do grafo;
- SERVICE: define um vértice que representa um serviço da coreografia;
- ANDS: define um vértice que representa um conector *fork* de conjunção;
- ANDJ: define um vértice que representa um conector *join* de conjunção;
- ORS: define um vértice que representa um conector *fork* de disjunção;
- ORJ: define um vértice que representa um conector *join* de disjunção;
- XORS: define um vértice que representa um conector *fork* de disjunção mutuamente exclusiva; e
- XORJ: define um vértice que representa um conector *join* de disjunção mutuamente exclusiva.

Para cada serviço, há uma descrição dos papéis que ele implementa assim como dos papéis dos quais ele depende. Esses papéis devem coincidir com operações providas e requisitadas pelos serviços que são conectados por arestas no grafo de processo. Por sua vez, para cada operação são descritas informações que podem ser usadas para estimar a demanda de recursos para o processamento

dessa operação. Conforme será discutido adiante, para as métricas de QoS implementadas nesse protótipo é suficiente restringir essas informações ao número médio de instruções usadas para processar cada requisição da operação.

Além dos atributos que são usados na estimativa e seleção de recursos, a representação dos serviços também considera outros que são essenciais para a sua implantação. São eles:

- URI das operações;
- tipo de serviço: SOAP ou REST. Este atributo define o processo de invocação do serviço, que é usado para configurar as dependências entre eles;
- categoria do serviço: NEW_SERVICE, para serviços implantáveis, e LEGACY_ SERVICE, para serviços legados;
- URI de acesso ao pacote de implementação do serviço;
- tipo do pacote de implementação do serviço: COMMAND_LINE ou TOMCAT. Este atributo define o processo de implantação e execução do serviço. Por simplicidade, limitou-se a estes dois tipos.

Serviços cujo tipo de pacote é definido como COMMAND_LINE devem ser associados a um pacote JAR que contém todas as dependências embutidas nelas. Por outro lado, serviços cujo tipo de pacote é definido como TOMCAT devem ser associados a um pacote WAR. Em ambos os casos, as dependências devem estar embutidas nos pacotes associados;

- *endpoint* do serviço;
- porta de execução do serviço; e
- versão do serviço.

Na definição das arestas, sempre que o vértice destino for um serviço, a operação utilizada deve ser especificada, sendo essa

informação nula nos casos em que o vértice destino constitui um conector ou um vértice final. Quando conectores de disjunção são utilizados, a probabilidade de execução do subgrafo deve ser fornecida como um dos atributos da aresta.

As restrições não funcionais são especificadas usando as definições e as considerações apresentadas anteriormente. No caso de restrições em relação a atributos do recurso, a linguagem utilizada é interpretada usando um analisador descendente recursivo, implementado como uma adaptação do analisador proposto por Fontoura *et al*. (2010). O código equivalente à restrição é automaticamente gerado usando Javassist (Chiba; Nishizawa, 2003), uma biblioteca que permite manipular *bytecodes* Java.

O componente da arquitetura gerador de grafo de dependências foi implementado usando o procedimento proposto e é realizada a partir da descrição das coreografias de entrada, especificadas isoladamente em cada arquivo XML. A representação gerada é mantida apenas internamente como objetos Java.

Além da métrica de vazão utilizada no código apresentado como exemplo, foi implementada, também, a métrica latência, que consiste no tempo decorrido entre a chegada da requisição no serviço e seu processamento. Maiores detalhes sobre a implementação dessas métricas serão discutidos na próxima seção.

Quanto às restrições em relação a atributos do recurso, foi implementada apenas a criação de restrições considerando um conjunto de atributos relacionados aos recursos. Estes atributos são listados a seguir:

- MINIMUM_RESOURCE_USE: indica a cobrança mínima tolerada. Nos casos em que provedores que estipulam cobrança mínima devem ser evitados, a restrição deve ser estabelecida indicando esse atributo como zero, conforme apresentado no exemplo.

- LOCATION: indica o país (com a região, opcionalmente) onde o recurso deve ser alocado.
- COST: indica o custo aceitável para alocação do recurso.
- CLOUD_PROVIDER_REPUTATION: um valor inteiro que indica a reputação do provedor. Quanto maior esse valor, maior a reputação do provedor.
- CLOUD_PROVIDER: indica qual provedor de nuvem deve ser usado na seleção de recursos. Nesse caso, os demais provedores serão ignorados.
- PUBLIC_CLOUD: um valor lógico indicando se ambientes de nuvem pública devem ser usados.
- AVAILABILITY: Disponibilidade é a probabilidade de o sistema estar em funcionamento. Apesar de esse atributo ser considerado uma propriedade de QoS, na abordagem ele é tratado como uma restrição em relação a atributos do recurso. Essa decisão foi tomada porque no protótipo ele é estimado usando unicamente propriedades dos tipos concretos de recurso que não tem relação com a capacidade e, portanto, não podem ser refletidas na representação dos tipos canônicos. Para estimar a disponibilidade, foi utilizado dados analíticos de tempo e de inatividade de provedores de nuvem públicos fornecidos pela CloudHarmony, uma companhia que provê dados sobre desempenho de serviços em nuvem. A metodologia empregada pela empresa não permite uma análise detalhada das estatísticas sobre inatividade de provedores de nuvem. Por exemplo, CloudHarmony monitora uma região elegendo uma única zona de disponibilidade nesta região, de forma que interrupções que ocorrem em zonas de disponibilidade não monitoradas não serão registradas. Apesar dessa limitação, argumenta-se que as

informações providas são suficientes para ter uma visão geral sobre a disponibilidade de provedores de nuvem pública.

O protótipo desenvolvido é disponibilizado como uma biblioteca Java. Dessa forma, a interface com o usuário é realizada por meio de chamadas à API provida por essa biblioteca.

Implementação da síntese de recursos

O primeiro componente da camada de síntese de recursos – Estimador de utilidade – é responsável por estimar os valores da função de utilidade U para as restrições de QoS. No protótipo desenvolvido, esse componente foi implementado considerando duas métricas de QoS: latência e vazão.

No caso de latência, a função de utilidade é calculada usando o tempo de processamento da requisição, T_p, que se refere à duração total do período que essa requisição permanece na fila para uso do recurso e em processamento. Foi utilizado Teoria de Filas (Lazowska *et al.*, 1984) para estimar este valor.

Com base em trabalhos anteriores (Goudarzi; Ghasemazar; Pedram, 2012; Vilaplana *et al.*, 2014; Xiong; Perros, 2009), cada recurso foi modelado como um centro de serviço com múltiplos servidores (modelo *M/M/c*). Ao fazer isso, assume-se que:

- existem c servidores, o que corresponde à quantidade de núcleos da CPU no recurso;
- chegadas de requisições ocorrem a uma taxa
$$\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_{|O_{s_i}|}$$
de acordo com um processo de *Poisson*, em que $\lambda_j \left(1 \leq j \leq \left|O_{s_i}\right|\right)$ é a taxa média de chegadas para uma certa operação no serviço sendo implantado. A carga sobre cada operação é representada no grafo de dependências, sendo estabelecida como uma propagação da carga estimada sobre a coreografia;

- o tempo de serviço tem uma distribuição exponencial com taxa média $\mu = \frac{1}{E(\text{tempo de serviço})}$, onde $E(X)$ denota o valor esperado de X. É assumida uma estratégia de escalonamento *round-robin* no processador. Dessa forma, μ pode ser calculado usando o tempo médio de serviço das operações utilizadas. O tempo de serviço para cada operação, por sua vez, é calculado em função do número de instruções de programa necessárias para implementar essa função e da capacidade do processador do recurso utilizado.
- o *buffer* possui capacidade infinita, de forma que não há limite no número de requisições que ele contém.

Tendo como base essas considerações, de acordo com Barbeau e Kranakis (2007) T_p é dado por:

$$T_p = \frac{1}{\mu} + \frac{C(c,\rho)}{c\mu - \lambda}$$

em que $\rho = \frac{\lambda}{\mu}$ e $C(c,\rho)$ é a probabilidade de que uma requisição que chega seja forçada a esperar na fila (porque todos os servidores estão ocupados), dada pela fórmula de Erlang (Takacs, 1969):

$$C(c,\rho) = \frac{\left(\frac{(c\rho)^c}{c!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!} + \left(\frac{(c\rho)^c}{c!}\right)\left(\frac{1}{1-\rho}\right)}$$

A métrica vazão indica o número de requisições atendidas em um determinado período de tempo. Por simplicidade, é assumido que U pode ser estimada para esta métrica usando:

$$\frac{1}{T_p}$$

A Tabela 5 apresenta a definição completa das métricas de QoS implementadas.

TABELA 5
Definição das métricas de QoS implementadas

Nome	D	$\leq$	$\oplus$	$\wedge$	$\vee$	U	$\odot$
Latência	R^+	$\leq$	$+$	mín.	máx.	T_p	$\wedge$
Vazão	R^+	$\leq$	$+$	máx.	mín.	$\frac{1}{T_p}$	$+$

Fonte: Elaboração própria.

Com relação ao componente Mapeador de recursos, a estimativa é implementada usando a heurística WS-HEU (Yu; Zhang; Lin, 2007). Contudo, essa heurística só permite obter uma solução nos casos em que os tipos de recurso disponíveis possuem capacidade suficiente para satisfazer as restrições de QoS, dada a carga imposta sobre os serviços. Nos casos em que não é possível processar essa carga dentro dos limites estabelecidos pelas restrições, é proposto como solução a criação de novas instâncias dos serviços.

Uma das maiores dificuldades em adicionar instâncias nesse caso é identificar quais serviços estão de fato sobrecarregados. A solução ideal seria estabelecer uma estratégia para determinar a capacidade de admissão da coreografia e, com base nisso, definir os serviços que precisam ter instâncias adicionadas para estender essa capacidade de forma a permitir cargas mais elevadas. Contudo, o desenvolvimento desta estratégia deve lidar com a definição do perfil de cada serviço isoladamente e da coreografia como um todo, de forma a determinar padrões de uso dos recursos. Estes padrões são influenciados por uma série de fatores que envolvem incerteza, como tipos de cliente e intervalo de tempo das requisições, o que torna o desenvolvimento da estratégia complexo.

Em virtude disso, são consideradas duas alternativas mais imediatas. A primeira, identificada como Estratégia I, consiste em

identificar o principal gargalo no conjunto de serviços com relação à estimativa de recursos. Para tal, é estabelecido um indicador calculado como a carga multiplicada pela demanda de recursos para atender cada requisição. Uma nova instância é criada para o serviço que apresenta maior valor para esse indicador.

A segunda alternativa considerada, identificada como Estratégia II, consiste em utilizar os pares que apresentam a maior razão entre o valor do recurso e o peso calculado, sendo criadas novas instâncias para os serviços que constituem esses pares. Em ambas as estratégias, caso os serviços selecionados para ter instâncias adicionadas sejam serviços legados, os seguintes na sequência estabelecida são, então, considerados.

Enquanto na primeira alternativa apenas uma única instância do serviço é adicionada por vez, a segunda permite que um número maior de instâncias seja criado cada vez que o insucesso for detectado, o que teoricamente faria com que o sucesso na síntese seja obtido mais rapidamente, dado que as restrições afetadas são aquelas fim-a-fim. Para verificar se isso realmente ocorre, foi executada a síntese de recursos com cargas variadas usando as coreografias do exemplo e os recursos disponíveis nos provedores considerados. De acordo com a análise, para os casos que ocorre insucesso na estimativa, o uso da Estratégia I (que adiciona instâncias apenas do serviço que constitui o gargalo) faz com que o resultado seja obtido muito mais rapidamente. Nas execuções realizadas, a Estratégia II (que considera os pares com pior razão entre valor e peso) fez com que a estimativa tenha que ser refeita, em média, nove vezes a mais que a alternativa. Isso acontece porque os objetos que apresentam a pior razão entre valor e peso não necessariamente constituem o motivo das restrições não serem satisfeitas. Dessa forma, criar instâncias desses serviços não faz com que sucesso seja obtido.

Com isso, mesmo adicionando apenas uma única instância a cada vez que insucesso for detectado, a alternativa que usa o gargalo constitui uma melhor opção. Uma estratégia para reduzir ainda

mais o tempo de processamento seria redimensionar o número de instâncias adicionais a cada vez que insucesso na estimativa fosse detectado novamente para uma mesma entrada. Por exemplo, poderia ser utilizada uma progressão que considera, a princípio, a adição de uma única instância; na segunda iteração, adicionam-se mais duas instâncias; na terceira iteração, mais quatro instâncias, e assim sucessivamente. Contudo, como pode ser visto nos resultados do experimento, a correção do caso de insucesso ocorre com reexecuções da estimativa de forma linear, fazendo com que essa estratégia leve à criação de instâncias desnecessárias.

Após a inclusão de instâncias e prováveis reexecuções da heurística WS-HEU, o resultado da estimativa de recursos é obtido como sendo o mapeamento de cada serviço na estrutura do grafo de dependências (e suas prováveis instâncias adicionais) para o tipo canônico selecionado. Este mapeamento é, então, usado como entrada para a seleção de recursos.

A seleção de recursos consiste em aplicar as restrições em relação a atributos do recurso estabelecidas. Seu conjunto é ordenado de acordo com as regras discutidas. Para os casos de restrições da mesma categoria, a ordem é estabelecida aleatoriamente. Após a ordenação, filtros e classificadores, implementados de acordo com as restrições, são aplicados sobre os tipos concretos referentes a cada tipo canônico selecionado na primeira etapa da síntese. Esse processo é realizado separadamente para cada serviço, levando em consideração apenas as restrições impostas sobre ele. Estas são avaliadas na sequência estabelecida pela ordenação, sendo que o resultado da aplicação de uma restrição é usado como entrada na restrição seguinte. O processo finaliza quando todas as restrições sobre o serviço tiverem sido avaliadas ou quando o resultado de uma restrição for um conjunto vazio, o que caracteriza um caso de insucesso na seleção de recursos, devendo a estimativa de recursos ser refeita. O tipo concreto para cada serviço é parcialmente selecionado como sendo o primeiro da lista de tipos concretos remanescentes.

