UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

BRUNO WINIEMKO VOLLINO

Descoberta de perfis de uso de web services

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profª. Drª. Karin Becker
Orientadora

Porto Alegre, novembro de 2013

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Vollino, Bruno Winiemko

Descoberta de perfis de uso de web services / Bruno Winiemko Vollino. – Porto Alegre: PPGC da UFRGS, 2013.

79 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Orientadora: Karin Becker.

1. Serviço web. 2. Mineração de dados. 3. Padrões de uso. 4. Perfis de uso. I. Becker, Karin. II. Título.

AGRADECIMENTOS

Agradeço à professora Karin, pela orientação, conselhos e paciência.

Aos demais professores com quem tive a oportunidade de aprender, pelas fundações necessárias para completar esta etapa.

À minha família, sem a qual não teria perseverado. Aos amigos do laboratório, pelo bom humor e discussões no RU.

Agradeço também ao PPGC da UFRGS, pela oportunidade de cursar o mestrado, e ao CNPq, pela bolsa e auxílio financeiro.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

BPEL	Business Process Evolution Language
HTTP	Hypertext Transfer Protocol
KDD	Knowledge Discovery in Databases
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language
WSDL	Web Services Description Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Durante o ciclo de vida de um web service, diversas mudanças são feitas na sua interface, eventualmente causando incompatibilidades em relação aos seus clientes e ocasionando a quebra de suas aplicações. Os provedores precisam tomar decisões sobre mudanças em seus serviços frequentemente, muitas vezes sem um bom entendimento a respeito do efeito destas mudanças sobre seus clientes. Os trabalhos e ferramentas existentes não fornecem ao provedor um conhecimento adequado a respeito do uso real das funcionalidades da interface de um serviço, considerando os diferentes tipos de consumidores, o que impossibilita avaliar o impacto das mudanças.

Este trabalho apresenta um framework para a descoberta de perfis de uso de serviços web, os quais constituem um modelo descritivo dos padrões de uso dos diferentes grupos de clientes do serviço, com relação ao uso das funcionalidades em sua interface. O framework auxilia no processo de descoberta de conhecimento através de tarefas semi-automáticas e parametrizáveis para a preparação e análise de dados de uso, minimizando a necessidade de intervenção do usuário.

O framework engloba o monitoramento de interações de web services, a carga de dados de uso pré-processados em uma base de dados unificada, e a geração de perfis de uso. Técnicas de mineração de dados são utilizadas para agrupar clientes de acordo com seus padrões de uso de funcionalidades, e esses grupos são utilizados na construção de perfis de uso de serviços. Todo o processo é configurado através de parâmetros, permitindo que o usuário determine o nível de detalhe das informações sobre o uso incluídas nos perfis e os critérios para avaliar a similaridade entre clientes.

A proposta é validada por meio de experimentos com dados sintéticos, simulados de acordo com características esperadas no comportamento de clientes de um serviço real. Os resultados dos experimentos demonstram que o framework proposto permite a descoberta de perfis de uso de serviço úteis, e fornecem evidências a respeito da parametrização adequada do framework.

Palavras-chave: Serviço web, mineração de dados, padrões de uso, perfis de uso.

Web services usage profiles discovery

ABSTRACT

During the life cycle of a web service, several changes are made in its interface, which possibly are incompatible with regard to current usage and may break client applications. Providers must make decisions about changes on their services, most often without insight on the effect these changes will have over their customers. Existing research and tools fail to input provider with proper knowledge about the actual usage of the service interface's features, considering the distinct types of customers, making it impossible to assess the actual impact of changes.

This work presents a framework for the discovery of web service usage profiles, which constitute a descriptive model of the usage patterns found in distinct groups of clients, concerning the usage of service interface features. The framework supports a user in the process of knowledge discovery over service usage data through semi-automatic and configurable tasks, which assist the preparation and analysis of usage data with the minimum user intervention possible. The framework performs the monitoring of web services interactions, loads pre-processed usage data into a unified database, and supports the generation of usage profiles. Data mining techniques are used to group clients according to their usage patterns of features, and these groups are used to build service usage profiles. The entire process is configured via parameters, which allows the user to determine the level of detail of the usage information included in the profiles, and the criteria for evaluating the similarity between client applications.

The proposal is validated through experiments with synthetic data, simulated according to features expected in the use of a real service. The experimental results demonstrate that the proposed framework allows the discovery of useful service usage profiles, and provide evidences about the proper parameterization of the framework.

Keywords: web service, data mining, usage patterns, usage profiles.

1 INTRODUÇÃO

Com o surgimento e popularização de novos modelos de distribuição de software e infraestrutura, como SaaS (*Software as a Service*), IaaS (*Infrastructure as a Service*) e PaaS (*Platform as a Service*), os web services se tornaram peça fundamental de muitas empresas, e sua evolução, uma questão estratégica para seus negócios. Diversas mudanças podem ocorrer em um web service durante seu ciclo de vida, para atender demandas dos consumidores, por questões de otimização ou contratuais,

Mudanças na interface do serviço (LEITNER et al., 2008), como adição/remoção de operações e alterações em parâmetros, ocorrem periodicamente em web services providos em larga escala, devido ao grande número de clientes e suas variadas demandas. As mudanças na interface de um serviço podem torná-lo incompatível com as aplicações cliente já existentes, o que impacta negativamente nos negócios da empresa.

Apesar de ser inevitável que mudanças ocorram, o provedor precisa, de alguma forma, ter ciência do impacto que uma mudança específica causa aos seus clientes, em termos dos custos de atualização de suas aplicações e dos custos potenciais para seu próprio negócio, decorrentes da quebra de consumidores importantes (FOKAEFS et al., 2011). O conhecimento sobre esse impacto permite que o provedor adote estratégias quanto à implementação e implantação de mudanças.

A compatibilidade de mudanças é tradicionalmente avaliada de acordo com a premissa do pior caso, no qual toda mudança incompatível entre duas versões de um serviço também é considerada incompatível em relação a todos os seus clientes (BECKER et al., 2008; FOKAEFS et al., 2011; ANDRIKOPOULOS; BENBERNOU; PAPAZOGLOU, 2011). Por exemplo, assume-se que a remoção de uma operação afeta todos os clientes do serviço, desconsiderando-se se a operação é ou não utilizada por todos eles.

Constata-se, no entanto, que serviços providos em larga escala frequentemente violam a premissa de necessidade de compatibilidade total entre clientes e interface, pois possuem uma grande variedade de clientes, os quais podem utilizar subconjuntos das funcionalidades disponíveis.

Mesmo que tomem decisões sobre mudanças na interface do serviço com frequência, os provedores podem desconhecer quem são e quantos são seus clientes, e com que frequência usam seus serviços (FOKAEFS et al., 2011). Entender de que formas os clientes utilizam os serviços é de valor inestimável no apoio à gestão do ciclo de vida dos web services (PAPAZOGLOU; ANDRIKOPOULOS; BENBERNOU, 2011). Assim, os provedores podem avaliar com clareza o impacto real de mudanças.

O framework WS-Evolv foi proposto para suprir esta lacuna, o qual tem por objetivos identificar e representar o uso real feito por clientes de web services, e, com base nesse uso, mensurar o impacto das alterações de interface dos serviços (YAMASHITA; BECKER; GALANTE, 2011; YAMASHITA et al., 2012; SILVA et al., 2012; VOLLINO;

BECKER, 2013a, 2013b; SILVA; BECKER; GALANTE, 2013). Um dos conceitos fundamentais do WS-Evolv é o de *perfil de uso* de web services, que são representações quantificadas de grupos de clientes que possuem padrões de consumo similares em relação às funcionalidades descritas na interface do serviço.

O WS-Evolv é composto por três módulos, responsáveis por gerenciar o versionamento, identificar e manter os perfis de uso e analisar os efeitos das mudanças no contexto do uso do serviço, como ilustrado na Figura 1.1.

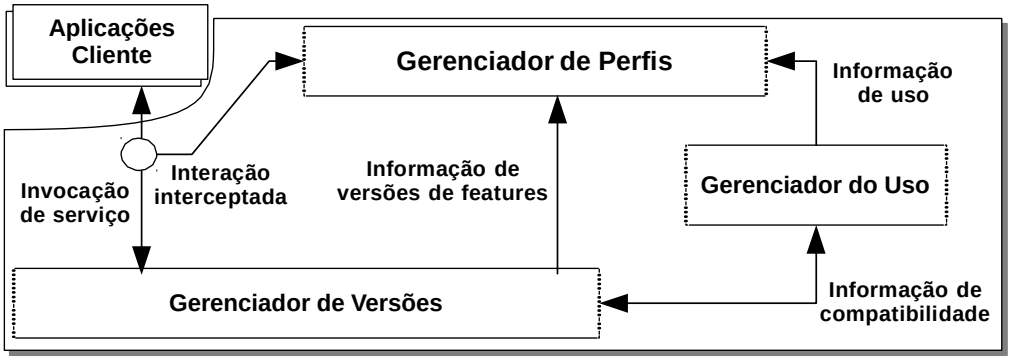


Figura 1.1: Framework de gerência da evolução de serviços.

O *Gerenciador de Versões* (YAMASHITA; BECKER; GALANTE, 2012) é responsável pelo versionamento do serviço e pela análise de compatibilidade entre versões. Ele mantém um repositório de descrições da interface de serviços versionadas, utilizando um modelo de versionamento em granularidade fina, no qual são versionadas somente as partes específicas do serviço que sofreram alterações. São denominadas *features* as partes versionáveis da interface de um serviço, como suas operações e tipos. O modelo de versionamento baseado em features permite identificar, entre duas versões de um serviço, quais são as mudanças específicas realizadas na sua interface.

O *Gerenciador de Perfis de Uso*, objeto do presente trabalho, é responsável por capturar, transformar e armazenar dados brutos de uso de serviços, e a partir deles, agrupar clientes que possuem padrões de uso semelhantes. Esses grupos são então representados através de perfis de uso.

O *Gerenciador do Uso* (SILVA; BECKER; GALANTE, 2013) reúne um conjunto de aplicações para avaliar o impacto de mudanças, através da análise de dados de uso, perfis de uso e dados de versionamento. Entre as aplicações já desenvolvidas estão (a) a quantificação do impacto de mudanças incompatíveis (YAMASHITA et al., 2012), (b) a avaliação de compatibilidade orientada ao uso (VOLLINO; BECKER, 2013b), e (c) um ambiente de BI que possibilita avaliar e correlacionar o impacto segundo diferentes perspectivas (uso, financeira) (SILVA et al., 2012; SILVA; BECKER; GALANTE, 2013).

A proposta inicial do WS-Evolv (YAMASHITA; BECKER; GALANTE, 2011) sugere a utilização de técnicas de mineração de dados para agrupar aplicações clientes, a fim de extrair perfis de uso, porém não detalha o processo de extração da informação.

1.1 Problema

Este trabalho trata o problema de como obter perfis de uso de serviço úteis, de serviços que possuem grande número de funcionalidades e que são utilizados em larga escala.

A hipótese levantada é de que é possível descobrir perfis de uso úteis através do agrupamento de clientes de acordo com o uso das funcionalidades do serviço, a partir da

análise de interações entre o provedor e seus clientes.

1.2 Objetivo

O objetivo deste trabalho é conceber um framework para auxiliar na descoberta de perfis de uso de serviços em meio a grandes volumes de dados. Esse framework pode ser utilizado para desenvolver o módulo Gerenciador de Perfis do WS-Evolv.

Para compor o framework faz-se necessário:

- um estudo de técnicas de monitoramento de web services, a fim de verificar os dados de uso disponíveis através de cada uma delas e a utilidade desses dados para a identificação de grupos de aplicações;
- definir como extrair e armazenar dados de uso, de forma a viabilizar o processo de descoberta de padrões de uso e agrupamento de clientes;
- definir como devem ser realizados os passos do processo para a descoberta de perfis de uso potencialmente úteis.

1.3 Contribuição

As contribuições do trabalho se encontram na definição do processo de descoberta de conhecimento através do qual os perfis de uso de web services são construídos, a partir de registros de requisições de clientes a serviços, e no framework que implementa esse processo de forma semi-automática. Toma-se por base o processo de descoberta de conhecimento em bases de dados (KDD - *Knowledge Discovery in Databases*) (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996), porém o framework proposto agrega valor ao processo através da semi-automatização de tarefas de preparação de dados e aplicação de técnicas de mineração de dados, de forma a minimizar a necessidade de intervenção do usuário. Os resultados desta pesquisa foram apresentados à comunidade científica por meio de publicações realizadas durante seu desenvolvimento (YAMASHITA et al., 2012; SILVA et al., 2012; VOLLINO; BECKER, 2013a, 2013b).

O framework engloba componentes para: (a) monitoramento de requisições de clientes, (b) processamento de logs de requisição e armazenamento em uma base de dados de uso de propósito geral, e (c) preparação de dados de uso, clustering de clientes de acordo com seus padrões de consumo e transformação dos grupos em perfis de uso.

Padrões de uso de serviços são formas específicas pelas quais os web services ou suas operações são utilizadas repetidamente por um grupo de usuários com propriedades similares (LIANG et al., 2006). Este trabalho contribui com técnicas para a identificação de padrões de uso ainda não explorados por trabalhos existentes em mineração de serviços, especificamente grupos de clientes baseados no uso de features.

A análise do uso tem sido usada para apoiar a recomendação de serviços (RONG; LIU; LIANG, 2009; LO et al., 2012; ZHANG et al., 2012; KANG et al., 2012; ZHANG; DING; CHI, 2011; YU, 2012), recomendação de composições de serviços (LIANG; CHUNG, 2007; ZHANG et al., 2009) e descoberta de processos de negócio (TANG; ZOU, 2010) (MOTAHARI-NEZHAD et al., 2011). Estas abordagens, em geral, lidam com o uso em nível de serviços, e identificam padrões de uso através de modelos de conhecimento como workflows de serviços ou operações, regras de associação de serviços ou matrizes de similaridade entre clientes que usam ou buscam os mesmos serviços.

O presente trabalho diferencia-se dos demais por apresentar um processo para não apenas identificar padrões de uso, mas também associá-los aos respectivos clientes do serviço, junto com quantificações desta relação. A identificação de padrões de uso, sem a relação dos clientes representados e sem a quantificação do uso, impossibilitam uma avaliação precisa do impacto externo de mudanças, pois tal impacto depende da importância dos clientes para a organização, no qual a frequência de requisições pode ser um fator influenciador.

Os padrões descobertos também apresentam um detalhamento maior em relação aos identificados pelos demais trabalhos, pois agrupam aplicações cliente de acordo com as operações e tipos de dados utilizados. A identificação de padrões de uso dos tipos é importante pois pequenas mudanças na interface podem se propagar por outras partes do serviço, como tipos compostos e operações, causando um grande e inesperado impacto (YAMASHITA et al., 2012). Desta forma, uma avaliação apropriada do impacto de mudanças pode ser atingida através de perfis de uso (SILVA et al., 2012), o que não é possível através da avaliação de compatibilidade e impacto baseada somente em versões do serviço.

A abordagem implementa o Gerenciador de Perfis do framework WS-Evolv, viabilizando sua utilização através de várias aplicações já propostas (YAMASHITA et al., 2012; SILVA et al., 2012; SILVA; BECKER; GALANTE, 2013; VOLLINO; BECKER, 2013b). Contudo o framework proposto também habilita a construção de vários outros mecanismos de análise com base em perfis de uso, visando, por exemplo, o redesign de serviços e processos, a manutenção de provedores implantados (otimização de serviços, redirecionamento de clientes, balanceamento de carga) e a recomendação de serviços.

1.4 Organização do trabalho

O trabalho está estruturado da sequinte forma. O Capítulo 2 apresenta a fundamentação teórica. O Capítulo 3 discute os trabalhos relacionados. O Capítulo 4 apresenta o framework para descoberta de perfis de uso, representado pelos componentes do Gerenciador de Perfis. O processo de KDD proposto para a geração dos perfis de uso é detalhado no Capítulo 5. Os experimentos realizados sobre o protótipo do framework e seus resultados são discutidos no Capítulo 6. O Capítulo 7 apresenta conclusões e discute trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são introduzidos os conceitos necessários ao entendimento do trabalho. São apresentados os conceitos e tecnologias utilizadas em web services, o ciclo de evolução de serviços, gestão de mudanças e o versionamento de serviços. Esses conceitos servem como base para compreender o que são perfis de uso e quais suas aplicações. Também é dada uma visão geral do processo de descoberta de conhecimento em bases de dados e de mineração de dados, especialmente no que se refere à tarefa de agrupamento, utilizada na descoberta dos perfis. Por fim, são discutidas alternativas para o monitoramento de serviços, a fim de entender como obter os dados de uso necessários à criação de perfis.

2.1 Web services

Web services (serviços web) são componentes de software utilizados para interoperar sistemas heterogêneos em uma rede de computadores. O agente que implementa o web service é denominado *provedor*, e aquele que o utiliza é denominado *cliente*, enquanto denomina-se *consumidor* a entidade que possui o cliente, e *provedor* a entidade que possui o web service (BOOTH et al., 2004).