Caso sejam adicionadas instâncias de serviços na estimativa de recursos, elas são ignoradas na filtragem e na classificação de recursos. Isso acontece porque as instâncias adicionais estão sujeitas às mesmas restrições, bastando considerar apenas o serviço original. Contudo, essas instâncias são consideradas na consolidação de recursos, uma vez que poderão permitir a consolidação de conjuntos diferentes de serviços. Isso é implementado inicialmente avaliando apenas a demanda do serviço original e o tipo canônico selecionado para este. Caso nenhum serviço possa ser consolidado nos tipos concretos disponíveis para o serviço original, as instâncias adicionais não serão avaliadas. Contudo, se ocorrer a consolidação, o processo se repete considerando a busca a partir de uma instância adicional até que não seja possível consolidar novos recursos.

Na implementação do algoritmo de consolidação de recursos, para estimar a distância entre dois recursos, necessária para calcular o fator de comunicação usado no algoritmo, foi utilizada a fórmula de Vincenty (1975), que calcula a distância entre duas localizações geográficas especificadas por suas coordenadas. Estas são obtidas por meio de consultas à API Google *Maps Geocoding* (Google, 2022b) usando como parâmetros o código ISO 3166-1 alpha-2 (ISO, 2022) do país e demais atributos do endereço (quando eles existem) da região onde os recursos serão implantados.

Os aspectos dinâmicos da abordagem, em parte coordenados pelo componente Adaptador de recursos, não foram implementados no protótipo desenvolvido. Conforme será discutido adiante, foram consideradas diversas mudanças possíveis que podem impactar a encenação das coreografias e, consequentemente, o gerenciamento de recursos, e foi proposto uma solução inicial para parte delas. Contudo, o tratamento de forma ampla das mudanças ocasionadas pela dinamicidade do cenário considerado foi mantido como trabalho futuro.

Implementação do *broker* de recursos

Na implementação do componente descobridor de recursos da camada de *broker* de recursos, é assumido que os dados que descrevem os tipos concretos são fornecidos em arquivos CSV específicos de cada provedor de nuvem. A descoberta automática de recursos é tratada como trabalho futuro. Ela pode ser implementada usando ferramentas como HtmlUnit (Bowler, 2022) para consultas às páginas Web dos provedores, em que os atributos dos recursos são descritos.

Para a implementação do componente alocador de recursos, bem como dos adaptadores que interagem com provedores de nuvem (*CP-Adapter*) e dos *probes* de monitoramento nas VMs instanciadas (*VM-Probe*), foi realizada uma adaptação do *Enactment Engine* (EE) (Leite, 2014; Leite *et al*., 2013a, 2013b), um dos componentes do projeto CHOReOS, desenvolvido pelo Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP) em parceria com instituições europeias. A escolha de adaptar essa plataforma em vez de realizar a implementação por completo foi tomada visando evitar que tarefas puramente de implementação relacionadas à alocação de recursos tivessem que ser refeitas e assim, fosse possível focar na principal contribuição da pesquisa que é a síntese de recursos.

Conforme ilustrado na Figura 40, o CHOReOS EE permite a implantação de uma coreografia de serviços dada a sua descrição em alto nível. Esta plataforma não considera o uso de linguagens de modelagem de coreografias, sendo a especificação feita usando a API disponibilizada. A implantação da coreografia inclui a instalação e configuração de toda a pilha de software necessária para a execução do serviço, o que inclui o sistema operacional e um servidor Web no caso de serviços Web. Os números na figura indicam a ordem com que os componentes de software são instalados.

A direção das setas vai do componente que coordena a instalação para o componente sendo instalado. A plataforma CHOReOS EE também realiza a configuração das dependências entre os serviços, de forma que a coreografia possa posteriormente ser encenada.

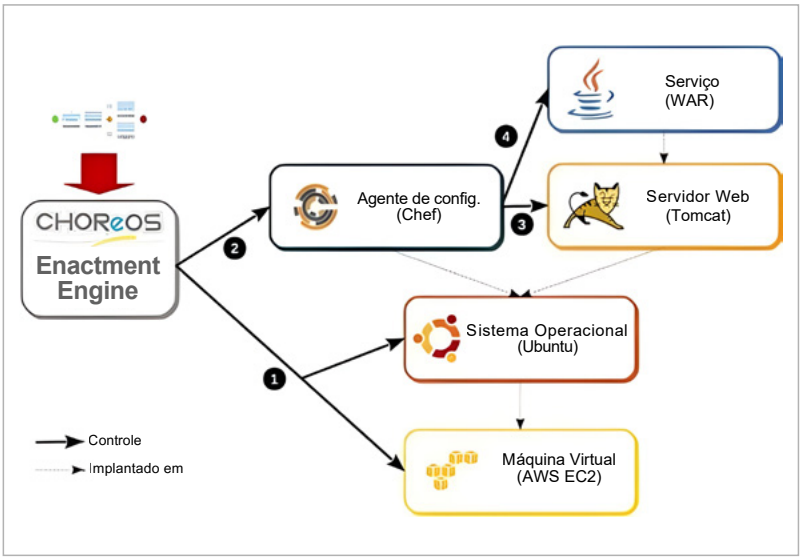


Figura 40 – Princípio geral do funcionamento do CHOReOS Enactment Engine
Fonte: Elaboração própria.

O funcionamento do CHOReOS EE se baseia no uso da ferramenta Chef, que consiste em um agente responsável pela configuração de uma VM, com base em *scripts* gerados de acordo com as características do serviço sendo implantado. Embora o componente utilize implementações preestabelecidas para o software – Ubuntu (Canonical, 2022b), utilizado como sistema operacional, e Tomcat (Apache, 2022) como servidor Web – a utilização de Chef como gerenciador de configuração facilita a eventual utilização de diferentes versões ou produtos alternativos, uma vez que ele abstrai as peculiaridades do *software* utilizado.

Apesar da seleção de recursos ter sido proposta no CHOReOS EE, seus desenvolvedores não implementaram essa funcionalidade,

sendo que um único tipo preestabelecido é usado na implantação de todos os serviços. Dessa forma, grande parte da adaptação realizada aqui, na implementação desse componente, consistiu em incluir, no processo de implantação de coreografias realizado pelo CHOReOS EE, a síntese de recursos desenvolvida. As demais modificações feitas na implementação original são listadas a seguir:

- atualização das bibliotecas utilizadas para versões mais recentes, visando melhoria no desempenho;
- expansão dos atributos usados na descrição de recurso, de forma a incluir os atributos propostos na abordagem, entre eles, custo e localização;
- inclusão do suporte a múltiplas regiões de um mesmo provedor, de forma que a localização do recurso possa ser levada em consideração na alocação;
- inclusão de um adaptador para uma nuvem pública simulada;[8]
- inclusão de um mecanismo de redirecionamento dos eventos gerados pelos *probes* para o componente analisador na arquitetura proposta;
- implementação de adaptadores para as entidades de domínio comuns nas duas implementações, que dizem respeito à representação de serviços, de coreografias e de recursos;
- correção de pequenos *bugs* no código.

Para tornar transparente a inclusão de nossa abordagem de síntese de recursos, permitindo seu uso em soluções baseadas no

8 Este foi implementado usando CloudSim (Buyya; Ranjan; Calheiros, 2009; Calheiros *et al.*, 2011), um *toolkit* extensível que permite a modelagem e simulação de sistemas de computação em nuvem e políticas de provisionamento de recursos nesse tipo de ambiente. Foi utilizado esse adaptador para simular chamadas a um provedor real e, assim, realizar testes do protótipo.

CHOReOS EE, não se alterou significativamente o processo de implantação de coreografias implementado por ele, sendo a síntese de recursos adicionada como uma etapa adicional nesse processo.

No protótipo, foi implementado um adaptador da notação do grafo de dependências para a representação usada no CHOReOS EE. Dessa forma, do ponto de vista desse componente, o conjunto de coreografias sendo implantadas consiste em uma única composição. Essa visão não tem impacto sobre o resultado esperado, uma vez que as dependências são configuradas da mesma forma.

Pelos mesmos motivos apresentados na seção anterior com relação aos aspectos dinâmicos da abordagem, a implementação do componente analisador de eventos foi estabelecida também como trabalho futuro.

A principal limitação das propostas encontrados na literatura para implantação de coreografias é que elas geralmente não consideram todas as atividades relacionadas ao gerenciamento de recursos. Diante disso, neste capítulo foi apresentado uma arquitetura para a execução coordenada da abordagem proposta para a modelagem de coreografias, restrições e recursos e para a descoberta, estimativa e seleção de recursos.

A arquitetura foi discutida com base em seu cenário de uso. Em seguida, foi apresentada a implementação de um protótipo de parte dessa arquitetura, a qual incorporou apenas os componentes diretamente relacionados às atividades efetuadas em tempo de implantação das coreografias. A implementação dos demais componentes, que são relacionados aos aspectos dinâmicos, isto é, que ocorrem em tempo de encenação das coreografias, foram mantidos como trabalho futuro. Contudo, é importante ressaltar que esses componentes são suficientes para validar a proposta apresentada neste livro, uma vez que o principal problema nele discutido diz respeito à satisfação de restrições não funcionais nas atividades relacionadas à implantação dos serviços.

7

Desdobramentos da abordagem desenvolvida

O principal objetivo deste livro – propor uma abordagem para a implantação automatizada e eficiente de múltiplas coreografias de serviço, sujeitas às restrições não funcionais associadas aos serviços e a sua encenação – foi atingido. Contudo, foram identificadas diversas possibilidades de extensão deste trabalho que, por restrição de tempo, não puderam ser desenvolvidas, além de alguns pontos que devem ser melhor investigados como desdobramento dos objetivos iniciais da pesquisa desenvolvida. Neste capítulo serão discutidos os principais.

Maior nível de abstração na avaliação de restrições não funcionais

Foi proposto um arcabouço que facilita a especificação de restrições não funcionais. Contudo, seu uso pressupõe que esta tarefa é realizada quantificando o valor-alvo para essas restrições. Isso é uma desvantagem, pois o usuário nem sempre consegue discriminar com exatidão os valores suportados para que a implantação e encenação das coreografias sejam realizadas de maneira satisfatória.

Em virtude dessa possível imprecisão, a experiência dos clientes no uso das coreografias pode ser insatisfatória ou a implantação das coreografias pode ser realizada de maneira excessivamente restrita, causando gastos desnecessários. Mesmo considerando a especificação dos valores-alvo usando faixas de tolerância, o problema persiste, pois ainda assim nem sempre é possível definir os limites com exatidão. Dessa maneira, uma possível extensão da abordagem seria oferecer um maior nível de abstração sobre a especificação e a consequente avaliação de restrições não funcionais.

Uma alternativa possível seria descrever o valor-alvo para as restrições usando, por exemplo, linguagem natural como em "O cumprimento de pedidos deve ser realizado de maneira rápida". Na fase inicial da pesquisa desenvolvida foi sugerida a adoção desta estratégia, como pode ser visto na publicação de Gomes, Costa e Bencomo (2014). De acordo com a definição proposta por Chung *et al.* (2012), o objetivo inicial era tratar restrições não funcionais como metas flexíveis (*soft-goals*), isto é, que precisam ser satisfeitas não absolutamente, mas de maneira suficientemente adequada, e para as quais o critério de satisfação não é definido de forma clara (quantificada).

Esse tipo de notação é desejável por ser facilmente compreendido pelo usuário, independentemente de seu conhecimento técnico. Essa exigência de alto nível do usuário seria interpretada de formas diferentes, dependendo dos serviços que impactam sobre ela e do contexto de seu uso, conforme realizado em alguns trabalhos (Maiden, *et al*., 2014; Torres; Bencomo; Astudillo, 2012). Com isso, a quantificação das restrições é realizada em tempo de execução, de acordo com a experiência dos clientes e das execuções prévias dos serviços.

Uma vez que a quantificação de restrições de maneira automática deve lidar com diversos desafios associados ao tratamento de incertezas e de influências entre as restrições, optou-se por

simplificar a especificação adotando a quantificação do valor-alvo pelo usuário, para que fosse possível focar no desenvolvimento da síntese de recursos. Contudo, essa estratégia pode ser incorporada em trabalhos futuros na arquitetura proposta como parte da camada de síntese de recursos.

Priorização de restrições não funcionais

Na abordagem proposta, não foi considerada a inclusão de prioridades sobre as restrições estabelecidas. Esse comportamento foi adotado por ser assumido que são estabelecidas apenas restrições essenciais, que deviam necessariamente ser satisfeitas com mesmo grau de importância. Contudo, em alguns cenários pode haver restrições opcionais que se não forem satisfeitas não serão um impacto significativo à encenação da coreografia. Por exemplo, pode-se definir que a latência de encenação de uma dada coreografia deve ser inferior a um certo valor e que é desejável que seus serviços sejam implantados em uma certa localidade, embora possa ser alterada caso isso seja necessário para que a latência requisitada seja satisfeita. Neste exemplo, a localidade requisitada deixa de ser uma restrição que necessariamente deve ser satisfeita pela síntese.

A priorização de restrições também pode ser estabelecida por meio da definição de pesos, que podem ser usados para influenciar a seleção de recursos favorecendo a satisfação de uma determinada restrição. Por exemplo, para uma dada coreografia pode haver restrições associadas à vazão e à segurança, cada qual com seu valor-alvo, de forma que a segurança tem maior prioridade que a vazão. Com isso, a seleção de recursos realizada pela síntese terá como resultado um conjunto que satisfaça ambas as restrições, mas que privilegiem a escolha de recursos que ofereçam um maior nível de segurança.

Apesar das vantagens descritas, a possibilidade de definir essa priorização tem o inconveniente de requerer que o usuário assuma mais responsabilidades, dificultando a implantação de coreografias. Essa dificuldade constitui outra razão pela qual preferiu-se simplificar a especificação de restrições. Além disso, a incorporação de prioridades nas restrições na abordagem requer a adequação do arcabouço de especificação de restrições, além da adaptação do mecanismo de síntese propriamente dito e da redefinição do conceito de função de utilidade adotado no trabalho, de forma a considerar a influência dos pesos sobre as restrições. Diante disso, o oferecimento de priorização de restrições como trabalho futuro requer, também, uma análise sobre o balanceamento entre os benefícios obtidos e o impacto na usabilidade e nos critérios avaliados para a solução.

Inclusão do usuário no *loop*

A estimativa e a seleção de recursos na abordagem adotam o custo de utilização dos recursos como critério de decisão nos casos em que há diferentes opções que satisfaçam todas as restrições. De maneira semelhante, na consolidação de recursos, a demanda de comunicação entre os serviços é usada como critério de decisão. O uso desses atributos foi fixado na abordagem por ser considerado que eles representam os critérios mais críticos nas atividades em questão. Contudo, é reconhecido que pode haver cenários em que o uso de outros atributos como critério de decisão pode ser algo necessário, por exemplo, quando se deseja implantar o maior número possível de serviços em um mesmo provedor.[9] Diante disso,

9 Na abordagem, o comportamento de privilegiar a seleção de recursos de um mesmo provedor pode ser obtido pela definição de restrições classificatórias usando, por exemplo, a reputação do provedor como critério. Contudo, a possibilidade de defini-lo como critério de decisão promoverá ainda mais sua ocorrência.

uma extensão possível do trabalho seria permitir a configuração dos critérios usados nas decisões.

Além de realizar essa configuração, é proposto que o usuário seja incluído no *loop* em outras etapas da abordagem. Mais precisamente, sugere-se, também, que poderiam ser oferecidas diferentes soluções como resultado da síntese de recursos, em que cada solução privilegiaria algum critério, como menor custo para utilização dos recursos, implantação em um único provedor, entre outros. Com base nos critérios estabelecidos para gerar os resultados, o usuário selecionaria a melhor opção, que seria usada como resultado final. Além disso, um aprimoramento ainda mais elaborado seria considerar feedbacks fornecidos pelo usuário sobre os resultados da síntese para aprimorar decisões futuras ou estabelecer novos critérios na geração de resultados.

Atualmente, essas estratégias não são consideradas porque sua adoção tem a desvantagem destacada na subseção anterior, que é conferir ao usuário maiores responsabilidades, dificultando o uso da abordagem.

Aperfeiçoamento da avaliação de restrições de QoS

Referente a restrições de QoS, há alguns aspectos que precisam ser melhor consolidados para que a abordagem possua maior robustez e seja mais alinhada aos cenários reais de sua utilização. São eles: precisão na estimativa de QoS, tratamento efetivo de restrições de QoS sensíveis a localização e estabelecimento de SLAs.

Estimativa de QoS

O isolamento da estratégia utilizada para estimar a QoS para cada métrica foi uma simplificação adotada no trabalho para que

fosse possível focar no desenvolvimento do mecanismo de síntese de recursos. Contudo, para que a abordagem proposta seja amplamente empregada em um cenário com múltiplos provedores de nuvem, é preciso aperfeiçoar como esse aspecto é tratado. Mais precisamente, é necessário propor abordagens concretas para a estimativa de QoS, além de tratamentos mais precisos para lidar com imprecisões nessas estimativas.

Há diversos resultados já consolidados que visam estimar a QoS com base em dados reais referentes aos provedores de nuvem. Nesse sentido, as estimativas são obtidas usando estratégias baseadas em *benchmarking* ou se baseando no uso prévio de recursos nestes provedores (Acs; Zsolt; Gergely, 2014; Iosup; Yigitbasi; Epema, 2011; LI *et al*., 2010; Schad; Dittrich; Quiané-Ruiz, 2010). Alternativa possível é considerar soluções que fornecem dados de monitoramento de atributos de QoS. Por exemplo, CloudHarmony, além de fornecerem dados sobre disponibilidade dos provedores (usados na implementação do protótipo desenvolvido), também dispõem informações sobre o desempenho de recursos nesses ambientes.

O grande desafio em incluir na abordagem a estimativa de QoS de maneira precisa, mesmo considerando soluções como essas, é lidar com a variabilidade do desempenho oferecido por recursos instanciados em nuvens públicas. As informações oferecidas pelos provedores públicos de IaaS sobre a capacidade de hardware e o desempenho dos tipos de recurso oferecidos não são suficientes para prever como as aplicações neles implantadas irão se comportar (Phillips; Engen; Papay, 2012). Isso acontece porque são utilizadas diferentes gerações de máquinas físicas na infraestrutura desses provedores, conforme declarado por eles mesmos (por exemplo, AWS e Microsoft Azure).

Além disso, há flutuação na utilização dessa infraestrutura a longo prazo (diferentes períodos do ano) e a curto prazo (diferentes horas do dia). Como consequência, o desempenho realmente

oferecido pode variar de acordo com o período de tempo em que a utilização é realizada e de acordo com a região utilizada, ou mesmo considerando recursos de uma mesma região, como indicado em Lenk *et al*. (2011), Oloughlin e Gillam (2014), Ou *et al*. (2012).

Foi considerado que este desafio não invalida a abordagem desenvolvida, uma vez que a estimativa de QoS é tratada como algo isolado. Dessa forma, o funcionamento da síntese de recursos na solução não sofre modificações, mesmo considerando mudanças na estratégia para estimativa de QoS.

Restrições de QoS sensíveis à localização dos recursos

No trabalho, assumiu-se que as restrições de QoS são especificadas com base em métricas de QoS diretamente relacionadas à capacidade dos recursos utilizados, como latência de processamento. Contudo, ao se considerar restrições fim-a-fim, existem algumas métricas de QoS cuja avaliação depende também da sobrecarga imposta pela comunicação entre os serviços. Um exemplo imediato seria tempo de resposta fim-a-fim, cujos valores devem considerar não somente a latência de processamento em cada serviço, mas também a latência de comunicação entre eles. Na abordagem, foi considerada a satisfação de restrições que se encaixam nessa categoria e foi proposto um tratamento inicial.

No tratamento proposto, as restrições de QoS sensíveis à localização dos recursos é avaliada em duas etapas. A primeira é realizada durante a estimativa de recursos, com a avaliação integral das demais restrições de QoS. Nessa etapa, apenas a parcela da restrição que é dependente da capacidade de recurso (como latência de processamento no tempo de resposta) é avaliada. Após isso, durante a seleção de recursos, para cada serviço que está relacionado a uma restrição de QoS ainda não totalmente avaliada, ou seja, aquelas que dependem da localização dos recursos, é selecionado um conjunto de tipos

concretos de recurso admissíveis, em vez de um único tipo concreto. Essa seleção é realizada durante a consolidação de recursos e considera um custo adicional admissível para os recursos candidatos. Dessa forma, o resultado da consolidação de recursos seria o mapeamento de conjuntos de serviços para uma tupla que representa o tipo mais favorável (selecionado de acordo com a estratégia de troca usando como critério a localidade), juntamente com outros tipos que podem ser considerados como substitutos a esse (embora possam representar uma redução global menor no atraso de comunicação).

Após a seleção de recursos, a parcela das restrições que é dependente da localização dos serviços (como latência de comunicação no tempo de resposta) seria verificada usando o tipo de recurso mais favorável especificado em cada tupla. Caso alguma restrição fosse violada ao considerar a parcela remanescente nesta restrição, seria proposta a criação de uma nova instância do problema MMKP para tentar mapear os serviços a tipos que permitam a satisfação das restrições ainda violadas. Para isso, os tipos admissíveis incluídos na consolidação de recursos seriam usados como recursos candidatos.

O tratamento proposto na avaliação dessa categoria de restrições foi motivado pelo fato de os tipos concretos de recurso geralmente possuírem tipos com capacidade equivalente em regiões distintas, tornando viável a substituição de modo a satisfazer as restrições sensíveis à localização. Contudo, o tratamento indicado precisa ser melhor fundamentado e ainda é preciso desenvolver uma análise mais aprofundada da sua viabilidade, assim como propor alternativas quando soluções factíveis não são encontradas. Por essa razão, o tratamento desse aspecto é tido como indicação de trabalho futuro.

Estabelecimento de SLAs

Além de melhorias na estimativa, é preciso tornar a abordagem proposta mais alinhada ao modelo de negócio atualmente

adotado em soluções de nuvem. Uma extensão, nesse sentido, seria considerar a criação de acordos de nível de serviço (*Service Level Agreements* – SLAs), com políticas de compensação e penalidade quando eles não são cumpridos. Esses acordos especificam as expectativas e obrigações mínimas das partes envolvidas no processo de negócio (Buco *et al*., 2004) e, dessa forma, seu estabelecimento pode oferecer maior confiança na utilização da abordagem, assim como permitir maior maleabilidade no tratamento de restrições uma vez que possíveis violações podem ser neles discriminadas.

Um desafio em propor essa adequação é gerenciar os múltiplos níveis de SLA, dado que qualquer acordo estabelecido com o usuário deve ter como base acordos firmados com provedores de nuvem. Contudo, os SLAs atualmente oferecidos por provedores de nuvem pública são limitados aos atributos de custo e disponibilidade, havendo pouca ou nenhuma garantia com relação ao desempenho oferecido e demais atributos de qualidade. Diante disso, além de propor mecanismos para o estabelecimento e gerenciamento de SLAs, é preciso, também, oferecer estratégias para obter garantias sobre os atributos não assegurados pelos provedores públicos, de forma a ser possível considerá-los.

Aperfeiçoamento do modelo de recursos

Por simplicidade, o modelo de recursos utilizado na síntese é gerado considerando apenas tipos de VM cuja instanciação é realizada sob demanda. Contudo, o modelo de negócio adotado por provedores de nuvem inclui um conjunto com diferentes categorias de recurso. Como discutido por Suleiman *et al*. (2012), além da cobrança por uso sob demanda, que constitui a base do modelo de nuvem, os recursos podem ser oferecidos, também, na forma de assinaturas por um determinado período de tempo,

considerando um modelo de cobrança pré-pago em que o pagamento é realizado antes da utilização dos recursos.

Alternativa é a cobrança por meio da combinação dessas categorias, com instâncias dedicadas definidas pela assinatura e adição de novas instâncias sob demanda, quando há a necessidade de uma quantidade maior de recurso. Opções mais sofisticadas também estão disponíveis. Por exemplo, a Amazon permite aos usuários licitarem a capacidade não utilizada de sua infraestrutura por meio de *Spot Instances* (Amazon, 2022e), por meio do qual na solicitação do recurso é fornecido um preço que o usuário está disposto a pagar, sendo o recurso instanciado somente se este preço for igual ou superior ao valor de mercado. Cada uma dessas categorias tem vantagens e desvantagens que podem ser exploradas na síntese de recursos, de forma a obter soluções ainda mais satisfatórias.

Diante disso, uma possível extensão na abordagem seria definir as categorias de recurso que fossem mais adequadas para cada cenário. Contudo, para que isso seja possível, é necessário conhecimento sobre o perfil de uso dos recursos pela aplicação que está sendo implantada, além de estratégias para que a flutuação de custo dos recursos seja refletida no modelo gerado e para que incertezas associadas a pagamentos prévios na alocação de recursos não causem prejuízos desnecessários.

Além da inclusão de novos tipos no modelo de recursos, a modelagem atualmente proposta deve ser aperfeiçoada de forma a refletir todos os aspectos existentes na utilização de recursos em nuvem. Mais precisamente, foram adotadas algumas simplificações, como ignorar a cobrança realizada por provedores de nuvem pública pelos dados trafegados em rede e pelo armazenamento de dados (imagens das VMs, discos adicionais etc.). Foi incorporado ao modelo apenas o custo associado à execução da VM, sendo os demais atributos ignorados. Essa simplificação foi adotada porque estas cobranças somente são realizadas quando um certo limite é ultrapassado.

Por exemplo, a cobrança por transferência de dados na Amazon só é realizada quando o volume de dados trafegados ultrapassa 1 GB por mês. Contudo, é evidente que esse aspecto deve ser incluído na abordagem para que os resultados obtidos representem de fato o custo esperado. Essa adequação também deve lidar com desafios associados ao conhecimento do perfil das aplicações e do tratamento de incertezas.

Outra característica que precisa ser melhor fundamentada na abordagem é o fato de que os provedores definem os parâmetros de suas ofertas de diferentes maneiras. Por exemplo, Microsoft Azure realiza a cobrança da transferência de dados de acordo com a largura de banda contratada, ao passo que a Amazon cobra pelo volume de dados transmitidos. Além dos desafios já discutidos, essa característica incorpora a necessidade de considerar este nível de heterogeneidade na geração do modelo de recursos.

Análise dos efeitos do compartilhamento de recursos

O compartilhamento de recursos, tanto físicos quanto virtuais, tem a vantagem de reduzir os custos associados não somente à implantação de serviços, mas também ao próprio gerenciamento dos recursos. Por exemplo, como forma de explorar a prevalência de sistemas com arquitetura *multicore* e *manycore*, essa estratégia é cada vez mais adotada no escalonamento de recursos físicos, com outras práticas conhecidas como computação verde (*Green Computing*) (Kurp, 2008), com o objetivo de reduzir o consumo de energia em *datacenters*.

A alocação de múltiplas VMs em um mesmo recurso físico é possível devido à capacidade de isolamento de desempenho oferecida pelos *hypervisors*, em componentes como núcleos da CPU, memória e disco. Contudo, componentes como *cache* da CPU e largura de banda para entrada e saída (em memória, em rede e em disco)

são muito difíceis de serem isolados nos *hypervisors* existentes atualmente (Novaković *et al.*, 2013). Isso pode comprometer o desempenho com o aumento do número de VMs colocalizadas na mesma máquina física. Adicionalmente, a colocalização de VMs aumenta o risco de exploração de vulnerabilidades de uma VM por outra, diminuindo a segurança (Kaufman, 2010).

Esses mesmos problemas são visualizados no nível das aplicações ao implantar dois ou mais serviços compartilhando a mesma VM, como realizado na técnica de consolidação de recursos proposta na abordagem. Nesse caso, além dos problemas listados, o isolamento entre serviços, que constitui uma das principais vantagens ao se utilizar virtualização, desaparece quando eles são implantados na mesma VM.

Entre as estratégias que podem ser utilizadas para lidar com essa dificuldade está a modelagem de desempenho dos serviços, visando a identificação de padrões de uso dos recursos. Com isso, a alocação de recursos pode ser potencialmente melhorada para permitir a colocalização de serviços de forma a impedir a interferência entre eles. Contudo, conforme evidenciado nas seções anteriores deste capítulo, a adoção dessa estratégia requer conhecimento sobre a aplicação (ou sobre cada serviço isoladamente) para que seja possível definir seu perfil. Além disso, o problema de degradação de desempenho é agravado ao se considerar o gerenciamento de recursos em múltiplos *datacenters* geograficamente distribuídos, uma vez que este problema aumenta devido à alta variação e frequente escassez de recursos de rede na Internet (Xu *et al.*, 2013). Ademais, a ausência de controle sobre os recursos físicos em nuvens públicas dificulta a resolução desse problema de maneira efetiva.