Web services possibilitam um baixo acoplamento entre os agentes que participam da comunicação, devido a duas características: a independência entre interface e implementação do serviço, e a comunicação através de troca de mensagens.

As interações entre agentes se dão através de uma interface definida pelo provedor, a qual abstrai a estrutura de software e hardware do web service. Ao publicar um serviço, o provedor disponibiliza uma descrição dessa interface, na qual são definidas as funcionalidades disponibilizadas pelo web service.

O W3C (*World Wide Web Consortium*), em sua arquitetura de web service (BOOTH et al., 2004), determina que a interface do serviço seja descrita em um formato processável por máquina, como, por exemplo, o WSDL (*Web Service Description Language*) (CHINNICI et al., 2007), e a utilização do formato SOAP (*Simple Object Access Protocol*) (W3C, 2007) para as mensagens. Esse modelo arquitetural, ao abstrair a interface da implementação do serviço, também permite que o provedor acrescente camadas de serviços de forma transparente ao cliente, como segurança, autenticação, processos de composição ou descoberta de serviços, por exemplo. A Figura 2.1 ilustra a disposição das camadas na arquitetura baseada em SOAP e WSDL, o que torna esse tipo de web service conhecido como *serviço SOAP*.

WSDL é uma linguagem utilizada para descrever a interface de web services, baseada em XML (*eXtensible Markup Language*) (BRAY et al., 2008), a qual possibilita definir em um documento quais as *operações* expostas e os *tipos* utilizados pelo serviço. Os tipos

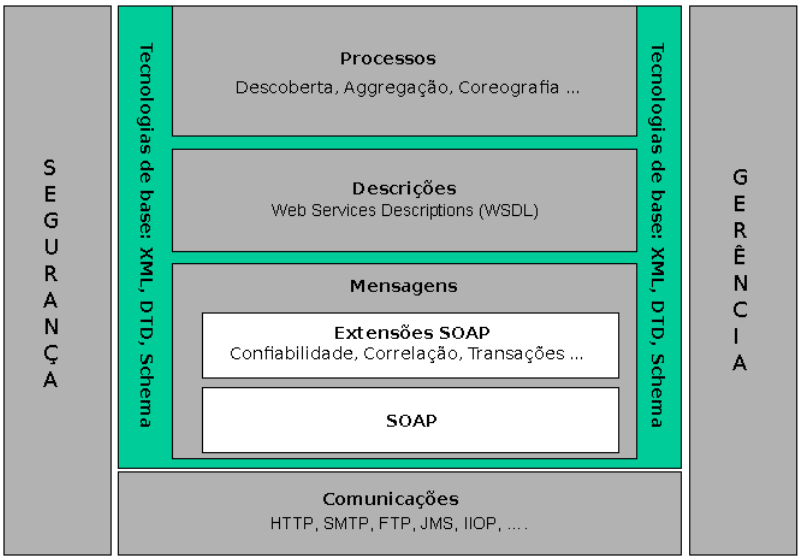


Figura 2.1: Arquitetura de um serviço SOAP (BOOTH et al., 2004).

podem ser básicos, como strings, numerais ou enumerações, ou complexos, os quais são compostos outros tipos. Cada operação possui um conjunto de parâmetros, os quais estão associados aos tipos definidos na interface do serviço. Os parâmetros referentes a tipos compostos também possuem seus parâmetros, o que resulta em uma estrutura hierárquica de parâmetros e tipos para cada operação. Cada operação é mapeada para uma URL distinta, através da qual o protocolo HTTP é utilizado para transmissão de mensagens. Os parâmetros da requisição podem ser incorporados a URL da operação, o que é comum em serviços RESTful (PAUTASSO; ZIMMERMANN; LEYMANN, 2008; FIELDING, 2000), ou submetidos através de uma mensagem, que é a forma padrão em serviços SOAP.

Uma mensagem SOAP é um documento XML que informa ao provedor a operação requisitada pelo cliente e os parâmetros utilizados na requisição. A forma como o provedor identifica a operação, parâmetros e tipos dos parâmetros depende do estilo de mensagem SOAP requerido pelo provedor, informado no WSDL do web service, o que se denomina *encoding* da mensagem. As informações enviadas nas mensagens variam de acordo com o estilo de *encoding* utilizado, que pode ser:

- Estilo RPC/encoded: o nome da operação é enviado no corpo da mensagem, assim como o nome de cada parâmetro e o nome de seus tipos.
- Estilo RPC/literal e Document/literal wrapped: o nome da operação é enviado no corpo da mensagem, assim como o nome de cada parâmetro, omitindo o nome dos tipos.
- Estilo Document/literal: os nomes dos parâmetros são enviados, porém o nome da operação e dos tipos são omitidos.

As mensagens são criadas através de um processo conhecido como *marshalling*, no qual os dados em memória de um agente são transformados em documentos XML, e decodificados pelo processo de *unmarshalling*, no qual os documentos são interpretados e transformados em dados em memória.

Neste trabalho, assumimos o uso de serviços SOAP e mensagens com encoding Document/literal. Estas premissas são importantes, pois delas dependem a forma como é

feito o monitoramento do web service e a identificação de features durante a extração de dados de uso, a partir das mensagens capturadas.

2.2 Evolução de serviços

Como é inerente a qualquer componente de software, os web services possuem um ciclo evolutivo, no qual são submetidos a constantes modificações. Em um ciclo de vida orientado a mudanças (PAPAZOGLOU; ANDRIKOPOULOS; BENBERNOU, 2011) são tratadas tanto mudanças funcionais como não funcionais do serviço.

As mudanças funcionais são aquelas que alteram a interface do serviço, o que inclui a alteração, adição e remoção de operações, parâmetros e tipos. Mudanças não funcionais são aquelas que não afetam a interface, como mudanças relacionadas à qualidade do serviço e aos contratos de uso.

Se as mudanças funcionais realizadas forem compatíveis com uma versão anterior do serviço, são denominadas *retrocompatíveis* em relação àquela versão. A adição de operações é um exemplo de mudança retrocompatível, pois não afeta os clientes atuais do serviço. É importante destacar que as mudanças podem se propagar através da interface do serviço, para partes que não sofreram alterações diretamente, mas que dependem das partes alteradas.

Quando uma nova versão do serviço possui apenas mudanças retrocompatíveis, a versão anterior pode ser substituída pela nova sem que os clientes sejam afetados. No entanto, a maior parte das mudanças funcionais são *incompatíveis* e costumam ser evitadas (PAPAZOGLOU; ANDRIKOPOULOS; BENBERNOU, 2011), pois causam a quebra de clientes do serviço.

Para gerenciar mudanças incompatíveis, os provedores adotam medidas como o versionamento do serviço e políticas de redirecionamento de requisições, a fim de tornar as mudanças transparentes aos clientes (LEITNER et al., 2008), o que também implica em um aumento nos custos de manutenção do serviço.

O aumento dos custos é consonante com o aumento do número de versões concomitantes de um serviço, o que torna insustentável manter a transparência de todas as mudanças. No momento em que uma versão antiga precisa ser descontinuada, as mudanças podem afetar os clientes que ainda a utilizam diretamente. As mudanças também podem afetar clientes indiretos, pois seu efeito é propagado para serviços que dependem do serviço modificado (PAPAZOGLOU; ANDRIKOPOULOS; BENBERNOU, 2011; SILVA et al., 2012).

As mudanças incompatíveis, tanto diretas quanto propagadas, nem sempre são documentadas de forma adequada pelo provedor (YAMASHITA et al., 2012). A identificação de mudanças e avaliação de seu impacto sobre os clientes diretos e indiretos de um serviço ainda são tarefas de difícil execução durante a evolução de serviços, o que dificulta a tomada de decisão quanto a realização de mudanças.

Neste trabalho considera-se que cada serviço pode possuir uma ou mais versões concorrentes, o que implica em lidar com o uso de múltiplas versões de um serviço durante o processo de descoberta de perfis.

2.3 Versionamento de serviços

O versionamento é uma das ferramentas utilizadas para gerenciar as mudanças da interface de um serviço durante seu ciclo de evolução, especialmente mudanças que geram

incompatibilidades em relação aos clientes.

O versionamento de serviços geralmente é feito de forma monolítica: uma nova versão do serviço é criada a cada alteração incompatível realizada em sua interface. Essa abordagem assume uma compatibilidade total entre o serviço e cada cliente. Quando uma nova versão é retrocompatível, não há necessidade de versionar, e a nova versão pode substituir uma versão antiga. Quando a nova versão é incompatível, a versão anterior é descontinuada, ou as versões podem ser mantidas em paralelo.

Uma mudança em uma parte da interface de um serviço pode afetar outras partes da interface, que não foram alteradas, mas que dependem daquelas que foram; as mudanças também podem se propagar através de outros serviços do portfólio de uma organização (PAPAZOGLOU; ANDRIKOPOULOS; BENBERNOU, 2011), afetando clientes indiretos do serviço (SILVA et al., 2012). O problema do versionamento monolítico é justamente a dificuldade em se identificar o que foi alterado e quais partes da interface são tornadas incompatíveis pela propagação das mudanças (YAMASHITA; BECKER; GALANTE, 2012; ZOU et al., 2008).

Devido à abordagem de versionamento monolítica, o provedor acaba, muitas vezes, não identificando corretamente os pontos da interface afetados pelas mudanças. Isso pode ocorrer com probabilidade ainda maior em serviços com um grande número de operações e tipos, pois se torna muito difícil e custoso verificar com precisão a extensão e propagação das mudanças. Com isso, é frequente que mudanças importantes não sejam documentadas, causando a quebra inesperada de clientes que não foram alertados corretamente (YAMASHITA et al., 2012).

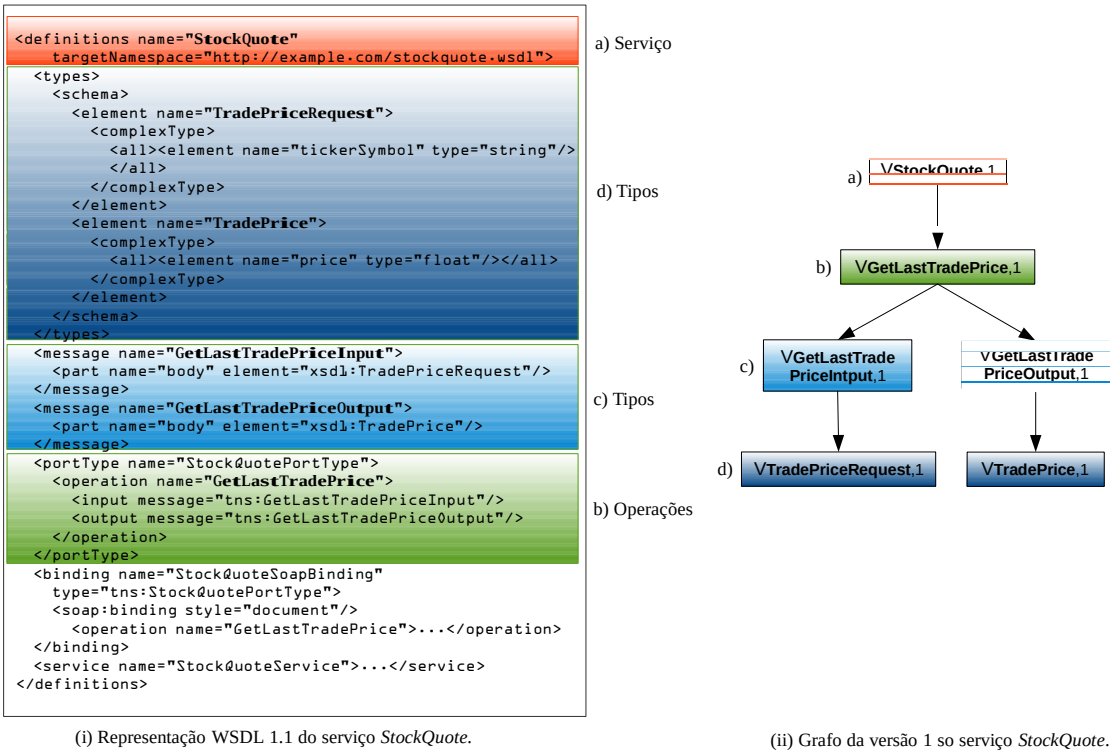
O versionamento na granularidade de *features* foi proposto com o objetivo de facilitar a identificação de uma mudança e dos seus efeitos (YAMASHITA; BECKER; GALANTE, 2012; YAMASHITA et al., 2012).

Esse modelo representa a interface de um serviço através de um conjunto inter-relacionado de versões, as quais representam partes específicas de um serviço: as operações, os tipos complexos e o serviço como um todo. Essas partes da interface descrevem as funcionalidade de um serviço (FANG et al., 2007), motivo pelo qual estes conceitos (serviço, operação e tipo) são denominados *features* de um serviço (LEE; KANG; LEE, 2002).

Tome-se por exemplo o mapeamento de uma descrição WSDL para um conjunto de versões de features ilustrado na Figura 2.2. Nesse exemplo, (a) o serviço 'StockQuote' é identificado na descrição da interface e mapeado para a versão de feature 'StockQuote,1'; (b) a versão de feature 'GetLastTradePrice,1' é criada para representar a operação 'GetLastTradePrice', e uma relação de dependência é estabelecida entre o serviço e a operação. A dependência indica que alterações feitas em 'GetLastTradePrice,1' afetarão também 'StockQuote,1'; (c) as mensagens da operação 'GetLastTradePrice' são mapeadas para os tipos 'GetLastTradePriceInput,1' e 'GetLastTradePriceOutput,1', dos quais a versão da operação depende; (d) os tipos dos parâmetros utilizados nas mensagens, 'TradePriceRequest' e 'TradePrice' são adicionados como dependências às features correspondentes.

O modelo de versionamento é descrito na Figura 2.3 usando um diagrama de classes UML. Uma feature possui uma ou mais versões, as quais podem depender de uma ou mais versões de outras features. Assim, uma versão de serviço é representada por um grafo de versões de features relacionadas.

Quando uma feature é alterada, são criadas novas versões para a feature em questão, bem como para todas as demais features que dependem dela. Utiliza-se o fragmento da descrição do serviço que corresponde à feature para identificar se houve alguma alteração




```
<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl">
  <types>
    <schema>
      <element name="TradePriceRequest">
        <complexType>
          <all><element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all><element name="price" type="float"/></all>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>
  <binding name="StockQuoteSoapBinding"
    type="tns:StockQuotePortType">
    <soap:binding style="document"/>
      <operation name="GetLastTradePrice">...</operation>
  </binding>
  <service name="StockQuoteService">...</service>
</definitions>
```

(i) Representação WSDL 1.1 do serviço *StockQuote*.

(ii) Grafo da versão 1 so serviço *StockQuote*.

Figura 2.2: Versionamento de features de um WSDL

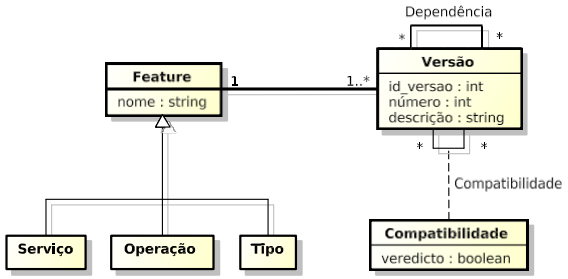


Figura 2.3: Representação UML do modelo de versionamento orientado a features.

em relação às versões anteriores, assim como sua compatibilidade.

Por exemplo, considerando a Figura 2.2, caso um novo documento WSDL do serviço StockQuote fosse publicado com uma alteração no tipo TradePrice, seria criada uma nova versão para esta feature (TradePrice,2), e para todas as features que dependem dela, direta (GetLastTradePriceOutput,2) ou indiretamente (GetLastTradePrice,2; StockQuote,2). A Figura 2.4 mostra o grafo de versões resultante da criação destas novas versões de features.

Neste trabalho, assume-se que os serviços são versionados de acordo com o modelo de versionamento orientado a features, e, para fins de simplificação, que os perfis de uso

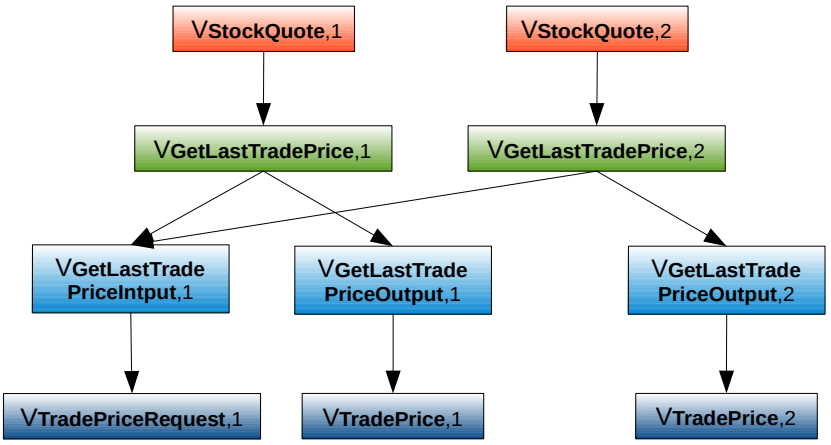


Figura 2.4: Versionamento de features de acordo com modificações na interface.

a serem descobertos devem conter features de uma única versão de serviço. Os termos feature e versão de feature serão utilizados como sinônimos, portanto, pois o uso de mais de uma versão da mesma feature não é avaliado, apesar de ser previsto pela abordagem proposta. A proposta também é válida para a descoberta de perfis de uso contendo diferentes versões de uma mesma feature, o que possibilita analisar o uso de clientes que consomem diferentes versões de um mesmo serviço.