Como trabalho futuro, estratégias como definir o perfil de uso de recursos pelas aplicações podem ser incluídas como parte do mecanismo de consolidação de recursos proposto. Com isso, as decisões tomadas ao realizar essa tarefa poderão levar em consideração

um melhor balanceamento entre a interferência no desempenho causada pelo compartilhamento de recursos e as reduções no custo e no atraso de comunicação proporcionadas ao implantar múltiplos serviços em um mesmo recurso. Porém, o desenvolvimento dessa melhoria requer, também, a análise de mecanismos para tratamento de problemas decorrentes da colocalização de VMs nos cenários em que é possível gerenciar os recursos físicos (por exemplo, quando se usa uma nuvem privada).

Adaptação dinâmica

O principal foco da abordagem apresentada é a estimativa e a seleção inicial dos recursos necessários para satisfazer um conjunto de restrições não funcionais impostas sobre um conjunto de coreografias de serviços. Contudo, um importante aspecto que deve ser considerado é garantir que essas restrições se mantenham satisfeitas durante a encenação das coreografias.

As restrições não funcionais podem ser violadas em tempo de execução devido a diferentes motivos. De imediato, existem aqueles que são inerentes ao cenário considerado, como flutuação na carga das coreografias implantadas ou submissão de novas coreografias que compartilham serviços com aquelas previamente implantadas (o que também causa flutuação na carga dos serviços implantados).

Além disso, a elasticidade de uso da infraestrutura, aliada à pressão para acomodar mudanças nas regras de negócio, requer que as coreografias de serviços também sejam adaptáveis (Leite *et al.*, 2013b). Dessa forma, é natural supor que poderão ocorrer alterações nas restrições não funcionais estabelecidas, além de mudanças diretas nas coreografias, como adaptações nos padrões de conexão dos serviços para refletir alterações na lógica da composição ou no modelo de negócio por ela implementado.

Em complemento às mudanças diretamente associadas às coreografias, a dinamicidade presente em ambientes de nuvem faz com que aplicações implantadas nesse tipo de ambiente tenham que lidar com mudanças que podem ocorrer nos recursos disponibilizados. Como exemplos de mudanças nessa categoria, pode haver alteração nos tipos concretos oferecidos ou variação nos tipos previamente existentes, como aumento no custo financeiro ou encerramento da disponibilização de um determinado tipo em algumas regiões. Na contramão disso, podem surgir novas oportunidades, como o oferecimento de novos tipos de recurso ou a ampliação de tipos disponibilizados em certas regiões.

Além de mudanças requisitadas e aquelas relacionadas ao ambiente, há também adaptações relacionadas à solução proposta, uma vez que suposições erradas ou estimativas mal elaboradas ou desatualizadas podem invalidar a seleção de recursos obtida. Por exemplo, a QoS de fato oferecida ao utilizar um recurso pode ser diferente da QoS estimada, fazendo com que a estimativa de recursos (e consequentemente a seleção) precise ser adaptada. Na proposta, todas essas mudanças foram consideradas e foi proposto uma arquitetura que permite a inclusão de mecanismos de adaptação para lidar com todas elas. De maneira preliminar, nesta seção é proposto uma solução apenas para parte daquelas mudanças listadas como inerentes no início dessa discussão. Optou-se por aperfeiçoar o tratamento da abordagem com relação às atividades realizadas durante a implantação inicial das coreografias e por limitação de tempo não foi possível aprofundar essa parte da pesquisa. Dessa forma, o tratamento das adaptações possíveis de maneira completa, assim como a consolidação da solução descrita a seguir, são considerados como indicação de trabalho futuro.

Conforme anunciado, é definido um tratamento para quando há mudanças na carga sobre serviços previamente implantados. Essas mudanças podem ser ocasionadas em dois cenários:

nova demanda sobre coreografias previamente implantadas ou inclusão de novas coreografias que compartilham serviços com aquelas já em encenação. Em ambos os casos, é proposto como estratégia para absorver as mudanças na carga a reexecução da síntese de recursos. O argumento para propor tal alternativa é que a abordagem de síntese proposta possui desempenho satisfatório suficiente para ser utilizada em tempo de execução. Outro argumento é que a solução obtida com a abordagem de síntese é estável, ou seja, as mudanças a serem realizadas na alocação de recursos serão pontuais, concentradas naqueles serviços que tiveram mudança na carga, sem afetar consideravelmente os demais serviços. Esses argumentos são baseados em experimentos cuja descrição é omitida neste livro por questão de espaço. Contudo, para que seja possível consolidar o uso desse tratamento como estratégia adaptativa é preciso explorar outros aspectos como, por exemplo, decidir os gatilhos que indicam a necessidade de reexecução da síntese.

Expansão da abordagem para outros componentes da pilha de software

A abordagem proposta tem como foco o gerenciamento de recursos apenas levando em consideração os serviços. Esta perspectiva é aceitável ao assumir que a sobrecarga causada por componentes que viabilizam a comunicação e a coordenação dos serviços é negligenciável. Contudo, em alguns cenários, sobretudo de maior escala, isso não é realista. Diante disso, uma das extensões propostas para o trabalho é considerar componentes do *middleware* de comunicação e aqueles responsáveis pela coordenação entre os serviços como entidades de primeira classe no problema.

Com isso, a estimativa e a seleção de recursos seriam realizadas levando em consideração, também, as demandas de recursos por esses componentes, fazendo com que estas atividades fossem

realizadas de maneira mais precisa. Contudo, para que isso seja possível, é necessário, entre outras coisas, investigar formas de estimar o impacto causado por estes componentes na QoS da coreografia, por exemplo, definindo modelos analíticos sobre desempenho no nível de *middleware* (Bouloukakis *et al*., 2017).

Outra consideração relevante é que na abordagem foi adotado uma visão monolítica dos serviços, na qual os componentes que implementam cada serviço são tratados como um único componente. Apesar de não inviabilizar a abordagem desenvolvida, o desacoplamento dos componentes (ou camadas) que formam um serviço pode permitir que a especificação de restrições não funcionais e o gerenciamento de recursos sejam realizados de maneira mais precisa. Por exemplo, se houvesse tal desacoplamento apenas a camada de dados do serviço de pagamento discutido no exemplo do cenário deveria ser implantada em um ambiente privado, podendo as demais camadas desse serviço ser implantadas de maneira distribuída em recursos alocados em nuvens públicas. Isso garante maior liberdade na implantação dos serviços, facilitando ainda mais a satisfação das restrições. A principal dificuldade em propor tal tratamento é o aumento na complexidade, uma vez que esse desacoplamento aumenta a granularidade e adiciona um nível extra de dependências. Com isso, além da influência que um serviço pode ter sobre outro, passa a ser necessário considerar a influência que componentes de um mesmo serviço têm entre si.

Descentralização da arquitetura

Da maneira como foi proposta, a execução dos componentes da arquitetura ocorre de maneira centralizada. Do ponto de vista da implantação das coreografias, isso não constitui necessariamente um problema, dado que as atividades são realizadas de maneira desacoplada de sua encenação. Contudo, ao se considerar a

realização dessas atividades em tempo de execução, a centralização da arquitetura passa a ser um obstáculo, sobretudo se for usado a mesma estratégia de síntese proposta em tempo de implantação. Isso acontece porque qualquer violação significativa nas restrições gerará a necessidade de processamento, tornando assim a síntese de recursos um gargalo.

A descentralização da arquitetura tem como motivação os mesmos benefícios alcançados com a coordenação dos serviços por meio de coreografias. Em outras palavras, a seleção de recursos por meio da cooperação entre componentes distribuídos da arquitetura permite que decisões sejam tomadas localmente, sem a necessidade de que haja sempre coordenação global. Contudo, é preciso revisar não somente a arquitetura, mas principalmente a implementação desenvolvida, além de propor soluções quando a descentralização não é algo imediato. Por exemplo, para garantir que soluções próximas do valor ótimo sejam obtidas, a avaliação de restrições de QoS na estimativa de recursos de forma distribuída pode requerer uma demanda de comunicação muito elevada entre os componentes que implementam essa funcionalidade.

Adequação da abordagem para o estilo arquitetural de microsserviços

A grande tendência no desenvolvimento baseado em serviços é propor o uso de serviços com escopo cada vez mais bem definido, assumindo a responsabilidade por tarefas reduzidas. Dessa forma, as aplicações passam a ser desenvolvidas como um conjunto de serviços pequenos, cada um funcionando em seu próprio processo e se comunicando por meio de mecanismos de baixa sobrecarga (Pahl, 2015). Além de benefícios como separação de interesses e modularização, esta estratégia de desenvolvimento, conhecida como estilo arquitetural de microsserviços (Newman, 2021), dá liberdade para

que serviços escalem de maneira independente, permitindo melhor uso dos recursos.

Apesar de a abordagem não oferecer uma solução que considera todos os aspectos desse cenário, isso é estabelecido como algo a ser realizado. De fato, algumas partes da abordagem proposta já foram projetadas tendo esse estilo arquitetural em mente. Um exemplo é o mecanismo de consolidação de recursos, que visa permitir o compartilhamento de recursos entre um conjunto de serviços, o que é uma premissa quando se tem serviços com escopo bem limitado e, consequentemente, baixa demanda de recurso. Desvantagens como a perda de isolamento entre os serviços, resultante do compartilhamento de VMs, podem ser facilmente amenizadas ao considerar adequações na abordagem proposta, por exemplo, usando contêineres como unidade de alocação em vez de VMs.

Considerações finais

Neste livro, foi apresentada uma abordagem que se beneficia da multiplicidade de tipos de recurso disponibilizados em ambientes de nuvem para implantar um conjunto de coreografias de serviços sujeitas a restrições não funcionais. Esta abordagem visa o uso eficiente dos recursos de forma a reduzir os custos associados à utilização dos recursos e ao atraso de comunicação entre os serviços. A adoção cada vez mais frequente de soluções baseadas em serviços vem mudando a forma como sistemas são concebidos, projetados e implementados. As aplicações passam a ser desenvolvidas como composições de serviços que são executados de maneira distribuída.

Neste novo cenário, coreografias de serviços constituem um modelo de composição promissor, por permitir que a coordenação desses serviços seja realizada sem a necessidade de controladores centrais. Todavia, a encenação de coreografias deve ser realizada visando não apenas as funcionalidades associadas à aplicação, mas também propriedades não funcionais oriundas do modelo de negócio que essa aplicação implementa ou resultantes da busca por soluções cada vez mais competitivas e com maior qualidade. Para garantir que essas propriedades sejam oferecidas, é preciso atuar nas atividades realizadas desde a implantação dos serviços, dado que o ambiente de execução tem forte impacto sobre sua satisfação. Isso gera diversos desafios, sobretudo, ao considerar que um mesmo serviço pode assumir diferentes responsabilidades resultantes de seu compartilhamento em múltiplas composições.

A abordagem proposta tem como objetivo permitir a implantação de um conjunto de coreografias visando a satisfação de restrições não funcionais requeridas sobre os serviços. Os fundamentos considerados nessa abordagem englobam atividades associadas a todo o ciclo de gerenciamento de recursos, além do desenvolvimento de modelos empregados na automação dessas atividades. Mais precisamente, foi apresentado mecanismos para a modelagem de coreografias de serviços com restrições não funcionais associadas. Tendo como base os modelos criados usando estes mecanismos, foi indicada a geração de uma estrutura que representa uma visão conjunta das coreografias e das restrições. O objetivo em gerar esta estrutura é unificar a representação de cada serviço e, assim, tornar explícitas as dependências entre os serviços, facilitando a identificação dos efeitos de seu provável compartilhamento.

Além disso, foi proposta uma forma de modelagem de recursos que considera um conjunto de atributos comumente utilizados na descrição dessas entidades, e que visa abstrair sua representação ao se considerar um ambiente de nuvem híbrida com infraestrutura privada e múltiplos provedores de nuvem pública. Devido à variedade de tipos de recurso, foi apresentado uma estratégia que, a partir da modelagem inicial, gera uma representação adicional dos recursos considerando a intersecção de tipos e, assim, reduz a sobrecarga da estimativa de recursos.

Alicerçados nos modelos de coreografias e recursos, foi desenvolvida uma estratégia de síntese que obtém a estimativa de recursos por meio da satisfação de restrições específicas para cada serviço e do balanceamento da contribuição dos serviços em restrições fim-a-fim, globalmente para todas as coreografias. Após a estimativa, a estratégia de síntese desenvolvida seleciona os tipos de recurso mais apropriados para os serviços enquanto promove o compartilhamento de recursos e reduz não somente o custo associado ao uso dos recursos alocados, mas também o atraso de comunicação

entre os serviços. Com base na abordagem desenvolvida, foi proposta uma arquitetura que coordena todas as atividades relacionadas ao gerenciamento de recursos.

Na abordagem, a satisfação de restrições não funcionais estabelecidas sobre um conjunto de coreografias representa o principal fator considerado para decidir a configuração de recursos adequada para implantar essas coreografias. Diante disso, um dos aspectos que fazem com que a abordagem desenvolvida seja eficiente é o fato da análise realizada na avaliação dessas restrições considerar os diferentes papéis assumidos pelos serviços devido ao seu provável compartilhamento. Essa eficiência é reforçada pela redução do atraso de comunicação entre os serviços ao se adotar a localidade dos recursos como outro critério na seleção, e ao se explorar a capacidade excedente nos tipos de recurso disponíveis, por meio da consolidação de recursos.

Além da utilização de uma estrutura conjunta para a representação de todas as coreografias, foi proposto outros métodos que favorecem a eficiência e o desempenho da abordagem de síntese desenvolvida. Entre eles, está a manipulação dos modelos de recurso como forma de viabilizar a utilização de um vasto conjunto de tipos; a categorização das restrições não funcionais associada ao tratamento da atividade de estimativa de recursos isoladamente da atividade de seleção de recursos; e a abstração tanto da estimativa de QoS quanto da interação com provedores de nuvem.

O uso da abordagem proposta pode beneficiar usuários interessados em implantar uma ou mais coreografias de serviços, por meio da automação das atividades relacionadas ao gerenciamento de recursos. Ao adotar essa abordagem, é possível implantar um conjunto de coreografias de maneira eficiente, pois se leva em consideração a interferência causada pelo compartilhamento de serviços entre elas, possibilitando satisfazer todas as restrições não funcionais enquanto é obtida a redução dos custos associados.

Portanto, a contribuição apresentada neste livro é ampla o suficiente para considerar outros cenários além da implantação de coreografias de serviços. A solução elaborada pode ser generalizada para categorias de problemas que apresentam as mesmas características do cenário considerado. Mais precisamente, a abordagem de síntese pode ser empregada (com as devidas modificações) em outros cenários em que seja necessário mapear um conjunto de recursos (em geral) para entidades que utilizam esses recursos na realização de alguma tarefa ou atividade de forma cooperada, sujeita a um conjunto de restrições. Por exemplo, algumas das contribuições desenvolvidas podem ser aplicadas no gerenciamento de recursos para redes elétricas inteligentes (*Smart Grids*) (Farhangi, 2009; Masters, 2013). As demais contribuições do trabalho são discutidas a seguir.

- *Definição de uma nova notação para representação de coreografias de serviços*: a representação de coreografias usando a notação do grafo de processo, e sua consequente transformação na notação do grafo de dependências, permite explicitar as dependências entre os serviços e, principalmente, explorar os efeitos do compartilhamento de um mesmo serviço entre múltiplas composições. Com isso, a implantação das coreografias modeladas pode ser facilmente realizada explorando esses aspectos.
- *Proposta de um arcabouço para a especificação de restrições não funcionais*: o arcabouço proposto abstrai a representação de restrições de QoS, propiciando que a especificação de restrições dessa categoria não fique subordinada a um conjunto preestabelecido de métricas. Além disso, a linguagem proposta para representação de restrições em relação a atributos do recurso facilita a incorporação de restrições dessa categoria por gerar automaticamente o código que é usado no processo de avaliação implementado pelo arcabouço.

- *Formalização do problema de síntese de recursos*: foi proposta a formalização do problema da síntese de recursos como um problema de otimização e foi discutido sua complexidade. A formalização proposta teve como objetivo tornar os conceitos relacionados não ambíguos, estabelecendo uma nomenclatura padrão adotada no decorrer do trabalho, além de evidenciar a complexidade do problema.
- *Indicação de uma estratégia para estimativa de recursos*: foi definida uma estratégia de estimativa de recursos baseada em um procedimento de emparelhamento entre os serviços que compõem as coreografias e tipos canônicos que representam os recursos disponíveis. Essa estratégia considera a adição de instâncias de serviços como forma de viabilizar a satisfação de restrições em cenários de grande demanda.
- *Proposta de uma estratégia para consolidação de recursos*: foi proposta uma estratégia para consolidação de recursos como forma de aproveitar possíveis capacidades excedentes nos recursos selecionados. Esta estratégia obtém, como resultado adicional, a redução no atraso de comunicação entre os serviços. A obtenção dessa redução é facilitada pela adoção de uma estratégia auxiliar que diminui a distância física entre os serviços implantados, considerando a troca de tipos de recurso por aqueles que possuem melhor localidade.
- *Definição de uma arquitetura que considera todas as atividades relacionadas à implantação de coreografias de serviços*: como forma de integrar as estratégias desenvolvidas, foi proposta uma arquitetura que inclui, também, aspectos das demais atividades relacionadas ao gerenciamento de recursos. Essa arquitetura descreve uma solução para a implantação de coreografias de serviços e para a adaptação de recursos em tempo de execução. Parte dela foi implementada em um protótipo que considera as atividades relacionadas à implantação dos serviços.

Por fim, uma das principais contribuições do trabalho é o fato da abordagem proposta permitir a implantação conjunta de múltiplas coreografias de serviços, oferecendo uma solução que contempla todas as atividades relacionadas ao gerenciamento de recursos. Além disso, a abordagem considera diferentes categorias de restrições não funcionais e diferentes padrões de interação entre os serviços. Grande parte desses aspectos são tratados de maneira muito superficial nos demais trabalhos encontrados, o que motivou o desenvolvimento da abordagem apresentada. Esta abordagem permite preencher essa lacuna, ao mesmo tempo em que reduz o custo de utilização dos recursos e o atraso de comunicação entre os serviços.

Referências

Aarts, Emile; Korst, Jan. *Simulated annealing and Boltzmann machines*: a stochastic approach to combinatorial optimization and neural computing. Hoboken: John Wiley & Sons, 1991.

Acs, Sandor; Zsolt, Nemeth; Gergely, Mark. A novel approach for performance characterization of IaaS Clouds. *In*: European conference on parallel processing, 2014, Porto. *Proceedings* [...]. Berlin: Springer, 2014. p. 109-120.

Adobe. *Selecting a cloud provider*. San Jose: Adobe, 2022. Disponível em: https://spark.adobe.com/page/pn39b/. Acesso em: 8 jan. 2022.

Aiello, Marco; Giorgini, Paolo. Applying the Tropos methodology for analyzing web services requirements and reasoning about Qualities of Services. *Department of Information and Communication Technology,* Povo, p. 1-17, maio 2004.

Alam, Sarfraz; Chowdhury, Mohammad; Noll, Josef. SenaaS: an event-driven sensor virtualization approach for internet of things cloud. *In*: IEEE international conference on networked embedded systems for enterprise applications, 2010, Suzhou. *Proceedings* [...]. Washington, DC: IEEE, 2010. p. 1-6.

Alhamazani, Khalid *et al*. An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing*, Berlin, v. 97, n. 4, p. 357-377, 2015.

Alonso, Gustavo *et al*. *Web services*: concepts, architectures and applications. Cham: Springer, 2004.

Alrifai, Mohammad; Risse, Thomas; Nejdl, Wolfgang. A hybrid approach for efficient Web service composition with end-to-end QoS constraints. *ACM Transactions on the Web*, Nova York, v. 6, n. 2, p. 1-31, 2012.

Amato, Flora; Moscato, Francesco. Exploiting cloud and workflow patterns for the analysis of composite cloud services. *Future Generation Computer Systems*, Amsterdã, v. 67, p. 255-265, 2017.

Amazon. *Amazon CloudWatch*. Seattle: Amazon, 2022a. Disponível em: https://aws.amazon.com/cloudwatch/. Acesso em: 7 jan. 2022.

AMAZON. *Amazon Web Services*. Seattle: Amazon, 2022b. Disponível em: https://aws.amazon.com/. Acesso em: 7 jan. 2022.

AMAZON. *AWS Pricing Calculator*. Seattle: Amazon, 2022c. Disponível em: https://calculator.aws/#/. Acesso em: 7 jan. 2022.

AMAZON. *AWS CloudFormation*. Seattle: Amazon, 2022d. Disponível em: https://aws.amazon.com/cloudformation/. Acesso em: 7 jan. 2022.

AMAZON. *Instâncias spot do Amazon EC2*: Execute cargas de trabalho tolerantes a falhas com desconto de até 90%. Seattle: Amazon, 2022e. Disponível em: https://aws.amazon.com/ec2/spot/. Acesso em: 13 jan. 2022.

AN ARCHITECTURAL blueprint for autonomic computing. 3. ed. Hawthorne: IBM, 2005.

APACHE. *Apache Tomcat*. Wilmington: The Apache Software Foundation, 2022. Disponível em: https://tomcat.apache.org. Acesso em: 12 jan. 2022.

ARDAGNA, Danilo *et al*. MODAClouds: A model-driven approach for the design and execution of applications on multiple clouds. *In*: INTERNATIONAL WORKSHOP ON MODELING IN SOFTWARE ENGINEERING, 4., 2012, Zurique. *Proceedings* [...]. Washington, DC: IEEE, 2012. p. 50-56.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: a survey. *Computer Networks*, Amsterdam, v. 54, n. 15, p. 2787-2805, 2010.

AUSTIN, Daniel *et al*. *Web services architecture requirements*. Cambridge: W3C, 2004.

AUTILI, Marco *et al*. Synthesizing an Automata-based Representation of BPMN2 Choreography Diagrams. *ModComp@MoDELS*, Atenas, p. 67-77, out. 2014.

AUTILI, Marco; INVERARDI, Paola; TIVOLI, Massimo. CHOREOS: large scale choreographies for the future internet. *In*: IEEE CONFERENCE ON SOFTWARE MAINTENANCE, REENGINEERING, AND REVERSE ENGINEERING, 2014, Antwerp. *Proceedings* [...]. Washington, DC: IEEE, 2014. p. 391-394.

BALA, Raj *et al*. Magic Quadrant for Cloud Infrastructure and Platform Services. *Gartner*, Stamford, 27 jul. 2021. Disponível em: https://www.gartner.com/doc/reprints?id=1-271OE4VR&ct=210802&st=sb. Acesso em: 8 jan. 2022.

BARBEAU, Michel; KRANAKIS, Evangelos. *Principles of ad-hoc networking*. Hoboken: John Wiley & Sons, 2007.

BARKER, Adam; WALTON, Christopher; ROBERTSON, David. Choreographing web services. *IEEE*, Washington, DF, v. 2, n. 2, p. 152-166, 2009.

Barreto, Charlton. Web services business process execution language version 2.0. *OASIS standard*, Woburn, v. 11, n. 120, p. 5, 2007.

Barros, Alistair; Dumas, Marlon; Oaks, Phillipa. A critical overview of the web services choreography description language. *BPTrends Newsletter,* n. 3, p. 1-24, 2005.

Barros, Alistair; Dumas, Marlon; Hofstede, Arthur. Service interaction patterns. *In*: international conference on business process management, 3., 2005, Nancy. *Proceedings* [...]. Berlim: Springer, 2005. p. 302-318.

Bartolini, Cesare *et al*. Non-functional analysis of service choreographies. *In*: international workshop on principles of engineering service-oriented systems, 4., 2012, Zurich. *Proceedings* [...]. Washington, DC: IEEE, 2012a. p. 8-14.

Bartolini, Cesare *et al.* Quality Requirements for Service Choreographies. *In*: international conference on web information systems and technologies, 8., 2012, Porto. *Proceedings* [...]. Setúbal: Scitepress, 2012b. p. 143-148.

Baset, Salman. Cloud slas: present and future. *ACM SIGOPS Operating Systems Review*, Nova York, v. 46, n. 2, p. 57-66, 2012.

Bencomo, Nelly *et al*. Report on the 7th international workshop on models@runtime. *ACM SIGSOFT Software Engineering Notes*, Nova York, v. 38, n. 1, p. 27-30, 2013.

Berkhin, Pavel. A survey of clustering data mining techniques. *In*: Kogan, Jacob; Nicholas, Charles; Teboulle, Marc. (ed.). *Grouping multidimensional data*. Berlim: Springer, 2006. p. 25-71.

Bernardi, Simona; Merseguer, José; Petriu, Dorina. *An UML profile for dependability analysis and modeling of software systems.* Saragoça: Universidad de Zaragoza, 2008a.

Bernardi, Simona; Merseguer; José; Petriu, Dorina. Adding dependability analysis capabilities to the marte profile. *In*: international conference on model driven engineering languages and systems, 11., 2008, Toulouse. *Proceedings [...]*. Berlin: Springer, 2008b. p. 736-750.

Bertolino, Antonia *et al.* Trends and research issues in soa validation. *In*: cardellini, Valeria *et al.* (ed.). *Performance and dependability in service computing*: concepts, techniques and research directions. Pensilvânia: IGI Global, 2012. p. 98-115.

Bi, Jing *et al*. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. *In*: IEEE international conference on cloud computing, 3., 2010, Miami. *Proceedings* [...]. Washington, DC: IEEE, 2010. p. 370-377.

Blair, Gordon *et al*. Perspectives on cloud computing: interviews with five leading scientists from the cloud community. *Journal of Internet Services and Applications*, Porto Alegre, v. 2, n. 1, p. 3-9, 2011.

Blair, Gordon; Bencomo, Nelly; France, Robert. Models@run.time. *Computer*, Nova York, v. 42, n. 10, p. 22-27, 2009.

Bocciarelli, Paolo; D'ambrogio, Andrea. A bpmn extension for modeling non functional properties of business processes. *In*: symposium on theory of modeling & simulation, 2011, Boston. *Proceedings* [...]. San Diego: Society for Computer Simulation Internacional, 2011. p. 160-168.

Boehm, Barry William; Brown, John; Lipow, Myron. Quantitative evaluation of software quality. *In*: international conference on software engineering, 2., 1976, San Francisco. *Proceedings [...]*. Washington, dc: ieee, 1976. p. 592-605.

Bouloukakis, Georgios *et al*. Timeliness evaluation of intermittent mobile connectivity over pub/sub systems. *In*: international conference on performance engineering, 2017, L'Aquila. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2017. p. 275-286.

Bouras, Ioannis *et al*. Mapping of Quality of Service Requirements to Resource Demands for IaaS. *In*: international conference on cloud computing and services science, 9., 2019, Creta. *Proceedings* [...]. Cham: Springer, 2020. p. 263-270.

Bowler, Mike. *HtmlUnit*. Toronto: Gargoyle Software, 2022. Disponível em: https://htmlunit.sourceforge.io. Acesso em: 12 jan. 2022.

Brasil. Presidência da República. Lei do Marco Civil da Internet no Brasil . *Diário Oficial da União*, Poder Executivo, Brasília, df, 24 abr. 2014.

Brasil. Secretaria de Coordenação de Sistemas. Departamento de Segurança da Informação e Comunicações. Norma complementar 14/in01/dsic/gsipr, de 30 de janeiro de 2012. Diretrizes relacionadas à segurança da informação e comunicações para o uso de computação em nuvem nos órgãos e entidades da administração pública federal. *Diário Oficial da União*, Poder Executivo, Brasília, df, 13 mar. 2018. p. 6.

Buco, Melissa *et al*. Utility computing sla management based upon business objectives. *ibm Systems Journal*, Riverton, v. 43, n. 1, p. 159-178, 2004.

Bundy, Alan; Wallen, Lincoln. Breadth-first search. *In*: bundy, Alan; Wallen, Lincoln (ed.). *Catalogue of artificial intelligence tools*: symbolic computation. Berlim: Springer, 1984. p. 13.

Burke, Paul. The output of a queuing system. *Operations Research*, Catonsville, v. 4, n. 6, p. 699-704, 1956.

Buschmann, Frank; Henney, Kevlin; Schimdt, Douglas. *Pattern-oriented software architecture*: on patterns and pattern language. Hoboken: John Wiley & Sons, 2007.

Business Process Model and Notation (bpmn). Version 2.0. Needham: omg, 2011.