2.4 KDD

KDD (*Knowledge Discovery in Databases*, ou descoberta de conhecimento em bases de dados) é o processo de se identificar padrões implícitos em grandes conjuntos de dados. Os padrões devem ser (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996):

1. **válidos**, no sentido de serem identificáveis, com certo grau de certeza, em novos dados pertencentes ao conjunto a partir do qual foram inferidos;
2. **novos**, o que significa não terem sido identificados anteriormente;
3. **potencialmente úteis** para o usuário ou para a tarefa onde são necessários;
4. **compreensíveis**, se não imediatamente, depois de um pós-processamento.

O processo de KDD é iterativo, pois envolve uma série de etapas que devem ser executadas e podem ser repetidas até que os resultados sejam satisfatórios, e interativo, pois quem o executa deve tomar decisões durante as suas várias etapas. Grande parte das decisões tomadas durante o processo são guiadas por um objetivo de negócio, o qual determina quais questões precisam ser respondidas ou elucidadas através do conhecimento a ser obtido. Por outro lado, as questões de negócio também devem se ajustar aos dados disponíveis, pois eles precisam conter a informação necessária para que se derive o conhecimento desejado.

A Figura 2.5 apresenta uma visão geral das cinco grandes etapas do processo de KDD:

1. **Seleção de dados**: é criado um conjunto de dados alvo, selecionados de acordo com as questões de negócio, extraídos e integrados das diversas fontes de dados em que podem estar presentes;

2. **Pré-processamento**: são executadas operações básicas para tratar inconsistências nos dados, como remoção de ruído e tratamento de dados faltantes;

3. **Transformação de dados**: consiste em transformar os dados e projetá-los de acordo com a tarefa de mineração. Pode ser necessária a redução da dimensionalidade do conjunto de dados (número de variáveis) e/ou a redução do volume de dados (número de instâncias);

4. **Mineração de dados**: envolve a escolha da tarefa de mineração, de acordo das questões de negócio que devem ser respondidas pelo modelo inferido, e a escolha e parametrização de uma técnica de mineração de dados, de acordo com as características do conjunto de dados alvo. A técnica escolhida é então aplicada sobre o conjunto de dados, em busca dos padrões de interesse;

5. **Interpretação e avaliação de padrões**: os padrões inferidos são interpretados e avaliados, o que pode envolver a remoção de padrões irrelevantes e criação de apresentações e visualizações dos resultados. Nesta etapa também avalia-se a necessidade de retornar a alguma das etapas anteriores, em busca de melhores resultados. O conhecimento descoberto pode, enfim, ser incorporado em sistemas ou documentado e informado às partes interessadas.

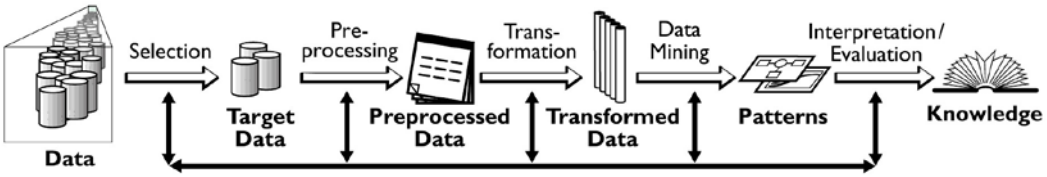


Figura 2.5: Processo de KDD (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Mineração de dados corresponde à extração de padrões implícitos em grandes volumes de dados (HAN; KAMBER, 2006). As técnicas de mineração de dados podem ser divididas de acordo com a tarefa de mineração que se propõem realizar, sendo que cada tarefa resulta em um tipo de padrão diferente e requer entradas em um formato específico. As tarefas de mineração de dados mais comuns são associação, agrupamento (*clustering*), classificação e predição (HAN; KAMBER, 2006).

A escolha da tarefa de mineração está sujeita às questões de negócio a serem respondidas, dado que os padrões resultantes da mineração de dados devem ser capazes de elucidá-la. A escolha da técnica de mineração de dados também pode ser determinada pelo conjunto de dados disponível, pois a eficiência e qualidade dos resultados obtidos através do algoritmo variam de acordo com as características do conjunto de dados.

A mineração de dados, por si só, não garante que os padrões obtidos atendam aos requisitos de validade, utilidade e compreensibilidade, de forma a representar conhecimento. As demais etapas do KDD visam garantir que o resultado final do processo será conhecimento útil, de fato.

Neste trabalho, o processo de KDD é implementado pelo framework de descoberta de perfis de uso de serviços. As etapas do KDD são refletidas pela sequência de execução dos componentes do framework, assim como as tarefas realizadas em cada etapa correspondem aos processos internos de cada componente.

2.5 Clustering

A tarefa de *clustering* (ou análise de agrupamentos) envolve a aplicação de uma forma de aprendizado não supervisionado, que se destina a identificar categorias de objetos em conjuntos de dados. Um agrupamento, ou *cluster*, é uma coleção de objetos de um conjunto de dados cujos elementos são similares entre si e distintos dos elementos contidos em outras coleções de objetos do mesmo conjunto de dados (HAN; KAMBER, 2006), de acordo com algum critério de similaridade. Os clusters representam categorias implícitas de objetos, que permitem descrever um conjunto de dados e entender padrões nele contidos.

O modelo de agrupamentos (ou *clustering*) gerado através de um algoritmo pode ser definido em termos de (TAN; STEINBACH; KUMAR, 2006):

- Organização: o modelo de clusters pode ser particional ou hierárquico;
- Exclusão: dada a distribuição das instâncias nos clusters, o clustering pode ser exclusivo, não exclusivo ou difuso.

Os clusters variam na forma como são definidos, e a utilidade de cada definição depende da natureza dos dados a serem agrupados. Para a formação de cada tipo de cluster é utilizado um tipo distinto de função de similaridade.

Os principais tipos de cluster são (TAN; STEINBACH; KUMAR, 2006):

- Bem separado: objeto pertencente ao cluster é mais similar a qualquer outro objeto do cluster do que a qualquer objeto fora do cluster.
- Baseado em protótipo: cada objeto do cluster é mais similar ao protótipo do seu cluster do que ao protótipo de qualquer outro cluster.
- Baseado em grafo: se os dados são representados por um grafo, então um cluster pode ser definido como um grupo de objetos que possuem conexões entre si.
- Baseado em densidade: cluster definido por uma região com grande densidade de objetos, dentro do espaço definido pelo conjunto de dados, em contraste à esparsidade à sua volta.
- Baseado em distribuição: um cluster é definido por uma distribuição estatística, e os objetos que o compõem são aqueles que possuem mais probabilidade de estar na sua distribuição do que na distribuição dos demais clusters.
- Propriedade compartilhada: o conjunto de instâncias possui clusters pré-definidos de forma arbitrária, denominados clusters naturais. Para identificá-los o algoritmo utilizado deve possuir uma heurística sensível aos padrões do domínio.

Neste trabalho, utiliza-se clustering para agrupar os clientes de um web service de acordo com a forma como utilizam as features de um serviço.

2.5.1 Algoritmos de clustering

As técnicas de clustering são caracterizadas pelo modelo de agrupamento (*clustering*) e o tipo dos *clusters* gerados. A escolha da técnica mais adequada depende da definição de cluster utilizada pelo algoritmo e pelo tipo de cluster que o analista espera encontrar

no seu conjunto de dados. Nesta seção, são descritos 4 algoritmos de clustering utilizados neste trabalho. Maiores detalhes sobre cada algoritmo podem ser encontrados em (MACQUEEN, 1967; ESTER et al., 1996; DEMPSTER; LAIRD; RUBIN, 1977; TAN; STEINBACH; KUMAR, 2006).

O K-Means (MACQUEEN, 1967) é um algoritmo que gera um clustering particional, composto de clusters baseados em protótipos. O usuário informa o número de clusters a serem gerados, e o algoritmo encontra os clusters com a menor distância entre as instâncias e os centroides de seus grupos, de forma iterativa. Os clusters baseados em centroides são sempre circulares, portanto o algoritmo não é adequado para encontrar clusters de formas arbitrárias. Também não é adequado para identificar clusters que possuem intersecções, ou tamanhos e densidades diferentes, pois os centroides que minimizam a soma das distâncias podem não corresponder aos centros dos agrupamentos esperados. Na Figura 2.6 observa-se como o K-Means pode não encontrar os resultados esperados, devido ao diferente tamanho e densidade dos clusters.

(a) Clusters esperados

(b) Clusters resultantes do k-means

Figura 2.6: Resultado do K-Means para clusters de tamanhos diferentes (TAN; STEINBACH; KUMAR, 2006).

O DBSCAN (ESTER et al., 1996) produz um clustering particional baseado em densidade, agrupando instâncias que podem ser conectadas. Essas conexões se baseiam em dois parâmetros: um raio em que as instâncias vizinhas devem ser encontradas, e o número mínimo de pontos que devem estar nesse raio. Qualquer instância que não satisfaça estes critérios é considerada ruído, e excluída do particionamento. O algoritmo infere o número de clusters do conjunto de dados, e é capaz de encontrar clusters com formatos arbitrários (ou clusters naturais). Porém, é difícil encontrar o melhor par de parâmetros para otimizar o algoritmo, e se houver uma grande variação de densidade entre os diferentes clusters esperados, ele é incapaz de obter bons resultados, pois qualquer par de parâmetros é inadequado. O algoritmo também não é adequado para a identificação clusters sobrepostos, ou seja, que não são bem separados. A Figura 2.7 ilustra um conjunto de clusters arbitrários identificados pelo DBSCAN.

O algoritmo de maximização da expectativa, ou *expectation maximization* (EM) (DEMPSTER; LAIRD; RUBIN, 1977), gera um clustering particional e clusters baseados em distribuição. A partir de um número k de clusters desejado, o algoritmo estima os parâmetros de k distribuições Gaussianas, e agrupa as instâncias de acordo com a probabilidade de pertencer a cada distribuição, utilizando o teorema de Bayes. O algoritmo EM é capaz de identificar clusters interseccionados e é pouco suscetível ao ruído. Contudo, a hipótese da distribuição Gaussiana somente permite a identificação de clusters elípticos.

Os algoritmos hierárquicos (TAN; STEINBACH; KUMAR, 2006) seguem a estratégia aglomerativa ou divisiva. O algoritmo aglomerativo inicia com clusters compostos de apenas uma instância, e a cada iteração aglomera os dois clusters mais semelhantes, de acordo com uma métrica de similaridare. O processo pode ser interrompido de acordo

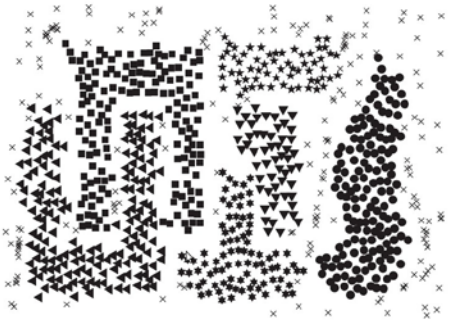


Figura 2.7: Clusters resultantes do DBSCAN (TAN; STEINBACH; KUMAR, 2006)

com o número de clusters desejado, gerando um clustering particional. É adequado para a identificação de subclusters, o que é um problema para outros tipos de algoritmos, e também pode lidar com clusters de diferentes tamanhos. O algoritmo é bastante custoso computacionalmente, e o clustering particional gerado pode não representar a divisão ótima do ponto de vista global, o que pode representar um problema na presença de ruído e alta dimensionalidade (TAN; STEINBACH; KUMAR, 2006).

Como no clustering hierárquico os clusters podem ser vistos como arestas de um grafo, que são ligadas a cada iteração, a forma como os clusters são unidos definem o tipo dos clusters gerados. Esse critério de união é denominado *critério de ligação* (*linkage criteria*). A técnica de *single linkage*, por exemplo, utiliza as duas instâncias mais próximas entre dois clusters para calcular a similaridade. Já a *mean linkage* utiliza a média das similaridades entre todas as instâncias de cada cluster, calculadas par a par.

2.5.2 Avaliação de clusters

Uma dificuldade encontrada na tarefa de clustering está na validação dos resultados obtidos. Visto que normalmente não existem classes previamente definidas sobre os dados analisados, a validação do conjunto de clusters geralmente depende da análise de um especialista, o qual julga a qualidade dos resultados de acordo com sua intuição sobre os clusters esperados, de acordo com suas premissas em relação ao domínio.

A validação é caracterizada como o processo de avaliar os resultados de um algoritmo de clustering, tarefa que pode ser auxiliada por métodos e índices que quantificam a qualidade dos clusters gerados, com base em critérios arbitrários. Estes índices de qualidade visam apoiar a avaliação do especialista quanto à qualidade da configuração do clustering resultante (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2002).

Existem três abordagens para avaliar clusters, segundo Halkidi, Batistakis e Vazirgiannis (2002):

1. Abordagem baseada em critério externo: avaliação da qualidade do conjunto de clusters gerado em relação a uma estrutura de partições esperada;

2. Abordagem baseada em critério interno: avaliação dos resultados utilizando somente as informações utilizadas ou geradas pelo próprio algoritmo de clustering, como os dados de entrada e os clusters identificados;

3. Abordagem baseada em critério relativo: avaliação da estabilidade de um algoritmo, através da comparação de resultados de diversas execuções, utilizando diferentes parâmetros de entrada.

Os índices externos de validação fornecem uma melhor avaliação dos resultados do clustering, porém exigem que um especialista pré-defina um conjunto de partições que represente as categorias ou classes de interesse a serem descobertas no conjunto de dados. São exemplos deste tipo de índices: o Índice de Rand, o Coeficiente de Jaccard, o Índice de Folkes e Mallows, a Correlação Γ de Hubert, como citados por Dalton, Ballarin e Brun (2009), e a F-Measure por contagem de pares (PFITZNER; LEIBBRANDT; POWERS, 2009).

A F-Measure por contagem de pares corresponde ao tradicional cálculo de F-Measure, porém aplicado a pares de instâncias. A métrica se baseia na contagem dos pares corretos e incorretos de instâncias dentro de cada cluster gerado, em relação aos pares de instâncias dentro dos grupos definidos pelo especialista. Nesta verificação de pares, são utilizadas as noções de resultados positivos, falso-positivos e falso-negativos. Um resultado positivo ocorre quando um par existe tanto no conjunto de clusters como nos grupos rotulados; falso-positivo, quando um par existe nos clusters mas não nos grupos rotulados; e falso-negativos, quando o par não existe nos clusters, mas existe nos grupos rotulados.

A fórmula da F-Measure é apresentada na equação 2.1, na qual TP representa o número de pares positivos, FP o número de pares falso-positivos, e FN o número de falso-negativos.

$$\frac{2 * TP}{2 * TP + FP + FN} \tag{2.1}$$

A F-Measure reflete a homogeneidade das instâncias dentro dos clusters e a heterogeneidade entre os clusters, e pode ser calculada mesmo que o número de clusters gerados não corresponda ao número de grupos identificados pelo especialista.

Os índices internos de validação não necessitam de informação do especialista, porém partem de uma hipótese sobre o tipo de cluster esperado. O Índice de Dunn (DUNN, 1974) e o índice de silhueta (ROUSSEEUW, 1987), por exemplo, baseiam-se na premissa de que os clusters gerados são bem separados, e portanto os avaliam como tal.

O índice de silhueta para um clustering é dado pela média das silhuetas de cada uma das instâncias no clustering. A silhueta de uma instância é calculada através da equação 2.2, onde a é a similaridade média da instância $\boldsymbol{i}$ em relação a seu próprio cluster, e b é a menor similaridade média entre $\boldsymbol{i}$ e qualquer outro cluster.

$$s(\boldsymbol{i}) = \frac{b(\boldsymbol{i}) - a(\boldsymbol{i})}{max a(\boldsymbol{i}), b(\boldsymbol{i})} \tag{2.2}$$

O índice SD (HALKIDI; VAZIRGIANNIS; BATISTAKIS, 2000) pondera a dispersão média das instâncias e a separação entre os clusters. A fórmula do índice SD é apresentada na equação 2.3, a qual soma dois componentes: $Scatt(C)$, que mede a dispersão média do clustering C; e $Dis(C)$, que mede a separação entre os clusters de C.