Buyya, Rajkumar. Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, Amsterdã, v. 25, n. 6, p. 599-616, 2009.

Buyya, Rajkumar; Beloglazov, Anton; Abawajy, Jemal. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. In: international conference on parallel and distributed processing techniques and applications, 2010, Las Vegas. *Proceedings* [...]. Ithaca: Cornell University, 2010. p. 1-12.

Buyya, Rajkumar; Ranjan, Rajiv; Calheiros, Rodrigo. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. *In*: international conference on high performance computing & simulation, 2009, Leipzig. *Proceedings* [...]. Washington, dc: ieee, 2009. p. 1-11.

Buyya, Rajkumar; Ranjan, Rajiv; Calheiros, Rodrigo. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. *In*: international conference on algorithms and architectures for parallel processing, 10., 2010, Busan. *Proceedings* [...]. Berlim: Springer, 2010. p. 13-31.

Calheiros, Rodrigo *et al*. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, Nova Jersey, v. 41, n. 1, p. 23-50, 2011.

Camargo, Raphael *et al*. Grid: An architectural pattern. *In*: conference on pattern languages of programs, 11., 2004, Monticello. *Proceedings* [...]. São Paulo: Department of Computer Science - usp, 2004.

Canonical. *Juju*. Londres: Canonical, 2022a. Disponível em: https://juju.is. Acesso em: 7 jan. 2022.

Canonical. *Ubuntu*. Londres: Canonical, 2022b. Disponível em: https://ubuntu.com. Acesso em: 12 jan. 2022.

Cavalcante, Everton *et al*. Optimizing services selection in a cloud multiplatform scenario. *In*: latin america conference on cloud computing and communications, 2012, Porto Alegre. *Proceedings* [...]. Washington, dc: ieee, 2012. p. 31-36.

Charrada, Faouzi Ben *et al*. Approximate placement of service-based applications in hybrid clouds. *In*: international workshop on enabling technologies, 21., 2012, Toulouse. *Proceedings* [...]. Washington, dc: ieee, 2012. p. 161-166.

Chauvel, Franck *et al*. Models@Runtime to Support the Iterative and Continuous Design of Autonomic Reasoners. *In*: international workshop on models@ run.time, 8., 2013, Miami. *Proceedings* [...]. Atenas: mrt, 2013. p. 26-38.

Chiba, Shigeru; Nishizawa, Muga. An easy-to-use toolkit for efficient Java bytecode translators. *In*: international conference on generative programming and component engineering, 2., Erfurt, 2003. *Proceedings* [...]. Berlim: Springer, 2003. p. 364-376.

Chung, Lawrence *et al*. *Non-functional requirements in software engineering*. Nova York: Springer, 2012.

Chung, Lawrence; Leite, Julio Cesar. On non-functional requirements in software engineering. *In*: borgida, Alexander. *et al*. (ed.). *Conceptual modeling*: foundations and applications. Berlim: Springer, 2009. v. 5600, p. 363-379.

Ciancone, Andrea *et al*. Klapersuite: An integrated model-driven environment for reliability and performance analysis of component-based systems. *In*: international conference on modelling techniques and tools for computer performance evaluation, 49., 2011, Zurique. *Proceedings* [...]. Berlim: Springer, 2011. p. 99-114.

Clark, Jim *et al*. (ed.). *ebxml Business Process Specification Schema Version 1.01*. Geneva: un/cefact; Oasis, 2001.

Cloudharmony. *Transparency for the cloud*. CloudHarmony, 2023. Disponível em: https://cloudharmony.com. Acesso em: 15 fev. 2023.

Cloudorado. *Cloud Server Comparison*. Cloudorado, 2022. Disponível em: https://www.cloudorado.com/cloud_server_comparison.jsp. Acesso em: 7 jan. 2022.

Clouds laboratory. *InterGrid*: Internetworking Islands Of Grids. Melbourne: clouds Laboratory, 2022. Disponível em: http://www.cloudbus.org/intergrid/. Acesso em: 8 jan. 2022.

Correia, Isabel; Gouveia, Luís; Saldanha-da-gama, Francisco. Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, Amsterdã, v. 35, n. 6, p. 2103-2113, 2008.

Daras, Petros *et al.* Why do we need a content-centric future Internet? Proposals towards content-centric Internet architectures. *Information Society and Media Journal*, European Commission, Networked Media Unit, 2009.

Decker, Gero *et al*. bpel4Chor: Extending bpel for modeling choreographies. *In*: ieee international conference on web services, 2007, Salt Lake City. *Proceedings* [...]. Washington, dc: ieee, 2007. p. 296-303.

Decker, Gero *et al*. Modeling service choreographies using bpmn and bpel4Chor. *In*: international conference on advanced information systems engineering, 20., 2008, Montpellier. *Proceedings* [...]. Berlim: Springer, 2008. p. 79-93.

Decker, Gero; Kopp, Oliver; Barros, Alistair. An Introduction to Service Choreographies. *It-Information Technology*, Berlim, v. 50, n. 2, p. 122-127, 2008.

Deng, Yi *et al.* cvm – a communication virtual machine. *Journal of Systems and Software*, Amsterdã, v. 81, n. 10, p. 1640-1662, 2008.

Di costanzo, Alexandre; De assunção, Marcos Dias; Buyya, Rajkumar. Harnessing cloud technologies for a virtualized distributed computing infrastructure. *ieee*, Washington, dc, v. 13, n. 5, p. 24-33, 2009.

Di marco, Antinisca *et al*. Yet another meta-model to specify non-functional properties. *In*: international workshop on quality assurance for service--based applications, 2011, Lugano. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2011. p. 9-16.

Dijkman, Remco; Dumas, Marlon; Ouyang, Chun. Semantics and analysis of business process models in bpmn. *Information and Software Technology*, v. 50, n. 12, p. 1281-1294, 2008.

Dillon, Tharam; Wu, Chen; Chang, Elizabeth. Cloud computing: Issues and Challenges. *In*: ieee international conference on advanced information networking and applications, 24., 2010, Perth. *Proceedings* [...]. Washington, dc: ieee, 2010. p. 27-33.

Dolstra, Eelco; Bravenboer, Martin; Visser, Eelco. Service configuration management. *In*: international workshop on software configuration management, 12., 2005, Lisboa. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2005. p. 83-98.

Eisa, Mona *et al*. Trends and directions in cloud service selection. *In*: ieee symposium on service-oriented system engineering, 2016, Oxford. *Proceedings* [...]. Washington, dc: ieee, 2016. p. 423-432.

Ej technologies. *The Award-Winning all-in-one Java Profiler*. Chicago: Ej Technologies, 2022. Disponível em: https://www.ej-technologies.com/products/jprofiler/overview.html. Acesso em: 10 jan. 2022.

Emerson, Allen. Temporal and modal logic. *In*: van leeuwen, Jan. (ed.). *Formal Models and Semantics*. Cambridge: mit Press, 1991. p. 995-1072.

Engler, Lasse. *bpel gold*: choreography on the service bus. Tese (Doutorado em Engenharia de Software) – Institute of Architecture of Application Systems, University of Stuttgart, Estugarda, 2009.

Farhangi, Hassan. The path of the smart grid. *ieee power and energy magazine*, Washington, dc, v. 8, n. 1, p. 18-28, 2009.

Ferry, Nicolas *et al*. Managing multi-cloud systems with CloudMF. *In*: nordic symposium on cloud computing & internet technologies, 2., 2013, Oslo. *Proceedings* [...]. Nova York: acm, 2013a. p. 38-45.

Ferry, Nicolas *et al*. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. *In*: ieee international conference on cloud computing, 6., 2013, Santa Clara. *Proceedings* [...]. Washington, dc: ieee, 2013b. p. 887-894.

Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. Tese (Doutorado em Ciência da Informação e Computação) – University of California, Irvine, 2000.

Flexera. *Cloud Management Platform*.Itasca: Flexera, 2022. Disponível em: https://www.flexera.com/products/cloud-management-platform. Acesso em: 7 jan. 2022.

Flügge, Martin; Tourtchaninova, Diana. Ontology-derived Activity Components for Composing Travel Web Services. *Berliner xml Tage*, p. 133-150, 2004.

Fontoura, Marcus *et al*. Efficiently evaluating complex boolean expressions. *In*: sigmod international conference on management of data, 2010, Indianopolis. *Proceedings* [...]. Nova York: acm, 2010. p. 3-14.

Foster, Ian *et al*. The physiology of the grid. *In*: berman, Fran; Fox, Geoffrey; Hey, Anthony (ed.). *Grid computing*: making the global infrastructure a reality. Hoboken: John Wiley & Sons, 2003. p. 217-249.

Foster, Ian *et al*. Cloud computing and grid computing 360-degree compared. *In*: grid computing environments workshop, 2008, Austin. *Proceedings* [...]. Washington, dc: ieee, 2008-2009. p. 1-10.

Foster, Ian; Kesselman, Carl; Tuecke, Steven. The Anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, Thousand Oaks, v. 15, n. 3, p. 200–222, 2001.

Fowler, Martin. *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley Professional, 2002.

Friesen, Donald; Langston, Michael. Variable sized bin packing. *siam journal on computing*, Philadelphia, v. 15, n. 1, p. 222-230, 1986.

Garey, Michael; Johnson, David. *Computers and intractability*: a guide to the theory of np-completeness. Nova York: wh Freeman, 1979.

Garg, Saurabh Kumar *et al*. Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers. *Journal of Parallel and Distributed Computing*, Amsterdã, v. 71, n. 6, p. 732-749, 2011.

Garg, Saurabh Kumar; Versteeg, Steve; Buyya, Rajkumar. A framework for comparing and ranking cloud services. *Journal of Future Generation Computer Systems*, Amsterdã, v. 29, n. 4, p. 1012-1023, 2013.

Georgantas, Nikolaos *et al*. Service-oriented distributed applications in the future internet: The case for interaction paradigm interoperability. *In*: european conference on service-oriented and cloud computing, 2., 2013, Málaga. *Proceedings [...]*. Berlin: Springer, 2013. p. 134-148.

Giese, Holger; Wagner, Robert. Incremental model synchronization with triple graph grammars. *In*: international conference on model driven engineering languages and systems, 9., 2006, Genova. *Proceedings* [...]. Berlim: Springer, 2006. p. 543-557.

Goldchleger, Andrei. *InteGrade*: um sistema de middleware para computação em grade oportunista. 2004. Dissertação (Mestrado em Ciência da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2004.

Goldman, Alfredo; Ngoko, Yanik; Milojicic, Dejan. An analytical approach for predicting QoS of web services choreographies. *In*: international middleware conference, 13., 2012, Montreal. *Proceedings* [...]. Nova York: Association of Computing Machinery, 2012. p. 1-6.

Gomes, Raphael *et al*. A model-based approach for the pragmatic deployment of service choreographies. *In*: Celesti, Antonio; Leitner, Philipp. (ed.). *Advances in service-oriented and cloud computing*: works of ESOCC 2015. Cham: Springer, 2016. v. 567, p. 153-165.

Gomes, Raphael; Costa, Fábio; Bencomo, Nelly. On modeling and satisfaction of non-functional requirements using cloud computing. *In*: Latin American conference on cloud computing and communications, 2., 2013, Maceió. *Proceedings* [...]. Washington, DC: IEEE, 2014. p. 1-6.

Google. *App Engine documentation*. Mountain View: Google, 2022a. Disponível em: https://cloud.google.com/appengine/docs. Acesso em: 7 jan. 2022.

Google. *Geocoding API*. Mountain View: Google Maps Platform, 2022b. Disponível em: https://developers.google.com/maps/documentation/geocoding/overview. Acesso em: 12 jan. 2022.

Google. *Create a VM with a custom machine type*. . Mountain View: Google Cloud, 2022c. Disponível em: https://cloud.google.com/compute/docs/instances/creating-instance-with-custom-machine-type. Acesso em: 11 jan. 2022.

Goudarzi, Hadi; Ghasemazar, Mohammad; Pedram, Massoud. SLA-based optimization of power and migration cost in cloud computing. *In*: International symposium on cluster, cloud and grid computing, 12., 2012, Ottawa. *Proceedings* [...]. Washington, DC: IEEE, 2012. p. 172-179.

Grozev, Nikolay; Buyya, Rajkumar. Performance modelling and simulation of three-tier applications in cloud and multi-cloud environments. *The Computer Journal*, Oxford, v. 58, n. 1, p. 1-22, 2015.

Gudgin, Martin *et al*. *SOAP version 1.2 – part 1*: messaging framework (second edition). Cambridge: W3C, 2007.

Guttmann-beck, Nili; Hassin, Refael. Approximation algorithms for minimum k-cut. *Algorithmica*, Berlim, v. 27, n. 2, p. 198-207, 2000.

Haeser, Gabriel; Ruggiero, Márcia Gomes. Aspectos teóricos de simulated annealing e um algoritmo duas fases em otimização global. *Trends in Computational and Applied Mathematics*, Minas Gerais, v. 9, n. 3, p. 395-404, 2008.

Hall, Mark *et al*. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, Nova York, v. 11, n. 1, p. 10-18, 2009.

Hartigan, John; Wong, Manchek. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society*, Nova Jersey, v. 28, n. 1, p. 100-108, 1979.

HASHICORP. *Terraform*. 2023. Disponível em: https://www.terraform.io. Acesso em: 15 fev. 2023.

HOFREITER, Birgit; HUEMER, Christian. A model-driven top-down approach to inter-organizational systems: From global choreography models to executable BPEL. *In*: IEEE CONFERENCE ON E-COMMERCE TECHNOLOGY AND IEEE CONFERENCE ON ENTERPRISE COMPUTING, E-COMMERCE AND E-SERVICES, 10., 5., 2008, Arlington. *Proceedings [...]*. Washington, DC: IEEE, 2009. p. 136-145.

HOWARD, Foster *et al*. Model-based analysis of obligations in web service choreography. *In*: ADVANCED INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 2006, Guadalupe. *Proceedings* [...]. Washington, DC: IEEE, 2006. p. 149.

HUANG, Kuo-Chan; SHEN, Bo-Jun. Service deployment strategies for efficient execution of composite SaaS applications on cloud platform. *Journal of Systems and Software*, Amsterdã, v. 107, p. 127-141, 2015.

HUMBLE, Jez; FARLEY, David. *Continuous Delivery*: Reliable Software Releases through Build, Test, and Deployment Automation. Boston: Addison-Wesley, 2010.