A fórmula de $Scatt(C)$ é apresentada na equação 2.4, na qual n é o número de clusters de C, e σ é a variância de um conjunto de amostras. A variância é um vetor calculado sobre um conjunto de amostras D, sendo que cada uma de suas dimensões é determinado pela equação 2.5, na qual a é um atributo de D, k é o seu número de instâncias, x é uma instância e v é seu centróide. $Dis(C)$ é definido pela equação 2.6, na qual d é o conjunto de distâncias entre os centróides v dos clusters de C, sendo n o número de clusters de C.

$$SD(C) = Scatt(C) + Dis(C) \tag{2.3}$$

$$Scatt(C) = \frac{\sum_{i=1}^{n} \sigma(c_i) \;/\; \sigma(C)}{n} \tag{2.4}$$

$$\sigma_a = \frac{\sum_{i=1}^{k}((x_i^a) - (v_i^a))^2}{n_a} \tag{2.5}$$

$$Dis(C) = \frac{d_{max}}{d_{min}} \sum_{k=1}^{n} \left(\sum_{z=1}^{n} v_k - v_z| \right)^{-1} \tag{2.6}$$

De acordo com Liu et al. (2010), tanto o Índice de silhueta como o SD são capazes de lidar com ruído, clusters de densidades diferentes e clusters de diferentes tamanhos, com a única desvantagem de não se comportarem bem na presença de subclusters, definidos como clusters muito próximos.

O índice S_Dbw (HALKIDI; VAZIRGIANNIS, 2001) é uma evolução do índice SD, combinando as medidas de compactação intra-cluster com um fator de densidade inter-cluster, com o objetivo de tornar o índice independente de tipo de algoritmo. Segundo Liu et al. (2010), o S_Dbw também possui a vantagem de lidar bem com subclusters.

Os índices relativos de validação visam avaliar a estabilidade de um algoritmo de clustering, o que corresponde a quantificar o efeito de alterações de parâmetros de entrada no resultado do clustering.

Nos experimentos realizados neste trabalho, o índice F-Measure é utilizado na avaliação de agrupamentos de clientes de um serviço, a partir de um conjunto de dados rotulados com o perfil de uso de cada cliente. Os índices de silhueta e SD foram incorporados ao framework proposto para auxiliar o usuário na avaliação dos agrupamentos obtidos durante a descoberta de perfis de uso.

2.6 Monitoramento de web services

Para se obter dados de uso de web service, é necessário interceptar as interações entre agentes, delas extrair dados úteis e armazená-los, o que torna necessário o uso de um monitor. O registro da atividade monitorada é comumente denominado *log*, o qual pode ser feito em arquivos ou bases de dados, e distribuídos em diferentes pontos de um sistema, de acordo com a localização do componente a ser monitorado.

Devido à sua natureza distribuída, é um desafio determinar em que ponto da arquitetura de um web service deve ser feita a captura e o armazenamento dos dados de uso (CHUVAKIN; PETERSON, 2009). Pode-se observar as interações tanto no consumidor como no provedor do serviço, assim como em cada componente cujos recursos são abstraídos através do web service (como uma base de dados).

Chuvakin e Peterson (2009) identificam quatro maneiras de se monitorar web services, cuja utilidade varia de acordo com a finalidade do monitoramento:

1. Servidor web: o servidor web é o responsável por receber as requisições HTTP e encaminhá-las ao web service. Ao criar logs das requisições HTTP, o servidor web pode registrar as invocações ao web service;

2. Mediador SOAP: é um sistema, possivelmente um web service, utilizado como um proxy para receber e encaminhar mensagens SOAP enviadas e recebidas pelo provedor. Pode estar localizado no consumidor ou no provedor;

3. Interceptador no framework de web services: interceptador localizado no framework de web services junto à implementação do serviço, muitas vezes utilizando o padrão *adapter* ou *interceptor*, capaz de interceptar as mensagens SOAP direcionadas e realizadas pelo provedor;

4. Registro em código: consiste na adição de chamadas a rotinas para registro dos dados de uso diretamente no código do web service.

Os servidores web possuem uma grande desvantagem em relação às demais alternativas de monitoramento, por registrarem somente as requisições aos métodos HTTP, e não o conteúdo das mensagens trocadas entre os agentes. Como destacado por Chuvakin e Peterson (2009), a parte central do monitoramento de web services está mesmo no servidor de aplicação, onde a implementação do provedor se localiza. Apesar disso, podem ser utilizados quando o registro das invocações ao serviço é suficiente (NAYAK, 2008).

O mediador SOAP, quando localizado no consumidor do serviço, caracteriza-se como um sistema que registra as requisições do cliente e as encaminha ao provedor. No provedor, os registros enviados pelas diferentes aplicações cliente devem ser coletados e integrados. Esta alternativa de monitoramento se torna pouco eficiente, na medida em que exige uma colaboração do consumidor e um esforço de integração por parte do provedor, condições que se tornam insustentáveis quando o serviço possui um grande número de clientes.

A implementação do mediador no provedor possui as vantagens de permitir a centralização das requisições a várias versões de um serviço, o que reduz a necessidade de integração de registros de log, e de manter a implementação do web service independente da estrutura do monitor. Como mediadores SOAP muitas vezes são encarregados de realizar tarefas de autenticação ou validação das mensagens ao serviço, essa infraestrutura existente pode ser aproveitada para o monitoramento das requisições. Isto também facilita o registro dos eventos de autenticação (CRUZ et al., 2004) e permite relacionar os clientes às suas requisições.

O interceptador de mensagens SOAP no framework de web services possui a vantagem de ser facilmente implementado, pois grande parte dos frameworks possuem interfaces para esta função. O interceptador pode ser configurado para mediar tanto mensagens de entrada quanto de saída de um web service. Assim como os mediadores, também permite a independência entre a implementação do serviço e o monitor.

As chamadas de log no próprio código do web service permitem registrar qualquer dado das requisições e das respostas. Esta alternativa, porém, possui um custo de implementação e manutenção muito grande, já que depende da alteração do código do agente.

Existem ainda alternativas menos usuais para o registro das interações. Por exemplo, o provedor pode estar sendo executado em uma plataforma de execução de aplicações, o que permite o registro de alguns tipos de eventos relativos ao acesso ao serviço (TANG; ZOU, 2010). A limitação desta abordagem é que os eventos registrados pela plataforma não permitem a identificação de detalhes contidos nas mensagens, como o uso de tipos.

As diferentes alternativas de monitoramento podem impor restrições em relação aos acesso aos dados contidos nas requisições. A Tabela 2.1 sumariza estas restrições de acesso para cada alternativa de monitoramento, considerando serviços SOAP. O comparativo leva em conta o acesso aos dados necessários para identificar o cliente, o momento em que a interação foi realizada, o serviço e operação requisitada, os parâmetros utilizados e seus tipos, e os dados transmitidos através dos argumentos da mensagem.

Dado	Log de web server	Mediador SOAP	Interceptador no framework	Log em código
Identificação do cliente		X	Depende	Depende
Tempo	X	X	X	X
Web service requisitado	X	X	X	X
Operação requisitada		X	X	X
Parâmetros utilizados		X	X	X
Dados de argumentos da requisição		X	X	X
Registro da resposta	X	X	X	X
Dados da resposta		X	X	X

Tabela 2.1: Métodos de monitoramento e suas restrições de acesso a dados de requisições.

O log do web server apresenta-se como a alternativa mais restritiva, pois não mantém registro dos documentos submetidos ao servidor, dado que as mensagens são geralmente documentos trocados através de requisições HTTP POST. Nesse caso, só é possível identificar o serviço acessado e o envio da resposta, pois os demais dados são enviados na mensagem SOAP.

Através de mediadores SOAP, interceptadores no framework de web services e log em código, é possível acessar todas as informações das mensagens, e portanto, identificar as operações, parâmetros e tipos utilizados em requisições e respostas, além de acessar os dados transmitidos nas mesmas.

A identificação do cliente pode não ser acessível a um interceptador no framework de serviços ou na implementação do serviço, nos casos onde se utiliza um mediador SOAP como serviço de autenticação. Os dados de autenticação podem ser enviados no corpo ou em um cabeçalho da mensagem, os quais podem ser removidos pelo mediador ao encaminhar a mensagem ao serviço.

Neste trabalho, o monitoramento de web services é utilizado para capturar o uso das features dos clientes de um serviço. A análise apresentada nesta seção permite que se escolha a forma de monitoramento mais adequada para a captura dos dados de uso brutos, os quais são processados e analisados ao longo do processo de descoberta de conhecimento.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos relevantes da área de mineração do uso de serviços, de forma a contextualizar a contribuição do trabalho proposto nesta área de pesquisa. Os trabalhos são segmentados de acordo com o nível de análise do uso praticado, e ao final do capítulo é apresentado um comparativo deles em relação ao trabalho proposto.

3.1 Padrões de Uso

Segundo Liang et al. (2006), o propósito da mineração de serviços é descobrir quais são as "formas específicas como os web services (ou suas operações) são utilizadas repetidamente por um grupo de usuários com propriedades similares", o que equivale ao conceito de padrões de uso de serviços. Esses padrões de uso podem ser representados por diversos modelos e em diferentes granularidades de informação, características que determinam suas possíveis aplicações e o nível de detalhamento das análises possíveis. De acordo com Liang et al. (2006), os padrões de uso podem ser classificados em três níveis:

1. Nível de requisição de usuário: análise das informações de usuários, da forma como interagem com serviços (como entidades indivisíveis) e das correlações entre essas informações;

2. Nível de template: análise da forma como os usuários compõem os web services e suas funcionalidades, especialmente na forma de composições de serviços e processos de negócio;

3. Nível de instância: onde são analisados os provedores do serviço, suas restrições não funcionais de uso e como os serviços podem ser compostos, dadas suas restrições.

As questões de negócio e a aplicação do conhecimento determinam o tipo de padrão e o nível de informação a serem obtidos. Para isso, as técnicas de mineração do uso podem se valer de dados provenientes de múltiplos níveis de análise. Os resultados de técnicas distintas também podem ser combinados para identificar correlações entre dados de uso de diferentes níveis.

Este trabalho aborda a descoberta de perfis de uso de serviços, o que envolve dados e padrões dos níveis de requisição de usuário e de template. Neste capítulo são apresentados trabalhos que envolvem a mineração de uso de serviço em ambos os níveis citados, os quais são, posteriormente, comparados ao trabalho desenvolvido.

3.2 Nível de Requisição de Usuário

No nível de requisição de usuário, as principais contribuições encontradas na literatura concentram-se na aplicação de recomendação de serviços. A recomendação de serviços tem por objetivo selecionar ou ranquear um conjunto de serviços de acordo com as preferências funcionais e/ou não funcionais de um usuário. Para isso, os sistemas de recomendação procuram identificar similaridades entre as preferências e características de diferentes usuários, e correlacionar essas informações com seus registros de invocação, satisfação e qualidade de serviço, para selecionar os serviços mais adequados às expectativas do usuário, de acordo com experiências prévias.

Rong, Liu e Liang (2009) explora a ideia de filtros colaborativos para recomendar serviços, valendo-se das preferências de invocação dos usuários de um sistema de recomendação. Na abordagem proposta, primeiramente são identificados usuários com interesses semelhantes, de acordo com a frequência de invocação dos serviços em seus histórico, utilizando a correlação de Pearson para encontrar os vizinhos mais similares. Posteriormente, as invocações de cada usuário são organizadas em conjuntos, de acordo com janelas de tempo pré-estabelecidas e então analisadas para a obtenção das regras de associação entre os serviços invocados. Essas regras são utilizadas para organizar um ranking de serviços no momento da recomendação. Nos experimentos são utilizados dados sintéticos de composição de serviços, a partir do conjunto de dados Movielens[1], utilizado em outros tipos de sistemas de recomendação.

A predição de valores de QoS é uma técnica bastante utilizada na recomendação de serviços. Lo et al. (2012) realizam a predição de valores de QoS considerando a localização do usuário. A partir de um histórico de invocações de serviço e dos valores de QoS relacionados a essas invocações, é utilizado um algoritmo de vizinhos mais próximos, adaptado ao contexto geográfico, para selecionar os clientes mais semelhantes ao usuário, que são, então, utilizados na predição. Zhang et al. (2012) agrupam usuários semelhantes em clusters difusos, utilizando o Coeficiente de Correlação de Pearson como métrica de similaridade entre valores de QoS. A predição é feita através de uma média dos valores dos usuários do cluster ao qual o usuário mais se assemelha. Em seus experimentos, ambos os trabalhos citados utilizam um conjunto de dados real de QoS de serviços, gerados a partir de invocações de um conjunto de clientes simulados.

Kang et al. (2012) recomendam serviços com base em interesses funcionais e preferências não funcionais implícitas dos usuários, sem a necessidade de submissões de consultas explícitas. Os interesses funcionais são identificados através da extração de funcionalidades dos WSDL invocados pelo usuário, na forma de termos frequentes, utilizando o TF-IDF. O mesmo é feito para cada serviço registrado para recomendação. As estatísticas são utilizadas para calcular a similaridade entre os serviços registrados e os interesses do usuário. Assume-se que cada serviço possui um vetor de valores pré-definido para parâmetros de QoS, o que possibilita verificar qual a importância dada pelo usuário a cada um dos parâmetros, e, posteriormente, a similaridade desses critérios com os valores de cada serviço. Cria-se então um ranking de serviços combinando a similaridade dos serviços invocados e a adequação às preferências de QoS. Os experimentos são realizados sobre um conjunto de dados de invocação sintético.

Zhang, Ding e Chi (2011) apresentam uma abordagem para recomendação de serviços que diferencia-se das anteriores, por utilizar um filtro colaborativo para encontrar usuários que apresentam requisitos de QoS similares, ao invés daqueles que apresentam

[1] http://www.grouplens.org/

valores de QoS similares em suas requisições. Após selecionar usuários semelhantes com base nos seus requisitos de QoS (somente para requisições que resultaram na invocação de um mesmo serviço), os serviços são ranqueados através de uma fórmula que pondera o grau de similaridade entre os usuários, a recência das invocações do serviço e a similaridade entre a consulta funcional do usuário e as consultas que originaram as invocações passadas. É proposto que o registro de logs de invocações seja feito no cliente e enviado periodicamente ao provedor. Os dados de requisições utilizados nos experimentos são sintéticos.

Yu (2012) propõe uma solução para o problema de *cold start* relacionado à recomendação de serviços. Os usuários são agrupados de acordo com sua similaridade em relação aos valores de um parâmetro de QoS, armazenados em uma matriz esparsa. Os valores faltantes dessa métrica de QoS são estimados, utilizando fatoração de matrizes. Uma árvore de decisão é inferida do conjunto de usuários e seus vetores de valores para cada serviço, utilizando os clusters como labels. A árvore de decisão pode então ser utilizada para construir um questionário que irá classificar um novo usuário em um dos clusters, permitindo a aplicação de um filtro colaborativo. Os experimentos são conduzidos sobre um conjunto de dados real de QoS de serviços, gerados a partir de invocações de um conjunto de clientes simulados.

3.3 Nível de Template

No nível de template, a recomendação de composições de serviços é uma das principais aplicações da mineração do uso, na qual são identificados grupos de serviços interoperáveis ou correlatos, de acordo com critérios funcionais, e que podem ser utilizados em conjunto para a realização de uma tarefa especificada pelo usuário do sistema de recomendação.

Liang e Chung (2007) fazem um estudo investigativo sobre a utilização da mineração do uso em nível de template. Propõem um método para a análise de composições de serviços correlatos, que se caracteriza pela descoberta de regras de associação entre serviços, a partir de um conjunto de requisições previamente organizadas em conjuntos de invocações de cada usuário. Os experimentos são realizados sobre um conjunto de dados simulado de invocações compostas.

Zhang et al. (2009) buscam encontrar comunidades de serviços (serviços que desempenham tarefas semelhantes), baseando-se na hipótese de que serviços com padrões de composição semelhantes estão relacionados a um mesmo tópico. Propõe um algoritmo de clustering de web services, identificados em um registro de composições previamente identificadas em logs de requisições de clientes. O algoritmo consiste na construção de um grafo de interações entre serviços, no qual os vértices representam serviços e as arestas possuem pesos, atribuídos de acordo com o número de interações entre os serviços. Os serviços são transformados em instâncias compostas dos pesos de suas interações, que são então agrupadas utilizando o algoritmo K-Means. O peso das arestas é também ponderado de acordo com a recência das invocações entre serviços, de forma a minimizar a importância de composições que não são mais utilizadas, o que reflete o efeito da evolução das composições. Os experimentos utilizam logs gerados por simulação.

Outra aplicação em nível de template é a identificação de processos de negócio, que consistem em padrões que descrevem o fluxo de execução de serviços ou de suas operações. Esses workflows podem ser utilizados na verificação de conformidade da implementação de um serviço, verificação da conformidade de requisições e na melhoria de

modelos de processo (AALST, 2012).

No trabalho de Motahari-Nezhad et al. (2011), um modelo de processo de negócio descoberto é definido como uma "visão de processo", por ser inferido a partir de um conjunto de observações. O modelo das possíveis visões de processos encontradas em um conjunto de observações é denominado "espaço de processos". O trabalho apresenta algoritmos para a descoberta semi-automática de processos de negócio em web services, identificados em logs de requisições de serviços, registradas em nível de operações. O processo de identificar workflows envolve a correlação e ordenação dos eventos dos logs de requisição, que é feita através da análise de valores de atributos passados como parâmetros entre as operações requisitadas. No trabalho sugere-se o uso de atributos identificadores de cliente e de sessão. Em seus experimentos, utilizam dois conjuntos de dados de invocação simulados, e um conjunto de dados reais de invocação, que possui porém, um número pequeno de clientes e operações.

Wang e Capretz (2011) constroem um grafo de dependências entre serviços para avaliar o impacto de mudanças em relação à entropia de um conjunto de serviços compostos, permitindo avaliar mudanças em operações e parâmetros. Inicialmente, um algoritmo de análise de links é utilizado para construir um grafo de dependências entre serviços, onde as arestas representam as mensagens enviadas, o que é extraído de descrições de composição (BPEL). Com base nas dependências, calcula-se a entropia de cada serviço e do sistema como um todo. Calculando-se a entropia antes e depois de possíveis alterações, é possível identificar qual causará o menor impacto em relação à entropia do sistema. Todos os serviços precisam ser conhecidos através de descrições de composição (BPEL), portanto não é possível avaliar o impacto a clientes externos. O impacto também não diz respeito ao uso real dos clientes, e sim à coesão do sistema com relação às dependências entre os serviços.

3.4 Comparação das abordagens

Esta seção sumariza os trabalhos discutidos nesse capítulo, e posiciona o trabalho proposto em relação a esse panorama. Na Tabela 3.1 é apresentado um comparativo das abordagens de mineração do uso, de acordo com sua aplicação prática, objetivo de análise, nível de padrão de uso, tipo do modelo obtido, e granularidade de informação do modelo.

No tocante à aplicação, verifica-se uma predominância de estudos voltados à descoberta de serviços relevantes (recomendação de serviços e de composições de serviços), e à avaliação do processo interno de desenvolvimento e otimização dos web services, como a descoberta de processos de negócio. A proposta deste trabalho apoia aplicações de gestão da evolução de serviços baseadas no uso real.

Em relação ao objetivo de análise, esta proposta visa a construção de perfis de uso de serviços, os quais identificam como grupos de clientes utilizam conjuntos de funcionalidades de um serviço, e quantificam esse uso conforme as invocações dos clientes agrupados. Em relação ao nível dos padrões de uso, portanto, são contemplados os níveis de requisição de usuário e de template.

Quanto ao tipo de modelo, contribui com clusters de aplicações cliente, mas cuja granularidade e aplicação variam em relação às propostas existentes. A granularidade do modelo é menor, pois é definida com base na granularidade das features utilizadas no agrupamento das aplicações (i.e. serviços, operações e tipos de dados), contidas nos dados de uso (ver Seção 2.3). Em quase todos os trabalhos analisados, a granularidade encontra-se no nível de serviço, como um todo.

	Aplicação	Objetivo	Nível de padrão de uso	Tipo de modelo	Granularidade do modelo
Rong, Liu e Liang (2009)	Recomendação de serviços	Ranquear serviços com base em invocações de usuários semelhantes	Requisição de usuário	Regras de associação	Serviços
Lo et al. (2012)	Recomendação de serviços	Prever valores de QoS de serviços utilizando valores de invocações semelhantes de usuários geograficamente próximos	Requisição de usuário	Matriz de similaridade entre usuários	Serviços
Zhang et al. (2012)	Recomendação de serviços	Prever valores de QoS de serviços utilizando valores de usuários com invocações semelhantes	Requisição de usuário	Clusters difusos de usuários baseado em invocações e valores de QoS	Serviços
Kang et al. (2012)	Recomendação de serviços	Recomendar serviços com base em preferências funcionais e não funcionais implícitas, de acordo com invocações passadas do usuário	Requisição de usuário	Similaridades entre usuário e serviços, vetor de preferências de QoS	Serviços
Zhang, Ding e Chi (2011)	Recomendação de serviços	Recomendar serviços com base em invocações de usuários com requisitos de QoS e invocações semelhantes	Requisição de usuário	Matriz de similaridade entre usuários	Serviços
Yu (2012)	Recomendação de serviços	Agrupar usuários de acordo com valores de QoS de invocações reais ou estimados	Requisição de usuário	Árvore de decisão, matriz de valores de QoS	Serviços
Liang e Chung (2007)	Recomendação de composições de serviços	Identificar composições frequentes entre conjuntos de serviços	Template	Regras de associação	Serviços
Zhang et al. (2009)	Recomendação de composições de serviços	Agrupar serviços compostos que desempenham tarefas semelhantes com base nos serviços que os compõem	Template	Clusters de serviços	Serviços
Motahari-Nezhad et al. (2011)	Otimização de serviços	Identificar workflows de serviços	Template	Modelo de workflow	Operações
Wang e Capretz (2011)	Avaliação do impacto de mudanças	Quantificar a importância de serviços em um sistema fechado e avaliar o efeito de mudanças em relação à entropia do sistema	Template	Matriz de entropia de serviços	Serviço

Tabela 3.1: Comparativo de trabalhos de mineração do uso.

A validação do trabalho proposto é feita através de experimentos sobre dados sintéticos de invocação de um serviço. É importante ressaltar que esta é uma forma comum de validar trabalhos em mineração de dados de uso de serviço, devido à dificuldade de se obter dados reais nesse domínio, dada sua natureza proprietária (NAYAK, 2008; MOTAHARI-NEZHAD et al., 2011).

Dos 10 trabalhos analisados, 4 utilizam dados sintéticos de invocação de serviços (KANG et al., 2012; ZHANG; DING; CHI, 2011; ZHANG et al., 2009; LIANG et al., 2006), 1 simula a composição de serviços a partir de dados de outro domínio (RONG; LIU; LIANG, 2009), e 1 não apresenta experimentos (WANG; CAPRETZ, 2011). Apenas 1 trabalho utiliza um conjunto de dados de invocações reais, o qual possui um número pequeno de clientes e operações, e outros dois conjuntos de dados sintéticos. Os demais 3 trabalhos utilizam dados de QoS de serviços reais, os quais podem ser obtidos através da simulação de clientes (LO et al., 2012; ZHANG et al., 2012; YU, 2012).

O trabalho proposto apresenta uma contribuição relevante, ao propor uma abordagem que diferencia-se das demais, em todos os aspectos analisados.

4 GERENCIADOR DE PERFIS DE USO

Este capítulo apresenta, inicialmente, uma visão geral do Gerenciador de Perfis de Uso, na qual suas funções e arquitetura são descritas resumidamente, assim como suas interações com os demais módulos do framework de gerência de evolução WS-Evolv. Em seguida, a estrutura dos perfis de uso de serviços é descrita. Nas seções seguintes, são apresentados em detalhes os componentes do gerenciador de perfis, responsáveis pelo monitoramento das interações entre provedor e clientes, extração e armazenamento de dados de uso, e pela geração dos perfis de uso.

4.1 Visão geral

O Gerenciador de Perfis de Uso, detalhado na Figura 4.1, é um dos módulos do framework WS-Evolv, que visa apoiar provedores na gestão da evolução de web services. O Gerenciador de Perfis possui 3 objetivos principais:

1. monitorar automaticamente requisições de clientes a serviços, e delas extrair dados de uso brutos em granularidade fina;
2. armazenar os dados de uso pré-processados em um repositório centralizado, o qual permite múltiplos tipos de análise;
3. permitir o desenvolvimento de um processo de KDD para gerar perfis de uso de serviços, com o mínimo possível de intervenção do usuário. Esse propósito é atingido através da pré-definição de tarefas do processo de KDD, permitindo que sejam configuradas através de parâmetros simples.

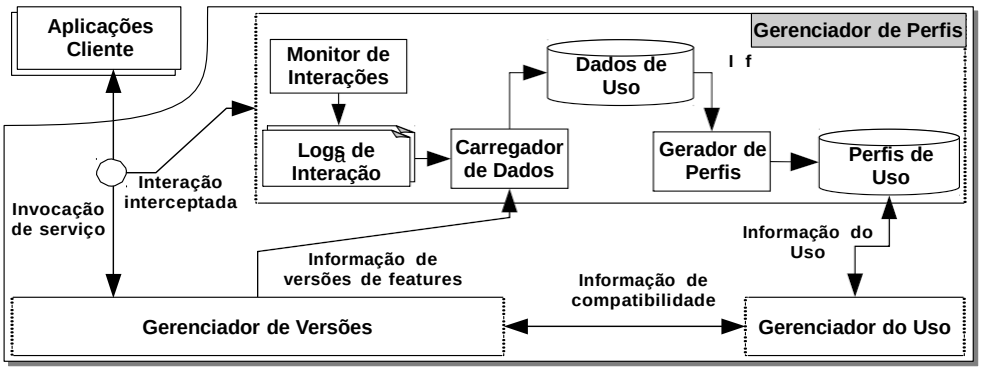


Figura 4.1: Framework de evolução de serviços e Gerenciador de Perfis.

O Gerenciador de Perfis possui como componentes:

1. um Monitor de Interações, responsável por interceptar as mensagens trocadas entre provedor e clientes de um web service;

2. um Carregador de Dados, responsável por extrair dados de uso de features a partir das mensagens capturadas pelo monitor, e por armazená-los em uma base de propósito geral;

3. e um Gerador de Perfis de Uso, responsável por agrupar clientes de acordo com seus padrões de uso, e por transformar estes grupos em perfis de uso de serviços.

Os componentes são detalhados nas Seções 4.3, 4.5 e 4.6, respectivamente.

4.2 Perfis de uso

Os perfis de uso são representações de grupos de aplicações cliente que apresentam padrões de uso similares em relação às funcionalidade descritas na interface de um serviço. Tais padrões identificam os serviços e operações utilizados pelos clientes e os tipos de dados trocados em suas interações. Denominamos *features* estas partes identificáveis da descrição da interface de um serviço.

A estrutura dos perfis de uso é ilustrada na Figura 4.2. Cada perfil está relacionado às aplicações que compartilham os padrões extraídos, e às features utilizadas em suas interações com o serviço. Métricas podem ser associadas às aplicações (por exemplo, o número total de requisições) ou às features (por exemplo, o número de interações requisitando uma operação ou envolvendo um tipo).

Baseado no modelo de versionamento orientado a features adotado no WS-Evolv (Seção 2.3), utilizamos features como as unidades que identificam e descrevem os perfis de uso. No entanto o processo de descoberta de perfis é relativamente independente da granularidade dos dados, e pode ser aplicado para qualquer conjunto de elementos identificáveis na interface dos serviços e armazenados pelo Gerenciador de Versões.

Com já mencionado, no que se refere a perfis de uso, utilizamos o termo *feature* como sinônimo de *versão de feature*, por questões de simplificação. Portanto, no modelo de perfis uma *Feature* corresponde a uma *Versão de Feature* do repositório de versões, cujo esquema corresponde ao modelo de versionamento apresentado na Figura 2.3. O identificador da *Feature* permite navegar entre o repositório de versões e o repositório de perfis, em ambas as direções.

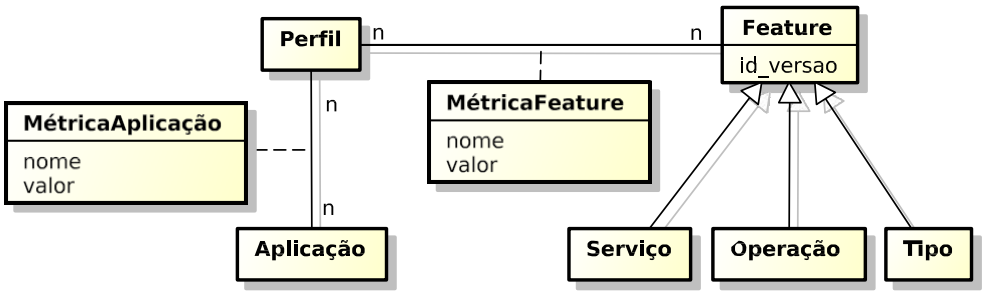


Figura 4.2: Estrutura de perfis de uso de serviços.

A escolha pela representação do uso na granularidade de features se deve à necessidade de identificar quais funcionalidades da interface do serviço são, de fato, utilizadas. O agrupamento de aplicações pelo uso de serviços permite identificar grupos que compõem

serviços de forma similar. O agrupamento por operação permite identificar aplicações que utilizam um conjunto semelhante de operações, dentre as presentes na interface dos diversos serviços. Já o agrupamento por tipo permite identificar grupos de aplicações que fazem um uso diferenciado de parâmetros opcionais das operações. As métricas relacionadas ao uso das features (*MétricaFeature*) permitem, ainda, diferenciar grupos de aplicações que utilizam os mesmos conjuntos de features, mas com frequências distintas.

A análise de perfis nesse nível de detalhamento pode revelar conhecimento útil para diversas aplicações. Por exemplo, sabendo quais operações do serviço estão sendo de fato utilizadas, ou com qual frequência são requisitadas, os provedores/designers de um serviço podem decidir pela realização de mudanças incompatíveis para melhorar a qualidade do serviço, as quais normalmente não seriam consideradas devido à possibilidade de quebra de aplicações, que no cenário de pior caso afetariam todos os clientes.

O conhecimento sobre quais operações são usadas em conjunto por um grupo relevante de aplicações também pode servir como guia para o redesenho da interface de serviços, de forma a otimizar a experiência de grupos específicos de clientes. Os provedores também podem estar interessados, por exemplo, em entender quais parâmetros opcionais de operações são utilizados por quais grupos de clientes, para decidir sobre a realização de mudanças em seus tipos de dados. Devido à variedade de casos nos quais podem ser aplicados, buscamos agregar nos perfis de uso tanta informação quanto possível, deixando a exploração desta informação a cargo do provedor, de acordo com seus requisitos de análise.

4.3 Monitor de interações

O Monitor de Interações é o componente responsável por interceptar e registrar em um log as mensagens trocadas entre as aplicações cliente e as versões do serviço utilizadas. Dado o nosso propósito de detectar padrões no uso de serviços, o Monitor de Interações deve ser capaz de interceptar e registrar as operações requisitadas e as mensagens trocadas, o que possibilita a identificação do serviço, operações e parâmetros utilizados.

Diversas opções de monitoramento foram apresentadas na Seção 2.6, onde vantagens e desvantagens foram discutidas. São elas o log no servidor web, mediador SOAP, interceptador no framework de web services e log em código.

Considerando o propósito do Gerenciador de Perfis, o log no servidor web está descartado, pois não registra os dados contidos em mensagens. O mediador SOAP localizado no cliente e o log no código do agente também não representam boas alternativas, pois a primeira resulta em um overhead no transporte de mensagens e na consolidação de logs, e a segunda aumenta os custos de desenvolvimento e manutenção do serviço.

Desta forma, as melhores opções são a implantação de um mediador SOAP gerenciado pelo provedor, ou de um interceptador no próprio framework do web service. Apesar do mediador SOAP representar um custo maior na comunicação com o cliente, ele pode ser utilizado para outras funções, como controle de autenticação. Já o interceptador possui a vantagem de utilizar a infraestrutura do framework do web service para decodificar a mensagem, o que representa uma economia de recursos em relação ao mediador SOAP. Ambas as alternativas possuem a vantagem de não serem afetadas pela evolução da interface do serviço.

Neste trabalho, assumimos que que cada versão do serviço possuirá um interceptador, o qual deve registrar as mensagens SOAP em um arquivo de log. Os logs são posteriormente processados para extração dos dados de uso de cada requisição, como detalhado

na Seção 4.5. Devido à necessidade de relacionar as requisições às suas respectivas aplicações cliente, também assume-se que as versões do serviço possuem um mecanismo de autenticação, o qual associa à mensagem um identificador único para cada aplicação. A exigência de autenticação é uma prática comum aos provedores de web services, que pode ser feita, por exemplo, através do envio de tokens de acesso.

4.4 Base de Dados de Uso

A Base de Dados de Uso é um repositório de propósito geral que centraliza o acesso a dados detalhados sobre o uso de serviços. Na medida em que possibilita o armazenamento de dados em diferentes níveis de detalhamento, o repositório permite diversos tipos de análise, e a experimentação de critérios distintos para a definição de perfis de uso.

O detalhamento do uso se dá através do registro das versões de features utilizadas pelas aplicações cliente em suas requisições. As versões de features correspondem ao serviço e operação invocados, e aos tipos utilizados nos parâmetros da operação, de acordo com a descrição da interface do serviço. Todas as features utilizadas em uma requisição devem estar versionadas, conforme o modelo de versionamento orientado à features apresentado na Seção 2.3, e armazenadas em um repositório mantido pelo Gerenciador de Versões (Figura 4.1).

O esquema da Base de Dados de Uso é detalhado na Figura 4.3. Cada *Aplicação* realiza uma ou mais *Interações*, que correspondem a uma invocação ao serviço ou resposta à invocação. Na *Interação*, uma ou mais *Features* são utilizadas.

Em cada *Interação*, somente um serviço e uma operação são utilizados. A *Interação* referencia diretamente estas duas *Features*, através do relacionamento *Interação Feature*. Um identificador (*id_versão*) permite associar cada versão de *Feature* à sua respectiva versão no repositório de versões.

Os tipos podem ser utilizados diversas vezes em uma Interação, por meio dos diferentes parâmetros nela contidos. O relacionamento *Interação Parâmetro* representa um parâmetro, o qual relaciona uma *Interação* a uma *Feature*, que nesse caso, é o tipo do parâmetro. Cada parâmetro utilizado na interação possui uma representação no repositório de versões, que é referenciada através do atributo *id_repositorio_versoes*. A informação sobre a obrigatoriedade de cada parâmetro é armazenada no atributo *obrigatorio*, informação esta obtida do repositório de versões.