HUMMAIDA, Abdul; PATON, Norman; SAKELLARIOU, Rizos. Adaptation in cloud resource configuration: a survey. *Journal of Cloud Computing*, Berlim, v. 5, n. 1, p. 1-16, 2016.

IOSUP, Alexandru; YIGITBASI, Nezih; EPEMA, Dick. On the performance variability of production cloud services. *In*: INTERNATIONAL SYMPOSIUM ON CLUSTER, CLOUD AND GRID COMPUTING, 11., 2011, Newport Beach. *Proceedings* [...]. Washington, DC: IEEE, 2011. p. 104-113.

ISO. *ISO 3166 Country Codes*. Vernier: ISSO, 2022. Disponível em: https://www.iso.org/iso-3166-country-codes.html. Acesso em: 12 jan. 2022.

ISSARNY, Valérie *et al*. Service-oriented middleware for the future internet: state of the art and research directions. *Journal of Internet Services and Applications*, Porto Alegre, v. 2, n. 1, p. 23-45, 2011.

JAVADI, Bahman; ABAWAJY, Jemal; BUYYA, Rajkumar. Failure-aware resource provisioning for hybrid cloud infrastructure. *Journal of Parallel and Distributed Computing*, Amsterdã, v. 72, n. 10, p. 1318-1331, 2012.

JENSEN, Finn *An introduction to Bayesian networks*. Londres: UCL Press, 1996.

KANG, Jangha; PARK, Sungsoo. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, Amsterdã, v. 147, n. 2, p. 365-372, 2003.

Kattepur, Ajay; Georgantas, Nikolaos; Issarny, Valérie. QoS composition and analysis in reconfigurable web services choreographies. ieee international conference on web services, 20., 2013, Santa Clara. *Proceedings [...]*. Washington, dc: ieee, 2013. p. 235-242.

Kaufman, Lori. Can public-cloud security meet its unique challenges? *ieee Security & Privacy*, Washington, dc, v. 8, n. 4, p. 55-57, 2010.

Kavantzas, Nickolas *et al*. (ed.). *Web services choreography description language version 1.0*. Cambridge: W3C, 2005.

Keahey, Kate. Nimbus: open source infrastructure-as-a-service cloud computing software. *In*: workshop on adapting applications and computing services to multi-core and virtualization, 2., 2009, Zurique. *Proceedings* [...]. Meyrin: cern, 2009.

Keller, Steven. Specifying software quality requirements with metrics. *System and Software Requirements Engineering*, Washington, dc, p. 145-163, 1990.

Kent, Stuart. Model driven engineering. *In*: international conference on integrated formal methods, 3., 2002, Turku. *Proceedings* [...]. Berlin: Springer, 2002. p. 286-298.

Khaluf, Lial; Gerth, Christian; Engels, Gregor. Pattern-based modeling and formalizing of business process quality constraints. *In*: international conference on advanced information systems engineering, 23., 2011, Londres. *Proceedings* [...]. Berlim: Springer, 2011. p. 521-535.

Khan, Shahadatullah. *Quality adaptation in a multi-session adaptive multimedia system*: model and architecture. 1998. Tese (Doutorado em Filosofia) – Department of Electronical and Computer Engineering, University of Victoria, Vitória, 1998.

Kim, Hyunjoo *et al*. Autonomic management of application workflows on hybrid computing infrastructure. *Scientific Programming*, Londres, v. 19, n. 2, 3, p. 75-89, 2011.

Kopp, Oliver *et al*. bpmn4tosca: A domain-specific language to model management plans for composite applications. *In*: international workshop on business process modeling notation, 4., 2012, Viena. *Proceedings* [...]. Berlim: Springer, 2012. p. 38-52.

Kurp, Patrick. Green computing. *Communications of the acm*, Nova York, v. 51, n. 10, p. 11-13, 2008.

Lambert, Douglas; Cooper, Martha. Issues in supply chain management. *Industrial Marketing Management*, Amsterdã, v. 29, n. 1, p. 65-83, 2000.

Lawton, George. Developing software online with platform-as-a-service technology. *Computer*, Washington, DC, v. 41, n. 6, p. 13-15, 2008.

Lazowska, Edward *et al*. *Quantitative system performance*: computer system analysis using queueing network models. Hoboken: Prentice Hall, 1984.

Leite, Leonardo Alexandre. *Implantação automatizada de composições de serviços web de grande escala*. 2014. Dissertação (Mestrado em Ciência da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

Leite, Leonardo *et al.* . Um middleware para encenação automatizada de coreografias de serviços web em ambientes de computação em nuvem. *In*: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 31., 2013, Brasília, DF. *Anais*... HAL Documentation, 2013a. p. 1-14.

Leite, Leonardo *et al*. A systematic literature review of service choreography adaptation. *Service Oriented Computing and Applications*, Berlim, v. 7, n. 3, p. 199-216, 2013b.

Lenk, Alexander *et al.* What are you paying for? Performance benchmarking for infrastructure-as-a-service offerings. *In*: IEEE International Conference on Cloud Computing, 4., 2011, Washington. *Proceedings* [...]. Washington, DC: IEEE, 2011. p. 484-491.

Leymann, Frank. *Web services flow language (WSFL 1.0)*. Armonk: IBM, 2001.

Li, Ang *et al*. CloudCmp: comparing public cloud providers. *In*: SIGCOMM conference on Internet measurement, 10., 2010, Melbourne. *Proceedings* [...]. Nova York: ACM, 2010. p. 1-14.

Liu, Changbin *et al*. Cloud resource orchestration: A data-centric approach. *In*: Conference on Innovative Data Systems Research, 11., 2011, Asilomar. *Proceedings [...]*. Mountain View: Creativer Commons, 2011. p. 1-8.

Liu, Yutu; Ngu, Anne; Zeng, Liang. QoS computation and policing in dynamic web service selection. *In*: International World Wide Web Conference on Alternate Track Papers & Posters, 13., 2004, Nova York. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2004. p. 66-73.

Luckham, David; Frasca, Brian. *Complex event processing in distributed systems*. Stanford: Stanford University, 1998.

Luxemburgo. European Parliament. *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data*. Luxemburgo: Parlamento Europeu, 1995. Disponível em: https://eur-lex.europa.eu/LexUriServ/%20LexUriServ.do?uri=CELEX:31995L0046:en:HTML. Acesso em: 8 jan. 2022.

MAIDEN, Neil *et al*. A requirements-led approach for specifying qos-aware service choreographies: An experience report. *In*: INTERNATIONAL WORKING CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY, 20., 2014, Essen. *Proceedings* [...]. Berlim: Springer, 2014. p. 239-253.

MANI, Anbazhagan. *Understanding quality of service for Web services*. Armonk: IBM Developerworks, 2002.

MANVI, Sunilkumar; SHYAM, Gopal Krishna. Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey. *Journal of network and computer applications*, Amsterdã, v. 41, p. 424-440, 2014.

MAO, Zexiang *et al*. A game theory of cloud service deployment. *In*: IEEE WORLD CONGRESS ON SERVICES, 9., 2013, Santa Clara. *Proceedings* [...]. Washington, DC: IEEE, 2013. p. 436-443.

MARTIN, Patrick *et al.* QoS-aware cloud application management. *In*: CATLETT, Charlie. *et al.* (ed.). *Cloud computing and big data*. Amsterdã: IOS Press, 2013. v. 13, p. 20.

MASHUP. *In*: WIKIPÉDIA: a enciclopédia livre. 2022. Disponível em: https://pt.wikipedia.org/wiki/Mashup. Acesso em: 10 jan. 2022.

MASTERS, Gilbert. *Renewable and efficient electric power systems*. 2. ed. Hoboken: John Wiley & Sons, 2013.

MELL, Peter; GRANCE, Timothy. *NIST working definition of cloud computing*. Gaithersburg: NIST, 2011.

MELLO, Rodrigo; YANG, Laurence. Prediction of dynamical, nonlinear, and unstable process behavior. *The Journal of Supercomputing*, Nova York, v. 49, n. 1, p. 22-41, 2009.

MENDLING, Jan; LASSEN, Kristian; ZDUN, Uwe. Transformation strategies between block-oriented and graph-oriented process modelling languages. Berlim: GITO Verlag, 2006.

MENZEL, Michael; RANJAN, Rajiv. CloudGenius: decision support for web server cloud migration. *In*: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 21., 2012, Lyon. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2012. p. 979-988.

MENZEL, Michael; SCHÖNHERR, Marten; TAI, Stefan. (MC2)2: criteria, requirements and a software prototype for Cloud infrastructure decisions. *Software: Practice and Experience*, Nova Jersey, v. 43, n. 11, p. 1283-1297, 2013.

MESSAGE Sequence Chart. Genebra: ITU-T, 2011.

MICROSOFT. *Azure*. Redmond: Microsoft Azure, 2022a. Disponível em: https://azure.microsoft.com. Acesso em: 7 jan. 2022.

MICROSOFT. *Pricing calculator*. Redmond: Microsoft Azure, 2022b. Disponível em: https://azure.microsoft.com/en-us/pricing/calculator/. Acesso em: 7 jan. 2022.

MILCHTAICH, Igal. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, Amsterdã, v. 13, n. 1, p. 111-124, 1996.

MODACLOUDS. *MOdel-Driven Approach for design and execution of applications on multiple Clouds*. Modaclouds, 2023. Disponível em: https://cordis.europa.eu/project/id/318484. Acesso em: 18 mar. 2023.

MOSAIC CLOUD. *Home*. Mosaic Cloud, 2022. Disponível em: https://www.mosaic-cloud.eu. Acesso em: 8 jan. 2022.

MOSCATO, Francesco *et al*. An analysis of mosaic ontology for cloud resources annotation. *In*: FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS, 2011, Szczecin. *Proceedings* [...]. Washington, DF: IEEE, 2011. p. 973-980.

MOSER, Martin; JOKANOVIC, Dusan; SHIRATORI, Norio. An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, Tóquio, v. 80, n. 3, p. 582-589, 1997.

MOSTOW, Jack. Toward better models of the design process. *AI magazine*, Palo Alto, v. 6, n. 1, p. 44, 1985.

NEWMAN, Sam. *Building microservices*. 2. ed. Sebastopol: O'Reilly Media, 2021.

NGAN, Le Duy; KANAGASABAI, Rajaraman. OWL-S based semantic cloud service broker. *In*: INTERNATIONAL CONFERENCE ON WEB SERVICES, 19., 2012, Honolulu. *Proceedings [...]*. Washington, DC: IEEE, 2012. p. 560-567.

NIELSEN, Jacob. Powers of 10: Time Scales in User Experience. *NN/g*, Fremont, 4 out. 2009. Disponível em: https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/. Acesso em: 8 jan. 2022.

NOVAKOVIĆ, Dejan *et al*. Deepdive: Transparently identifying and managing performance interference in virtualized environments. *In*: ANNUAL TECHNICAL CONFERENCE, 2013, San Jose. *Proceedings* [...]. Berkeley: Usenix, 2013. p. 219-230.

OLOUGHLIN, John; GILLAM, Lee. Towards performance prediction for public infrastructure clouds: An EC2 case study. *In*: IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE, 5., 2013, Bristol. *Proceedings* [...]. Washington, DC: IEEE, 2014. p. 475-480.

O'Reilly, Tim. What Is Web 2.0. Design Patterns and Business Models for the next generation of software. *O'Reilly Media,* 30 set. 2005. Disponível em: https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html. Acesso em: 21 set. 2022.

Ostermann, Simon *et al*. A performance analysis of EC2 cloud computing services for scientific computing. *In*: International Conference on Cloud Computing, 2009, Munique. *Proceedings* [...]. Berlim: Springer, 2010. p. 115-131.

Ou, Zhonghong *et al*. Exploiting hardware heterogeneity within the same instance type of Amazon EC2. *In*: USENIX Workshop on Hot Topics in Cloud Computing, 4., 2012, Boston. *Proceedings* [...]. Berkeley: Usenix, 2012. p. 1-5.

Pahl, Claus. Containerization and the PaaS cloud. *IEEE Cloud Computing*, Washington, DC, v. 2, n. 3, p. 24-31, 2015.

Palma, Derek; Spatzier, Thomas (ed.). *Topology and orchestration specification for cloud applications version 1.0*. Woburn: Oasis, 2013.

Papadimitriou, Dimitri. Future Internet – the cross-ETP vision document. *Lecture Notes in Computer Science*, v. 7281, 2009.

Papazoglou, Michael *et al*. Service-oriented computing: State of the art and research challenges. *Computer*, Washington, DC, v. 40, n. 11, p. 38-45, 2007.

Parwekar, Pritee. From internet of things towards cloud of things. *In*: International Conference on Computer and Communication Technology, 2., 2011, Allahabad. *Proceedings* [...]. Washington, DC: IEEE, 2011. p. 329-333.

Pavlovski, Christopher; Zou, Joe. Non-Functional Requirements in Business Process Modeling. *In*: Asia-Pacific Conference on Conceptual Modelling, 5., 2008, Wollongong. *Proceedings* [...]. Darlinghurst: Australian Computer Society, 2010. p. 103-112.

Pelleg, Dan; Moore, Andrew. X-means: Extending k-means with efficient estimation of the number of clusters. *In*: International Conference on Machine Learning, 17., 2000, Stanford. *Proceedings [...]*. São Francisco: Morgan Kaufmann, 2000. p. 727-734.

Peltz, Chris. Web services orchestration and choreography. *Computer*, Washington, DC, v. 36, n. 10, p. 46-52, 2003.

Perforce. *Docs*: Marionette Collective. Portland: Puppet, 2011. Disponível em: https://docs.huihoo.com/puppet/mcollective/. Acesso em: 7 jan. 2022.

Perforce. *Puppet*: infrastructure automation & compliance at enterprise scale. 2023. Disponível em: https://www.puppet.com. Acesso em: 15 fev. 2023.

PETCU, Dana *et al*. Experiences in building a mOSAIC of clouds. *Journal of Cloud Computing: Advances, Systems and Applications*, Berlim, v. 2, n. 1, p. 1-22, 2013.

PHILLIPS, Stephen; ENGEN, Vegard; PAPAY, Juri.. Snow white clouds and the seven dwarfs. IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE, 3., 2011, Athens. *Proceedings* [...]. Washington, DC: IEEE, 2012. p. 738-745.