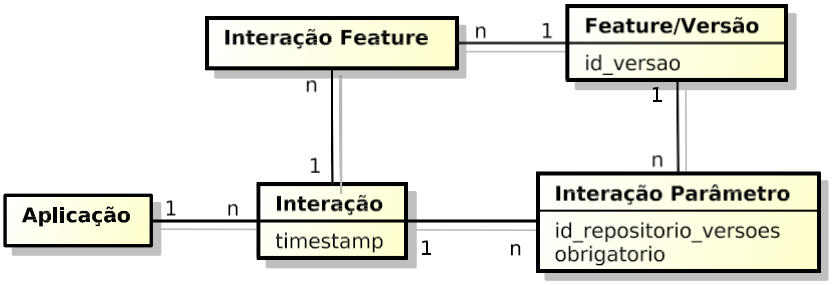


Figura 4.3: Esquema da base de dados de uso.

No gerenciador de perfis, a base de dados de uso é carregada com dados extraídos pelo Carregador de Dados (Seção 4.5), a partir de registros de requisições criados pelo Monitor de Interações (Seção 4.3). Os dados de uso são então utilizados no processo de geração de perfis (Seção 4.6).

A base de dados de uso abstrai a complexidade de se lidar com os dados brutos contidos nas mensagens das requisições ao serviço, facilitando o acesso aos dados de uso das aplicações cliente. Desta forma, a base de dados de uso pode ser utilizada diretamente como como fonte de dados para outras aplicações em que o uso de serviços deve ser analisado, como é o caso dos trabalhos citados no Capítulo 3.

4.5 Carregador de Dados

O Carregador de Dados é o componente responsável por transformar dados brutos das interações entre aplicações cliente e serviços em um conjunto de registros de requisições e features utilizadas, como representado no esquema da base de dados de uso (Figura 4.3).

O carregador recebe dois parâmetros de entrada. O primeiro é o log de mensagens SOAP de uma versão de serviço, registrado pelo Monitor de Interações. O segundo é o identificador da versão de serviço à qual o log se refere, conforme o modelo de versionamento utilizado pelo Gerenciador de Versões (Seção 2.3). Assume-se que cada versão do serviço possui seu próprio monitor de interações, o que permite a identificação prévia da versão.

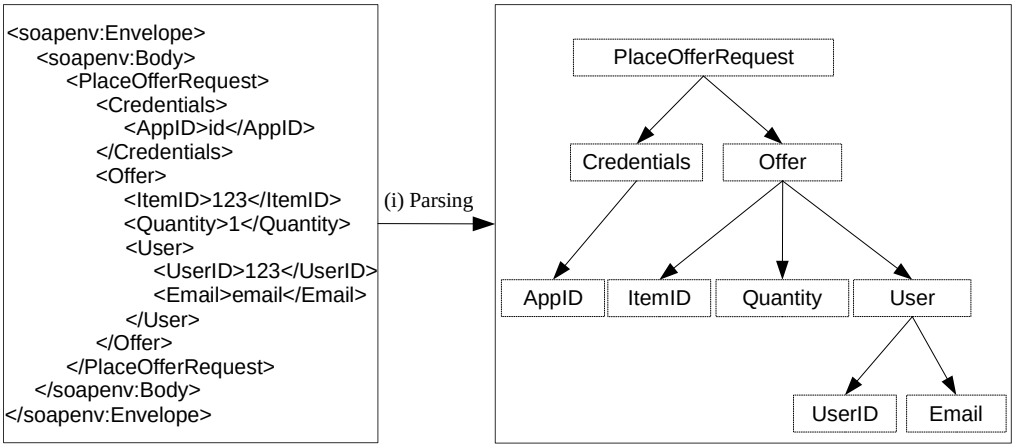
Assume-se também que as mensagens SOAP registradas nos logs seguem o encoding literal, o que significa que somente a hierarquia dos parâmetros e seus valores são informados, como ilustrado na Figura 4.4. Desta forma, o Carregador de Dados necessita identificar a operação e os tipos de parâmetros utilizados, implicitamente representados na mensagem.

Dadas estas premissas, o processo de carga de uma interação segue uma sequência de quatro etapas. Na primeira etapa, faz-se o parsing da mensagem SOAP, onde são extraídas as informações necessárias para a identificação das features. Na segunda etapa, é feita identificação da operação utilizada. Na terceira etapa, identifica-se os parâmetros e tipos utilizados. Por fim, é feito o armazenamento na Base de Dados de Uso.

O parsing da mensagem SOAP consiste na extração da hierarquia de parâmetros utilizados na interação. Utilizando um parser XML, os nomes dos parâmetros são identificados recursivamente e armazenados em uma estrutura de dados hierárquica, conforme ilustrado na Figura 4.4. Também é extraído o identificador da aplicação que realizou a interação, o que permite relacionar, na Base de Dados de Uso, cada interação à sua respectiva aplicação. Esse identificador de cliente é obrigatório, porém pode variar quanto à localização no corpo da mensagem, o que exige uma customização do parser de acordo com o serviço analisado.

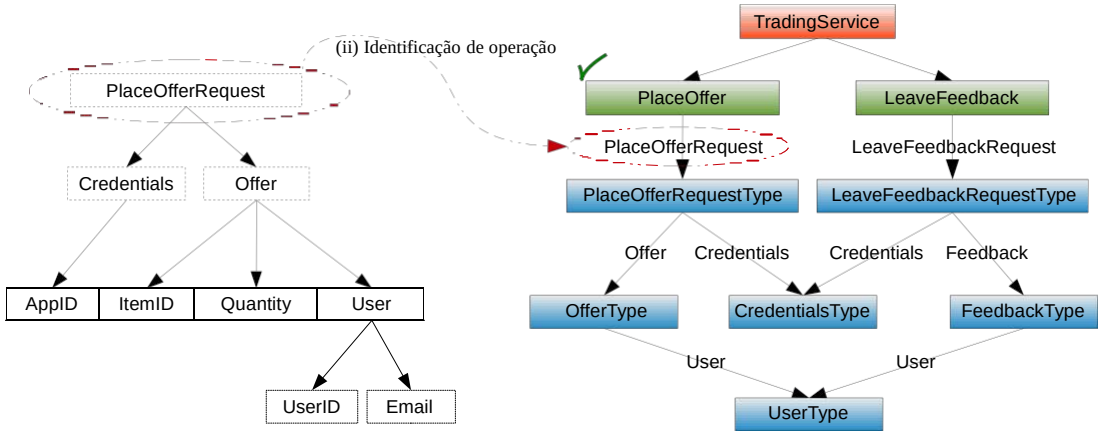
Ao processar uma mensagem, o carregador acessa a descrição do serviço no repositório de versões, identificando a operação requisitada com base em seus parâmetros. Como ilustrado na Figura 4.5, o carregador compara o conjunto de parâmetros da operação, posicionado no primeiro nível da hierarquia de parâmetros da mensagem (constituído unicamente por PlaceOfferRequest), aos parâmetros de cada operação do serviço no repositório de versões. A operação (PlaceOffer) é identificada por inferência, quando verifica-se que seus parâmetros correspondem exatamente aos parâmetros informados na mensagem.

Para identificar os parâmetros e tipos utilizados, o carregador acessa recursivamente a hierarquia de parâmetros da operação registrada no repositório de versões, comparando-a com a hierarquia de parâmetros informada na requisição. O identificador e nome dos parâmetros e tipos são identificados e relacionados à interação na Base de uso. Considerando a Figura 4.5, o carregador iniciaria a recursão pelo parâmetro 'PlaceOfferRequest', percorrendo em seguida os parâmetros de 'PlaceOfferRequestType' e 'OfferType', iden-



(a) Mensagem SOAP (b) Estrutura hierárquica de parâmetros

Figura 4.4: Extração de parâmetros da mensagem SOAP.



(a) Estrutura hierárquica de parâmetros (b) Serviço no Repositório de versões.

Figura 4.5: Identificação de operação por inferência

tificando os tipos de cada um.

Nesse processo, o carregador descarta requisições que não estão de acordo com a descrição do serviço, e são, portanto, inválidas. Por exemplo, a requisição seria descartada caso o parâmetro obrigatório 'Offer' não fosse informado, ou se o parâmetros 'AppId' do tipo 'CredentialsType' estivesse com um tipo incorreto.

A aplicação, sua interação e as features utilizadas são, então, armazenadas na Base de Dados de Uso. É importante ressaltar que o Carregador de Dados armazena somente a estrutura das requisições (as operações e tipos utilizados), e que os dados transmitidos pelas partes envolvidas são dispensáveis, com exceção do identificador do cliente e do timestamp da requisição. O carregador armazena no banco de dados de uso o identificador de cada feature no repositório de versões, o que permite relacionar os dados em ambas as direções.

4.6 Gerador de Perfis de Uso

O Gerador de Perfis de Uso é o componente do Gerenciador de Perfis responsável por:

1. apoiar a condução de um processo de KDD para identificar grupos de aplicações que possuem padrões de uso de serviços em comum;

2. construir perfis de uso a partir dos grupos de aplicações identificados.

A descoberta de perfis é desenvolvida através da parametrização de um conjunto de tarefas pré-definidas, apresentadas na Figura 4.6, de forma a ocultar do usuário do framework parte da complexidade natural de um processo de KDD. Um mínimo de interação é exigido nas etapas do processo, nas quais o usuário:

1. Seleção: provê parâmetros para selecionar dados da base de dados de uso, de forma a atender os requisitos de análise;

2. Transformação: seleciona entre alternativas de transformação de dados pré-definidas;

3. Clustering e avaliação: parametriza algoritmos de agrupamento e compara os resultados utilizando métricas;

4. Construção de perfis: inicia a geração automática de perfis para os agrupamentos validados.

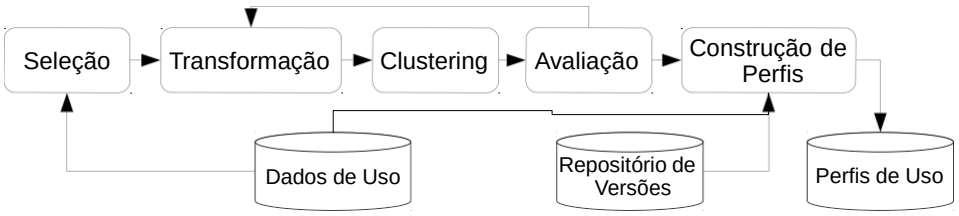


Figura 4.6: As tarefas do processo de descoberta de perfis.

No Capítulo 5, todas as tarefas realizadas durante a descoberta de perfis de uso são descritas em detalhe.

4.7 Considerações finais

Neste capítulo foi apresentada uma visão geral sobre framework proposto, nomeado Gerenciador de Perfis de Uso. O framework possibilita integrar as tarefas de monitoramento de serviços, extração e armazenamento de dados de uso, e descoberta de perfis de uso de serviços.

O Gerenciador de Perfis pode ser utilizado como componente do framework de evolução de serviços, no qual atua como provedor de conhecimento para ferramentas que auxiliam nas decisões de provedores, assim como em outras aplicações onde a análise do uso se faz necessária. O nível de granularidade da representação do uso nos perfis possibilita desde a análise do uso de serviços e operações, até de tipos de dados, o que permite verificar o uso de parâmetros opcionais.

A Base de Dados de Uso proposta permite o armazenamento e acesso a dados de uso de diversos clientes a diferentes serviços, e é flexível quanto à disponibilidade de dados e necessidade de análise dos usuários, pois não impõe qual o nível de detalhamento mínimo para os dados de entrada. O Carregador de Dados dá o suporte necessário para a extração de dados de uso a partir de mensagens SOAP, de forma integrada ao sistema de

versionamento orientado a features, possibilitando unificar logs de diferentes serviços e de versões de um mesmo serviço.

As tarefas de coleta, extração e pré-processamento de dados são essenciais para o Gerador de Perfis de Uso, as quais são detalhadas no próximo capítulo.

5 DESCOBERTA DE PERFIS DE USO

Este capítulo descreve o processo de descoberta de perfis de uso, ilustrado na Figura 4.6. O processo é executado pelo Gerador de Perfis (Figura 4.1), o qual tem por objetivo dar apoio à condução do processo de KDD, reduzindo sua complexidade, do ponto de vista do usuário. Para tanto, o Gerenciador de Perfis busca minimizar a necessidade de interação com o usuário através de tarefas parametrizáveis, as quais são detalhadas no decorrer do capítulo.

5.1 Preparação de dados

A preparação dos dados é crucial na descoberta de perfis, pois influencia a forma como os grupos de clientes são formados pelos algoritmos de agrupamento. As funções de similaridade e algoritmos de agrupamento são muito sensíveis às características dos dados de entrada, portanto são fortemente influenciados pelo filtros e transformações aplicados.

Os dados precisam ser cuidadosamente selecionados, de forma alinhar os padrões de uso descobertos com os objetivos de negócio, e as transformações devem ajustar os dados selecionados aos objetivos de mineração e características dos algoritmos utilizados. As tarefas de seleção e transformação foram pré-definidas considerando possíveis objetivos de análise, no que tange à evolução de serviços.

Neste trabalho, a seleção e preparação de dados são feitas de forma a reduzir a redundância nos dados de uso das features, sabendo-se que muitas delas, por serem utilizadas pela maioria dos clientes, não se prestam à identificação de padrões. A preparação de dados, no escopo do trabalho, visa otimizar os resultados da mineração de dados para atender às necessidades de informação de aplicações voltadas à evolução de serviços, principalmente a identificação de mudanças e avaliação de impacto de mudanças. Desta forma, o trabalho não tem por objetivo exaurir as possibilidades de preparação, e sim verificar a qualidade de soluções em um contexto específico.

5.1.1 Seleção de dados

A seleção de dados tem como objetivo limitar o conjunto de dados utilizado na mineração, para que sejam utilizados somente dados que possam vir a fornecer conhecimento útil ao usuário, o que também inclui a remoção de volumes de dados que não auxiliam ou prejudicam a descoberta de padrões.

Para o cumprimento dos objetivos da seleção, é favorável que os dados estejam armazenados em uma base integrada, que facilite a realização de consultas de forma flexível, como o é a Base de Dados de Uso. A partir desta base, foram incorporadas ao framework tarefas pré-definidas de seleção de dados, parametrizáveis por dois fatores: tempo e gra-

nularidade dos dados.

O uso é temporal, o que significa que os cliente podem, ao longo do tempo, mudar a forma como usam os serviços (por exemplo, em relação às operações requisitadas), o que é algo esperado nos ciclos de vida desacoplados de serviços e aplicações cliente. O provedor do serviço pode estar interessado nos padrões de uso com uma validade temporal, como no último mês, ou desde o lançamento da última versão do serviço. Para atender a essa necessidade, o componente de seleção é parametrizado com timestamps inicial e final de um período de interesse, o qual determina o intervalo de tempo em que se inserem as interações selecionadas.

A granularidade se refere ao nível de detalhe utilizado para agrupar as aplicações. O usuário pode analisar o uso no nível de operações, ou em mais detalhes, de acordo com as operações e os tipos de dados trocados nas mensagens. No primeiro caso, são considerados similares os clientes que usam as mesmas operações, enquanto que no segundo, são similares de acordo com as estruturas das mensagens trocadas.

A granularidade é particularmente importante para a análise de questões referentes à evolução de serviços. Clusters definidos por similaridade de operações permitem avaliar questões complexas, com por exemplo, reconhecer quais conjuntos de operações são utilizadas em conjunto com maior frequência, e por quais usuários, ou verificar se operações pouco usadas individualmente também são pouco usadas em conjunto com outras. Já clusters definidos por similaridade de operações e tipos poderão se distinguir pelo uso ou não de parâmetros opcionais, o que permite decidir sobre sua remoção ou obrigatoriedade.

A escolha do nível de granularidade das features determina os dados a serem extraídos da Base de Dados de Uso. Se o nível de operação for escolhido, a consulta à base de dados deve selecionar todas as features referenciadas através do relacionamento 'Interação Feature', o qual relaciona as interações dos clientes às features correspondentes aos serviços e operações (Figura 4.3). Somente são selecionadas as features utilizadas em pelo menos uma interação no intervalo de tempo definido.

Se o uso dos tipos for requerido, as features referenciadas através do relacionamento 'Interação Parâmetro' (Figura 4.3) também serão retornadas. Esse relacionamento representa os parâmetros utilizados na interação, de forma que as features referenciadas por ele correspondem aos tipos de dados.

A eliminação de features irrelevantes é importante para o posterior agrupamento das aplicações, posto que a alta dimensionalidade e redundância de atributos são fatores preponderantemente prejudiciais para a qualidade dos resultados de algoritmos de agrupamento.

Com esse intuito, é recomendável selecionar somente a parte variável das requisições envolvendo uma determinada operação, ou seja, os tipos relacionados a uma árvore de parâmetros que podem não ser informados em uma requisição. Esta árvore de parâmetros deve ter sua raiz em um parâmetro opcional, de forma que todos os parâmetros nos níveis inferiores, mesmo que obrigatórios, podem ser omitidos da requisição.

No caso inverso, quando o tipo é referenciado por parâmetro obrigatório da requisição, o tipo em questão será utilizado de forma idêntica por todas as aplicações. Esses parâmetros não apenas não servem para distinguir grupos de aplicações, como fazem com que aplicações que deveriam ser consideradas diferentes se assemelhem, produzindo um forte viés. A seleção dos tipos relevantes é feita através de um filtro opcional que retém os tipos utilizados por apenas parte dos clientes.

Para ilustrar quais os tipos de dados relevantes em uma mensagem, a Figura 5.1 exibe um serviço com as operações Op1 e Op2, e suas respectivas estruturas complexas de

mensagens definidas em termos de 4 tipos. As caixas pontilhadas denotam parâmetros opcionais (P2, P6), e as sólidas, os parâmetros obrigatórios. O tipo T1 não é selecionado, pois os parâmetros P1 e P3 são obrigatórios, portanto eles sempre são utilizados em requisições a Op1 e Op2. O tipo T2 é selecionado, pois é referenciado somente pelo parâmetro opcional P2. Desta forma, aplicações cliente que requisitam Op1 e Op2, com mensagens que incluem T2, são considerados diferentes daqueles que não utilizam T2. Note que T4 não é selecionado, pois ele também é referenciado pelo parâmetro obrigatório P5, e portanto, será utilizado em todas as requisições.

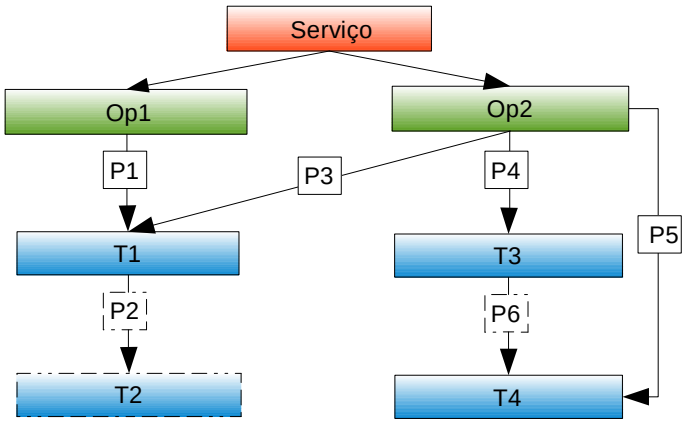


Figura 5.1: Parâmetros obrigatórios e opcionais de um serviço.

5.1.2 Transformação de dados

O objetivo da etapa de transformação é transformar e consolidar os dados selecionados em uma forma adequada para a execução da etapa de mineração. Para a execução da tarefa de agrupamento, é necessário projetar os dados em um conjunto de instâncias, de acordo com um critério de similaridade a ser utilizado na criação de grupos. Na descoberta de perfis de uso de serviços, as instâncias representam aplicações clientes do web service, e a similaridade entre elas é determinada pelo conjunto de features que utilizam.

A transformação envolve dois tipos de representação do uso das features por aplicações: *binário* e *ponderado*. Nas duas formas de transformação, os dados de uso selecionados são projetados em um formato tabular, o qual sumariza o uso das features pelas aplicações cliente. Nesta tabela, as instâncias são representadas pelas linhas, onde cada linha corresponde a uma aplicação cliente, e as features correspondem às colunas. O valor de uma célula é a sumarização do uso de uma feature por uma aplicação. A representação escolhida impacta enormemente o critério de similaridade utilizado no agrupamento dos clientes, o que pode resultar em perfis de uso distintos.

A transformação de dados binária corresponde à representação simples do uso das features pelas aplicações. Ou seja, considera-se somente que a aplicação utiliza ou não uma feature, sem levar em conta o número de interações nas quais a feature é utilizada. A Tabela 5.1 apresenta um exemplo do resultado da transformação binária. Nesta tabela, as células assumem o valor 1 quando a aplicação usa a feature relacionada, ou 0, quando a aplicação não a usa.

Do ponto de vista de similaridade, a escolha desta transformação implica considerar como similares as instâncias que utilizam o mesmo conjunto de features, e como diferentes aquelas que não usam features em comum.

Aplicação	Operação 1	Operação 2	Tipo 1	Tipo 2
App1	1	0	1	0
App2	1	0	1	0
App3	1	1	1	1
App4	0	1	0	1
App5	0	1	0	1
App6	1	1	1	1

Tabela 5.1: Um exemplo de transformação binária.

A Figura 5.2 apresenta uma representação gráfica dos dados binários contidos na Tabela 5.1, considerando apenas features Operação 1 (eixo horizontal) e Operação 2 (eixo vertical). Nota-se que as instâncias de acordo com a preparação binária formam 3 agrupamentos naturais: clientes que utilizam somente a Operação 1 (App1, App2), somente a Operação 2 (App4, App5), ou ambas (App3, App6).

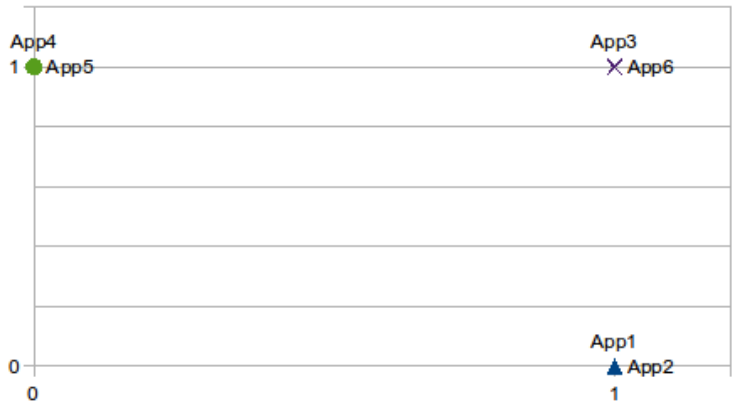


Figura 5.2: Dados de transformação binária

Já no segundo tipo de representação, ponderada, o uso é representado pelo número de interações nas quais a feature é utilizada pela aplicação. A Tabela 5.2 apresenta um exemplo do resultado da transformação ponderada. Nesta tabela, as células assumem o valor correspondente ao número de interações em que as features são usadas pelas aplicações.

Aplicação	Operação 1	Operação 2	Tipo 1	Tipo 2
App1	10	0	10	0
App2	8	0	8	0
App3	8	2	10	1
App4	0	10	0	10
App5	0	8	0	8
App6	2	8	1	10

Tabela 5.2: Um exemplo de transformação ponderada.

Do ponto de vista de similaridade, a escolha desta transformação implica considerar como similares as aplicações que utilizam as mesmas features em um número igual de interações.

O gráfico da Figura 5.3 ilustra ilustra os dados ponderados da Tabela 5.2, considerando apenas a frequência do uso das features Operação 1 (eixo horizontal) e Operação 2 (eixo vertical). As instâncias da preparação ponderada formam 2 agrupamentos naturais, através dos quais identifica-se quais clientes utilizam a Operação 1 com maior frequência (App1, App2, App3) ou a Operação 2 com maior frequência (App4, App5, App6).

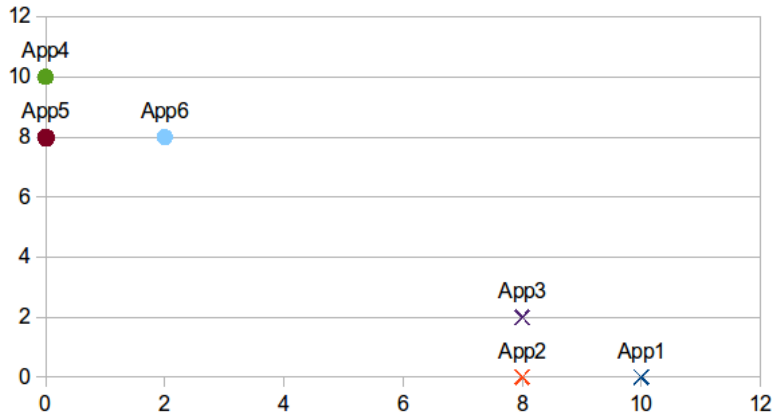


Figura 5.3: Dados de tranformação ponderada.

No contexto de evolução de serviços, o agrupamento usando preparação binária é mais adequado para a identificação dos clientes que serão afetados por certas mudanças, pois aplicações que utilizam um mesmo conjunto de features tendem a pertencer ao mesmo perfil, independentemente da frequência do uso. Por exemplo, considerando os grupos da Figura 5.1 e uma alteração incompatível na Operação 1, é possível identificar perfeitamente os dois grupos de aplicações afetados pela mudança.

Já a preparação ponderada é mais adequada para gerar de perfis para o cálculo o impacto de mudanças, visto que o critério de similaridade ponderado permite que instâncias que usam as mesmas features, mas com frequências distintas, estejam em grupos diferentes. Por exemplo, considerando os grupos da Figura 5.3 seria possível mensurar com maior precisão o impacto causado por alterações incompatíveis sobre aplicações que usam um mesmo conjunto de features. Supondo uma alteração incompatível na Operação 2, e sabendo que clientes que utilizam a Operação 1 com frequência são mais importantes para o provedor do que os que utilizam a Operação 2, verifica-se que o impacto da alteração será baixo, pois o grupo de aplicações mais importantes (App1, App2, App3) será pouco afetado.

Estas duas formas de transformação são utilizadas neste trabalho porque são decisivas para a análise das questões tratadas pelo framework WS-Evolv. Outras duas formas de transformação, comuns para otimização de técnicas de clustering (TAN; STEINBACH; KUMAR, 2006), também foram consideradas no framework.

A primeira transformação adicional tem relação com a redução de dimensionalidade, realizada através da eliminação de features nunca usadas e de tipos de dados irrelevantes, processos realizados durante a etapa de seleção, e descritos na Seção 5.1.1.

A segunda se refere à padronização, e consiste na normalização de dados ponderados através do z-score. O z-score (TAN; STEINBACH; KUMAR, 2006) substitui o valor de um atributo de uma instância pela distância desse valor em relação à mediana dos valores desse atributo em todas as instâncias, em unidades de desvio padrão. Esta transformação

auxilia a reduzir o viés produzido pela presença de atributos com escalas muito diferentes, o que ocorre, por exemplo, quando uma operação é utilizada com frequência muito maior que as demais.

5.2 Clustering

Encontrar o tipo de algoritmo de agrupamento mais adequado aos dados disponíveis é um desafio, pois os dados de uso de serviços não possuem, a priori, qualquer propriedade em particular que permita a identificação da definição mais apropriada de agrupamento e a técnica de agrupamento correspondente.

A abordagem adotada para esse problema é permitir que o usuário do framework parametrize e execute um conjunto pré-selecionado de algoritmos, compare os resultados obtidos através de métricas de validação disponibilizadas pelo framework, e selecione o melhor resultado segundo seus critérios.

O conjunto de algoritmos providos pelo framework foi escolhido com a intenção de obter agrupamentos particionais contendo clusters de diferentes tipos, ou seja, definidos sob diferentes critérios. Foram adotados quatro algoritmos de classes distintas, de acordo com o tipo de clusters resultantes: K-Means (baseado em centróide), Hierárquico aglomerativo (bem separado), DBSCAN (baseado em densidade) e Expectation-Maximization (EM; baseado em distribuição). Os algoritmos são discutidos em maior detalhe na Seção 2.5. O framework pode ser estendido através da inclusão de novos algoritmos de clustering, de acordo com as necessidade do usuário.

O conjunto de dados de entrada é o mesmo para todos os algoritmos, o qual é preparado previamente pelo usuário, conforme detalhado na Seção 5.1. Quanto à parametrização de cada algoritmo, o usuário precisa informar:

- K-Means: número de clusters a ser criado.
- EM: número de clusters a ser criado.
- DBSCAN: número mínimo de pontos e raio utilizados como critério de ligação entre instâncias.
- Hierárquico: número de clusters a ser criado e tipo de ligação entre os clusters.

A definição dos melhores parâmetros para um algoritmo dependem das características do conjunto de dados de entrada, como sua densidade, por exemplo. Como as características do conjunto de dados de uso de um web service não podem ser definidas *a priori*, os melhores parâmetros para um algoritmo só são conhecidos a partir da avaliação do clustering resultante de sua execução. Para auxiliar na seleção dos parâmetros, é necessário realizar experimentos, visando avaliar os clusterings resultantes de diferentes algoritmos e parametrizações, executados sobre dados de uso que representem características comuns a grande parte dos web services. Um conjunto inicial de experimentos foi desenvolvido, e é reportado no Capítulo 6.

A execução de cada algoritmo resulta em um clustering, cujos clusters são determinados pela forma com que os clientes usam as features de um serviço, considerando o critério de similaridade definido pela preparação dos dados de entrada (binária, ponderada), e pelo tipo de cluster característico do algoritmo utilizado.

A execução de todos os algoritmos parametrizados resulta em um conjunto de clusterings. O usuário pode comparar os agrupamentos obtidos, através de métricas de validação internas e externas, e escolher o melhor, segundo seus critérios. Os clusters são então utilizados na construção dos perfis de uso de serviços.

5.3 Validação

No framework proposto, a validação de resultados consiste em avaliar os agrupamentos de clientes de um serviço, gerados por diferentes algoritmos de clustering, comparar os resultados e determinar se um deles atende a um nível de qualidade esperado. A escolha de um clustering satisfatório leva à construção dos perfis de uso a partir dele. Caso contrário, o usuário pode retornar às etapas de preparação de dados e parametrização de algoritmos, a fim de otimizar seus resultados.

A validação de agrupamentos é uma tarefa difícil, pois geralmente não há informação disponível sobre quais as partições esperadas pelo usuário, ou, no contexto to trabalho, quais instâncias deveriam fazer parte de um mesmo perfil de uso. Por isso, escolher o algoritmo e parametrização mais adequados para um conjunto de dados é um desafio.

O índice F-Measure por contagem de pares (PFITZNER; LEIBBRANDT; POWERS, 2009) foi integrado ao framework para auxiliar na avaliação da qualidade de um agrupamento gerado em relação a um agrupamento esperado. O índice pode ser utilizado quando o conjunto de dados possui uma rotulação, mesmo que de parte das instâncias, informando quais clientes deveriam pertencer aos mesmos perfis de uso. A F-Measure pode ser utilizada em gabaritos incompletos, pois permite comparar agrupamentos com número de clusters distintos. Nesse caso, somente as instâncias rotuladas são utilizadas no cálculo do índice.

O valor de F-Measure do clustering reflete quão homogêneas são os clientes dentro de cada cluster e quão heterogêneo são os clusters entre si. Em um clustering com valor de F-Measure igual a zero, nenhum agrupamento possui dois clientes de um mesmo perfil, o que significa que nenhum perfil foi identificado. Em um clustering com valor de F-Measure igual a 1, cada cluster possui apenas instâncias de um mesmo perfil, e nenhum cliente de um mesmo perfil está presente em mais de um cluster. Para o usuário, isso significa que, quanto maior o valor de F-Measure, melhor é o agrupamento obtido pelo algoritmo. Mais detalhes sobre a F-Measure são encontrados na Seção 2.5.2.

Em situações reais, dificilmente serão encontrados gabaritos sobre os perfis de uso de clientes, pois é para *descobrir* os grupos desconhecidos pelo usuário que as técnicas de clustering são utilizadas. Nesse cenário, é preciso valer-se do conhecimento de um especialista para a validação dos agrupamentos, contando com auxílio de índices de avaliação internos.

Os índices de avaliação internos, apesar de não tão precisos quanto os internos, fornecem um bom referencial para a comparação da qualidade dos clusterings criados por diferentes algoritmos e parametrizações. Dois índices internos de avaliação foram implementados no framework, sendo eles o índice de silhueta (ROUSSEEUW, 1987) e o índice SD (HALKIDI; VAZIRGIANNIS; BATISTAKIS, 2000). Ambos os índices foram selecionados por apresentarem um bom desempenho sob diferentes possíveis características de um conjunto de dados, como ruído e clusters de tamanhos e densidades diferentes (LIU et al., 2010).

O índice de silhueta quantifica a coesão e separação dos clusters, por meio do cálculo da média de distâncias entre instâncias de um mesmo cluster e de clusters distintos. Por

utilizar distâncias intra e inter-cluster como critérios de qualidade, esse índice se torna mais adequado para a avaliação de clusters bem separados. Os valores do índice de silhueta variam entre -1 e 1, sendo -1 o pior caso, e 1 o melhor. Um valor negativo indica que, em média, a distância intra-cluster é maior que inter-cluster. Para selecionar bons clusterings, de acordo com os critérios do índice, o usuário deve aceitar somente valores de silhueta positivos, priorizando os resultados com o maior valor.

O índice SD é semelhante ao índice de silhueta, porém calcula a coesão de um cluster com base na dispersão das instâncias, e a separação baseia-se nas distâncias entre centroides. Segundo esses critérios, o índice é mais adequado para a avaliação de clusters baseados em protótipo. O SD possui um valor mínimo de 0, sem limite para o valor máximo, sendo zero o valor ótimo. Para selecionar bons clusterings, de acordo com os critérios do índice, o usuário deve priorizar os resultados com os menores valores de SD.