POULIN, Michael. Collaboration patterns in the SOA ecosystem. *In*: WORKSHOP ON BEHAVIOURAL MODELLING, 3., 2011, Birmingham. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2011. p. 12-16.

PROGRESS CHEF. *Powerful Policy-Based Configuration Management System Software*. Seattle: Progress Chef, 2022. Disponível em: https://www.chef.io/products/chef-infra. Acesso em: 7 jan. 2022.

QIU, Zongyan *et al*. Towards the theoretical foundation of choreography. *In*: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 16., 2007, Alberta. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2007. p. 973-982.

QUINTON, Clément; ROMERO, Daniel; DUCHIEN, Laurence. Automated selection and configuration of cloud environments using software product lines principles. *In*: INTERNATIONAL CONFERENCE ON CLOUD COMPUTING, 7., 2014, Anchorage. *Proceedings* [...]. Washington, DC: IEEE, 2014. p. 144-151.

RAHMAN, Mustafizur *et al*. A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and Computation*: Practice and Experience, Nova Jersey, v. 23, n. 16, p. 1990-2019, 2011.

RAY, Matt. *Spiceweasel*. 2022. Disponível em: https://github.com/mattray/spiceweasel. Acesso em: 7 jan. 2022.

REIG, Gemma; ALONSO, Javier; GUITART, Jordi. Prediction of job resource requirements for deadline schedulers to manage high-level SLAs on the cloud. *In*: IEEE INTERNATIONAL SYMPOSIUM ON NETWORK COMPUTING AND APPLICATIONS, 9., 2010, Cambridge. *Proceedings [...]*. Washington, DC: IEEE, 2010. p. 162-167.

RIGHTCLOUDZ. *Simplyfing Cloud Decisions*. RightCloudz, 2023. Disponível em: http://rightcloudz.com. Acesso em: 15 fev. 2023.

RHEE, Wansoo; TALAGRAND, Michel. Martingale inequalities and NP-complete problems. *Mathematics of Operations Research*, Catonsville, v. 12, n. 1, p. 177-181, 1987.

ROMAN, Gruia-Catalin. A taxonomy of current issues in requirements engineering. *Computer*, Washington, DC, v. 18, n. 4, p. 14-23, 1985.

Rosario, Sidney; Benveniste, Albert; Jard, Claude. Flexible probabilistic QoS management of transaction based web services orchestrations. *In*: ieee international conference on web services, 2009, Los Angeles. *Proceedings* [...]. Washington, dc: ieee, 2009. p. 107-114.

Rosenberg, Florian *et al*. Integrating quality of service aspects in top-down business process development using ws-cdl and ws-bpel. *In*: ieee international enterprise distributed object computing conference, 11., 2007, Annapolis. *Proceedings* [...]. Washington, dc: ieee, 2007. p. 15.

Saaty, Thomas. How to make a decision: the analytic hierarchy process. *European journal of operational research*, Amsterdã, v. 48, n. 1, p. 9-26, 1990.

Saaty, Thomas. Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales*, Madrid, v. 102, n. 2, p. 251-318, 2008.

Sahni, Sartaj. Approximate algorithms for the 0/1 knapsack problem. *Journal of the acm*, Nova York, v. 22, n. 1, p. 115-124, 1975.

Sandru, Calin; Venticinque, Salvatore. Agents layer to support cloud applications. *In*: fortino, Giancarlo *et al.* (ed.). *Intelligent distributed computing vi*. Berlim: Springer, 2013. p. 281-286.

Sandru, Calin; Petcu, Dana; Munteanu, Victor Ion. Building an open-source platform-as-a-service with intelligent management of multiple cloud resources. *In*: ieee international conference on utility and cloud computing, 5., 2012, Chicago. *Proceedings* [...]. Washington, dc: ieee, 2013. p. 333-338.

Sap. *sap Business ByDesign*. Newtown Square: sap, 2022. Disponível em: https://www.sap.com/products/business-bydesign.html. Acesso em: 7 jan. 2022.

Schad, Jörg; Dittrich, Jens; Quiané-ruiz, Jorge-Arnulfo. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the vldb Endowment*, Nova York, v. 3, n. 1-2, p. 460-471, 2010.

Shachter, Ross. Probabilistic inference and influence diagrams. *Operations research*, Catonsville, v. 36, n. 4, p. 589-604, 1988.

Singh, Ashish; Chatterjee, Kakali. Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, Amsterdã, v. 79, p. 88-115, 2017.

Singh, Sukhpal; Chana, Inderveer. A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of Grid Computing,* Dordrecht, v. 14, n. 2, p. 217-264, 2016.

Siqueira, Frank; Cahill, Vinny. Quartz: A QoS architecture for open systems. *In*: ieee international conference on distributed computing systems, 20., 2000, Taipei. *Proceedings* [...]. Washington, dc: ieee, 2000. p. 197-204.

Sirin, Evren; Hendler, James; Parsia, Bijan.. Semi-automatic composition of web services using semantic descriptions. *In*: workshop on web services: modeling, architecture and infrastructure, 1., 2003, Angers. *Proceedings [...]*. Setúbal: iceis, 2003. p. 17-24.

Soltani, Sima; Martin, Patrick; Elgazzar, Khalid. Quaram recommender: case-based reasoning for IaaS service selection. *In*: international conference on cloud and autonomic computing, 2014, Londres. *Proceedings* [...]. Washington, dc: ieee, 2014. p. 220-226.

Strauch, Steve *et al.* Non-functional data layer patterns for cloud applications. *In*: ieee international conference on cloud computing technology and science, 4., 2012, Taipei. *Proceedings* [...]. Washington, dc: ieee, 2012. p. 601-605.

Su, Jianwen *et al*. Towards a theory of web service choreographies. *In*: international workshop on web services and formal methods, 4., 2007, Brisbane. *Proceedings [...]*. Berlin: Springer, 2008. p. 1-16.

Suleiman, Basem *et al*. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *Journal of Internet Services and Applications*, Porto Alegre, v. 3, n. 2, p. 173-193, 2012.

Sun, Yih Leong *et al*. Mapping application requirements to cloud resources. *In*: european conference on parallel processing, 2012, Bordeaux. *Proceedings* [...]. Berlim: Springer, 2012. p. 104-112.

Sundareswaran, Smitha; Squicciarini, Anna; Lin, Dan.. A brokerage-based approach for cloud service selection. *In*: ieee international conference on cloud computing, 5., 2012, Honolulu. *Proceedings* [...]. Washington, df: ieee, 2012. p. 558-565.

Takacs, Lajos. On Erlang's formula. *The Annals of Mathematical Statistics*, Waite Hill, v. 40, n. 1, p. 71-78, 1969.

Torres, Romina; Bencomo, Nelly; Astudillo, Hernan. Mitigating the obsolescence of quality specifications models in service-based systems. *In*: international workshop on model-driven requirements engineering, *2., 2012, Chicago. Proceedings [...]*. Washington, dc: ieee, 2012. p. 68-76.

UML PROFILE for modeling and analysis of real time embedded systems. Needham: OMG, 2009.

VAQUERO, Luis *et al*. A break in the clouds: towards a cloud definition. *ACM Sigcomm Computer Communication Review*, Nova York, v. 39, n. 1, p. 50-55, 2008.

VAQUERO, Luis; RODERO-MERINO, Luis; MORÁN, Daniel. Locking the sky: a survey on IaaS cloud security. *Computing*, Berlim, v. 91, n. 1, p. 93-118, 2011.

VILAPLANA, Jordi *et al*. A queuing theory model for cloud computing. *The Journal of Supercomputing*, Berlim, v. 69, n. 1, p. 492-507, 2014.

VINCENT, Hugues *et al*. CHOREOS: scaling choreographies for the internet of the future. *In*: MIDDLEWARE'10 POSTERS AND DEMOS TRACK, 11., 2010, Bangalore. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2010. p. 1-3.

VINCENTY, Tyrone. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, Bristol, v. 23, n. 176, p. 88-93, 1975.

VISUALVM. *All-in-One Java Troubleshooting Tool*. VisualVM, 2022. Disponível em: https://visualvm.github.io. Acesso em: 10 jan. 2022.

WANG, Hongbing *et al*. Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition. *Journal of Parallel and Distributed Computing*, Amsterdã, v. 100, p. 71-84, 2017.

WANG, Minhong *et al*. On-demand e-supply chain integration: A multi-agent constraint-based approach. *Expert Systems with Applications*, Amsterdã, v. 34, n. 4, p. 2683-2692, 2008.

WIEDER, Alexander *et al*. Conductor: orchestrating the clouds. *In*: INTERNATIONAL WORKSHOP ON LARGE SCALE DISTRIBUTED SYSTEMS AND MIDDLEWARE, 4., 2010, Zurique. *Proceedings* [...]. Nova York: Association for Computing Machinery, 2010. p. 44-48.

WOLF, Karsten. Does my service have partners? *In*: JENSEN, Kurt; AALST, Wil. (ed.). *Transactions on Petri Nets and other models of concurrency II*. Lecture Notes in Computer Science. Berlim: Springer, 2009. v. 5460, p. 152-171.

WRIGHT, Peter *et al*. A constraints-based resource discovery model for multi-provider cloud environments. *Journal of Cloud Computing: Advances, Systems and Applications*, Nova York, v. 1, n. 1, p. 1-14, 2012.

WU, Linlin; GARG, Saurabh Kumar; BUYYA, Rajkumar. SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments. *In*: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER, CLOUD AND GRID COMPUTING, 11., 2011, Newport Beach. *Proceedings* [...]. Washington, DC: IEEE, 2011. p. 195-204.

Wu, Quanwang *et al*. Broker-based SLA-aware composite service provisioning. *Journal of Systems and Software*, Amsterdã, v. 96, p. 194-201, 2014.

Xiong, Kaiqi; Perros, Harry. Service performance and analysis in cloud computing. *In*: congress on services-I, 2009, Los Angeles. *Proceedings* [...]. Washington, DC: IEEE, 2009. p. 693-700.

Xu, Fei *et al*. Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. *Proceedings of the IEEE*, Washington, DC, v. 102, n. 1, p. 11-31, 2013.

Yang, Lingyun; Foster, Ian; Schopf, Jennifer. Homeostatic and tendency-based CPU load predictions. *In*: international parallel and distributed processing symposium, 2003, Nice. *Proceedings* [...]. Washington, DC: IEEE, 2003, p. 9.

Ye, Zhen; Bouguettaya, Athman; Zhou, Xiaofang. QoS-aware cloud service composition based on economic models. *In*: international conference on service-oriented computing, 10., 2012, Xangai. *Proceedings [...]*. Berlin: Springer, 2012. p. 111-126.

Ye, Zhen; Zhou, Xiaofang; Bouguettaya, Athman. Genetic algorithm based QoS-aware service compositions in cloud computing. *In*: international conference on database systems for advanced applications, 16., 2011, Hong Kong. *Proceedings* [...]. Berlim: Springer, 2011. p. 321-334.

Yu, Jia; Buyya, Rajkumar; Tham, Chen Khong. QoS-based scheduling of workflow applications on service grids. *In*: IEEE international conference on E-science and grid computing, 1., 2005, Melbourne. *Proceedings [...]*Washington, DC: IEEE, 2005. p. 5-8.

Yu, Jia; Buyya, Rajkumar; Tham, Chen Khong.. Cost-based scheduling of scientific workflow applications on utility grids. *In*: international conference on E-science and grid computing, 1., 2005, Melbourne. *Proceedings* [...]. Washington, DC: IEEE, 2006. p. 8.

Yu, Tao; Zhang, Yhang; Lin, Kwei-Jay. Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Transactions on the Web*, Nova York, v. 1, n. 1, p. 6, 2007.

Zaha, Johannes Maria *et al*.. Service interaction modeling: Bridging global and local views. *In*: IEEE international enterprise distributed object computing conference, 10., 2006, Hong Kong. *Proceedings* [...]. Washington, DC: IEEE, 2006a. p. 45-55.

Zaha, Johannes Maria *et al*. Let's dance: A language for service behavior modeling. *In*: otm confederated international conferences, 2006, Montpellier. *Proceedings* [...]. Berlim: Springer, 2006b. p. 145-162.

Zeleny, Milan (ed.). *Multiple criteria decision making Kyoto 1975*. Berlim: Springer Science & Business Media, v. 123, 2012.

Zhang, Qi; Cheng, Lu; Boutaba, Raouf. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, Porto Alegre, v. 1, p. 7-18, 2010.

Zou, Joe; Pavlovski, Christopher. Modeling architectural non functional requirements: from use case to control case. *In*: ieee international conference on e-business engineering, 2006, Xangai. *Proceedings* [...]. Washington, dc: ieee, 2006. p. 315-322.

Créditos

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

Créditos

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

Formato 160 x 230mm

Tipografia Myriad Pro Bold 12/18 (títulos)
Chaparral Pro 12/18 (texto)

Imagem da Capa
Cloud Computing Key
de Tarik Kizilkaya no Canva

Raphael de Aquino Gomes é doutor em Ciência da Computação pela Universidade Federal de Goiás, com período sanduíche no Institut National de Recherche en Informatique et en Automatique, França. Atua como professor efetivo do Instituto Federal de Goiás/Câmpu Goiânia, onde coordena o Mestrado Profissional em Tecnologia, Gestão e Sustentabilidade. Seus interesses de pesquisa se detêm em sistemas distribuídos, com foco em computação em nuvem, IoT, sistemas adaptativos, inteligência artificial e energias renováveis.

Esta obra surge quando a computação em nuvem e a arquitetura orientada a serviços se consolidam como pilares fundamentais para o desenvolvimento de sistemas distribuídos modernos. Aborda, de forma competente, a complexidade da implantação de coreografias de serviços em ambientes de nuvem, propondo soluções para desafios que vão desde a modelagem de serviços até a alocação eficiente de recursos, consideradas as restrições não funcionais e a heterogeneidade de provedores de nuvem.

Esta publicação tem potencial para repercutir a pesquisa por ela veiculada, habilitando cenários do tipo ganha-ganha entre provedores de nuvem e seus clientes. Além disso, abre uma série de perspectivas de estudo, como a exploração da inter-relação com a alocação elástica de recursos, característica da computação em nuvem, e a integração com sistemas de gerenciamento e alocação de recursos em nuvem.

ISBN 978 65 83687 01 2
9 786583 687012

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Goiás

editora ifg

Associação Brasileira
das Editoras Universitárias