Para que possam prover uma compreensão a respeito do melhor clustering, a avaliação feita a partir dos índices internos precisa corresponder à avaliação feita a partir dos índices externos. Ou seja, dado que a F-Measure quantifica a similaridade do clustering gerado em relação ao clustering esperado, o melhor clustering de acordo com a F-Measure deve ser o melhor clustering segundo os índices de silhueta e SD. A fim de verificar o desempenho dos índices internos na presença de dados de uso de serviços, foram realizados experimentos nos quais os três índices são comparados quanto à avaliação de clusterings de clientes (Capítulo 6).

Após validado, o clustering selecionado é utilizado na construção dos perfis de uso.

5.4 Construção de perfis de uso

Agrupamentos e perfis de uso diferem quanto a sua natureza. Os agrupamentos contém somente as features que podem ser usadas para distinguir entre grupos de aplicações, como resultado do passo de preparação (Seção 5.1). Já os perfis são uma representação enriquecida dos grupos de aplicações (Figura 4.2), os quais incluem todas as features usadas, juntamente de métricas que indicam a importância do grupo de aplicações e das features utilizadas pelo grupo. Neste trabalho foram definidas duas métricas (número de interações por aplicação e por feature), mas outras podem também ser adotadas.

Considerando o exemplo da Figura 5.1, as features Op1, Op2 e T2 são submetidas como entrada para o algoritmo de agrupamento. Um agrupamento resultante pode indicar que somente Op1 é utilizada, sem o parâmetro opcional P2, do tipo T2. Desta forma, o perfil conteria as features Service, Op1 e T1 (parâmetros obrigatório de P1), com suas respectivas métricas. As métricas devem ser recuperadas diretamente da Base de Dados de Uso, pois durante as atividades de preparação de dados, diversas features são excluídas do conjunto de dados utilizado na mineração, além de terem seus valores de uso transformados.

Cada perfil de uso corresponde a um cluster no clustering validado pelo usuário do framework. O primeiro passo para a construção de um perfil consiste em calcular as métricas relacionadas às aplicações cliente, as quais correspondem ao relacionamento 'MétricaAplicação' na Figura 4.2. Isto significa recuperar, através da Base de Dados de Uso, o número total de interações realizadas pelas aplicações contidas no cluster, durante o período selecionado na preparação dos dados.

Na segunda etapa da construção de um perfil de uso, as métricas relacionadas à cada feature do perfil são calculadas, as quais correspondem ao relacionamento 'MétricaFeature' na Figura 4.2. Isto significa recuperar, para cada aplicação cliente contida em um

cluster, as features utilizadas e o número de interações envolvidas no período selecionado, através de consultas à Base de Dados de Uso. A forma como o uso das features é recuperado depende do tipo da feature contida no cluster. Para possibilitar essa diferenciação, a feature utilizada pela aplicação é recuperada do repositório de versões, por meio do identificador da versão da feature contido nos dados preparados.

Para calcular as métricas quando a feature for operação ou serviço, recupera-se na Base de Dados de Uso o número de interações nas quais a feature é utilizada pela aplicação, no período especificado durante a preparação dos dados. Isto é feito pela contagem dos registros do relacionamento 'Interação Feature' entre interações da aplicação e feature em questão 4.3. Esta recuperação é feita somente para serviços e operações utilizados pela aplicação, os quais estão presentes no conjunto de dados preparados e apresentam um valor de uso diferente de zero.

A necessidade de incluir um tipo no perfil de uso, ao contrário das operações e serviços, não pode ser verificada a partir do conjunto de dados preparados. Muitos tipos utilizados pela aplicação não estão presentes nos dados preparados para o clustering, devido à eliminação de dados desnecessários durante a etapa de seleção, o que ocorre, por exemplo, quando o uso do tipo é idêntico para todas as aplicações. Desta forma, é preciso verificar o uso de cada tipo encontrado na hierarquia de parâmetros de uma operação.

A partir da operação recuperada do repositório de versões, precorre-se recursivamente a árvore de seus parâmetros, consultando na base de uso para recuperar o número de interações nas quais o tipo é utilizado. Essa consulta consiste em contar as interações que possuem pelo menos um registro de relacionamento 'Interação Parâmetro' com a feature em questão 4.3.

O processo de recuperação do uso de tipos é otimizado de três maneiras: a) a recursão em um ramo da árvore de parâmetros é interrompida quando o tipo de um parâmetro não é usado, pois os parâmetros abaixo dele também não poderão ter sido usados; b) somente se consulta o número de interações na base de dados dos tipos que ainda não foram consultados, visto que um tipo pode ser referenciado por mais de um parâmetro; c) não se consulta o uso de tipos referenciados por parâmetros obrigatórios de uma operação, ou provenientes de um ramo de parâmetros também obrigatórios, pois o uso da operação implica no uso do tipo. O número de interações nas quais o tipo é utilizado é igual ao número de interações nas quais a operação é utilizada.

O pseudo algoritmo da Figura 5.4 descreve esse procedimento, a ser repetido para cada cluster. A partir das instâncias do cluster recebido por parâmetro, o algoritmo computa, para cada aplicação do cluster, o número total de interações por ela realizadas (linha 6); para cada atributo do cluster recupera a representação da feature correspondente no repositório de versões (linha 8); caso a feature seja uma operação ou serviço, recupera o número de interações nas quais a feature é utilizada pela aplicação (linhas 10-11), e associa esse dado à aplicação (linha 12); caso a feature seja uma operação, percorre recursivamente sua hierarquia de parâmetros, coletando o número de interações nas quais são utilizados cada tipo de dado e associando esses dados à aplicação (linhas 14-16); Por fim, são sumarizados no perfil de uso o número de interações nas quais a aplicação utiliza cada feature (linhas 20-22).

Os perfis de uso criados podem ser, então, armazenados na Base de Dados de Uso 4.3, onde poderão ser consultados posteriormente.

```
 1: procedure CONSTRUIRPERFIS(cluster, dataInicial, dataFinal)
 2:     perfil ← Perfil()
 3:     instancias ← cluster.instancias
 4:     perfil.apps ← cluster.apps
 5:     for all app from perfil.apps do
 6:         app.numeroInteracoes ← bdUso.numeroInteracoes(dataInicial, dataFinal, app)
 7:         for all attr from cluster.atributos do
 8:             ft ← repositorioVersoes.feature(attr.idVersao)
 9:             if (ft is Operacao or ft is Servico) and instancias(app)(attr) > 0 then
10:                 usoFeature ← bdUso.numeroInteracoes(
11:                         dataInicial, dataFinal, app, ft)
12:                 app.mapaUso(ft) ← usoFeature
13:                 if ft is Operacao then
14:                     app.mapaUso ← percorrerParametros(
15:                         dataInicial, dataFinal, app,
16:                         ft, app.mapaUso, usoFeature)
17:                 end if
18:             end if
19:         end for
20:         for all (feature, valorUso) from app.mapaUso do
21:             perfil.mapaUso(feature) ← perfil.mapaUso(feature) + valorUso
22:         end for
23:     end for
24:     return p
25: end procedure
```

Figura 5.4: Algoritmo de construção de perfis de uso.

5.5 Considerações finais

Neste capítulo foi apresentado o processo para a descoberta de perfis de uso de serviços utilizado pelo framework proposto, o qual tem por objetivo reduzir a complexidade do processo de KDD, do ponto de vista do usuário final. Esse objetivo é atingido a partir das pré-definição das tarefas do KDD.

De uma forma geral, o framework permite que o usuário analise o uso do serviço sem a necessidade de conhecer detalhes sobre a forma como os perfis de uso são descobertos. O usuário não precisa saber como funciona o versionamento dos serviços em baixo nível, como os dados de uso são armazenados e acessados, como precisam ser preparados, quais dados precisam ser mantidos para a obtenção de bons resultados e como os clusters podem ser convertidos em perfis de uso úteis. Além disso, estas tarefas, quando executadas individualmente, exigem o uso de diversas ferramentas e de inúmeras conversões entre formatos de entrada e saída, o que é abstraído pelo framework proposto.

A forma proposta para a seleção permite que o usuário customize as consultas ao banco de dados de uso, escolhendo entre diferentes níveis de granularidade de dados, de acordo com suas necessidades de análise, e lide com o fator temporal da análise do uso, ao permitir filtrar somente interações de um determinado período. A seleção também otimiza os algoritmos de clustering, sem a necessidade de intervenção do usuário, ao excluir do conjunto de dados as features que podem causar viés nos agrupamentos.

O framework permite que questões relacionadas à evolução de serviços sejam tratadas com mais facilidade, pois dispõe de duas formas de transformação fundamentais para

a derivação de perfis de uso voltados às análises de compatibilidade e de impacto de mudanças. Através dos diversos algoritmos de clustering e das métricas de avaliação disponibilizadas, o framework também permite que os usuários escolham o agrupamento mais adequado de acordo com seus critérios de qualidade.

6 EXPERIMENTOS

Neste capítulo são apresentados experimentos realizados com o framework proposto, utilizando-se diferentes formas de preparação de dados, algoritmos de agrupamento, parametrizações e métricas de validação supervisionadas e não supervisionadas. Os experimentos têm por objetivo avaliar a capacidade do framework de produzir perfis de uso de serviço úteis, e também prover indicadores sobre as melhores parametrizações a serem utilizadas, de acordo com as necessidades de um analista.

6.1 Protótipo

Foi construída uma prova de conceito para o Gerenciador de Perfis, o qual apoiou a condução dos experimentos. Os componentens implementados foram o Carregador de Dados e o Gerador de Perfis (Figura 4.1). O Monitor de Interações não foi implementado, pois o log de requisições SOAP foi gerado por simulação (Seção 6.3). O protótipo do Gerenciador de Perfis foi implementado na linguagem Scala [1].

O Carregador de Dados implementado recebe um arquivo texto contendo uma série de requisições SOAP, e as processa através de um parser de *streaming* de XML, devido ao tamanho elevado do arquivo de log. A Base de Dados de Uso na qual as interações são carregadas é um banco relacional gerenciado pelo PostgreSQL [2].

No Gerador de Perfis, os dados são selecionados, transformados e armazenados em um arquivo no formato ARFF [3]. A preparação é totalmente realizada pelo framework, com exceção do filtro de z-score, proveniente do Weka (HALL; FRANK; HOLMES, 2009). Os algoritmos de clustering utilizados também provém do Weka. As métricas de avaliação e a construção de perfis foram implementadas como parte do framework.

O Gerenciador de Perfis interage com o Gerenciador de Versões do WS-Evolv, implementado em Java (YAMASHITA, 2012). O modelo de versionamento de features é representado na forma de um grafo e armazenado no repositório de versões, o qual é gerenciado pelo OrientDB [4], um banco de dados orientado a grafos (ALVES; BECKER, 2013).

6.2 Visão Geral dos Experimentos

O objetivo dos experimentos realizados é validar o framework proposto e avaliar sua capacidade de produzir perfis de uso de serviço úteis, a partir de um log de interações,

[1] http://www.scala-lang.org/
[2] http://www.postgresql.org/
[3] http://weka.wikispaces.com/ARFF
[4] http://www.orientdb.org/

com um mínimo de parametrização e avaliação de um especialista. Dada a ausência de logs reais de interação, foram gerados logs sintéticos através da simulação de requisições de clientes a um serviço, simulando a implementação de um serviço real, a versão 767 do serviço *eBay Trading* [5].

A interface do eBay Trading descreve mais de 150 operações, as quais estão relacionadas a centenas de tipos de dados. A documentação do serviço apresenta casos de uso típicos, os quais organizam as operações em workflows, que podem ser usados de forma independente ou combinados para gerar aplicações. Assume-se que as diferentes formas nas quais esses workflows podem ser combinados são capazes de caracterizar conjuntos de aplicações com comportamentos similares.

As requisições foram simuladas de forma a representar grupos de clientes pertencentes a perfis de uso pré-definidos, com um certo nível de ruído. O log foi processado e carregado na Base de Dados de Uso, da qual foram extraídos conjuntos de dados distintos, variando o nível de detalhe das features contidas (operações ou tipos) e o tipo de representação (binária ou ponderada). Foram desenvolvidos experimentos utilizando quatro algoritmos de agrupamento do Weka (HALL; FRANK; HOLMES, 2009): K-Means, EM, DBSCAN e hierárquico aglomerativo (ligação pela média). Diversos experimentos foram realizados, explorando os valores possíveis para os parâmetros de cada algoritmo, dos quais são reportados apenas aqueles com os melhores resultados. Mais informações sobre os algoritmos utilizados no framework e seus parâmetros são encontradas na Seção 5.2.

Como resultado dos experimentos, espera-se gerar clusters que correspondam aos padrões de uso injetados, e identificar os melhore(s) algoritmo(s) de clustering e parametrização para cada tipo de conjunto de dados. O critério usado para avaliar os resultados se baseia em três aspectos: o número de clusters gerados; o número de perfis distintos representados pelos clusters, considerando que um cluster representa o perfil que nele predomina (aquele com maior número de aplicações); e a F-Measure por contagem de pares (Seção 2.5.2).

Por fim, também foram realizados experimentos com índices de avaliação internos, os quais comparam os resultados da avaliação supervisionada com os resultados da avaliação não supervisionada. O índice de silhueta (ROUSSEEUW, 1987) e o índice SD (HALKIDI; VAZIRGIANNIS; BATISTAKIS, 2000) foram calculados a partir dos agrupamentos resultantes, e comparados com os valores da F-Measure. Para prover uma compreensão que auxilie o usuário na validação de resultados, a avaliação do melhor clustering, segundo os índices não supervisionados, precisa corresponder à avaliação segundo a F-Measure.

6.3 Dados utilizados

Foi adotada a versão 753 do serviço eBay Trading e 7 workflows que representam casos de uso típicos, presentes na documentação da API[6]. Cada workflow descreve uma série de operações executadas sequencialmente para a realização de uma determinada tarefa.

A feramenta para testes de web services JMeter[7] foi utilizada para gerar requisições às operações pertencentes aos workflows, de acordo com determinadas probabilidades.

[5] http://go.developer.ebay.com/developers/ebay/products/trading-api
[6] developer.ebay.com/DevZone/XML/docs/WebHelp
[7] jmeter.apache.org

As probabilidades de execução dos workflows geram uma variação nos dados de uso de clientes de um mesmo perfil. A Figura 6.1ilustra o perfil simulador 'Buyer', o qual possui 5% de probabilidade de executar o workflow 'Get Token' e 95% de probabilidade de executar o workflow 'Buy Item'.

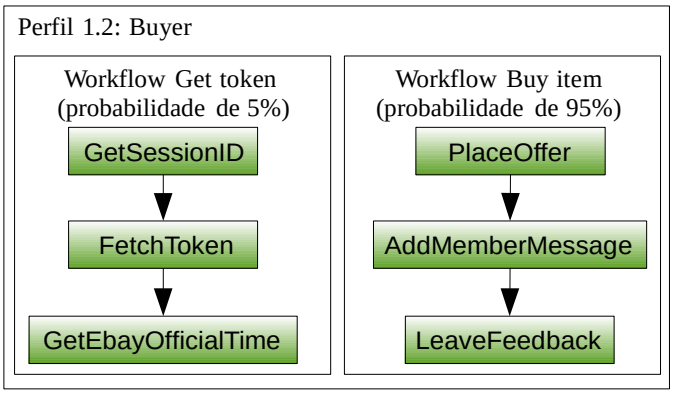


Figura 6.1: Exemplo de perfil simulado.

Como sumarizado pela Tabela 6.1, foram simuladas 525 aplicações cliente, distribuídas em 6 perfis, as quais executaram 448.703 requisições a 42 operações distintas. O diagrama de Venn na Figura 6.2 mostra o relacionamento entre perfis, destacando suas operações em comum. Três dos perfis são sub-conjuntos de outros (P1.2, P1, P2), e dois perfis (P6 e P8) usam o mesmo conjunto de operações, mas com frequências distintas.

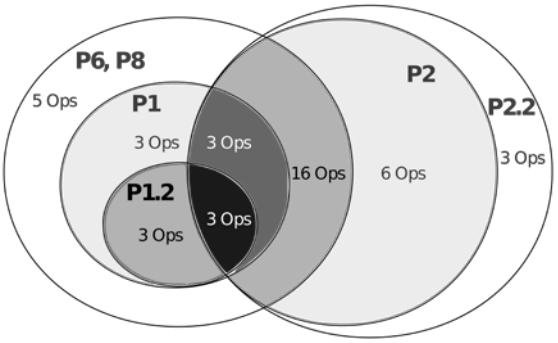


Figura 6.2: Perfis simulados e suas intersecções.

Perfil Injetado	Aplicações	Operações	Requisições
P1	100	12	89,996
P1.2	50	6	44,017
P2	100	28	82,538
P2.2	25	31	20,841
P6	125	33	103,232
P8	125	33	108,079
Total (distintos)	525	42	448,703

Tabela 6.1: Sumarização dos dados simulados.

Os logs gerados consistem em mensagens SOAP usando encoding literal, as quais foram pré-processadas e carregadas na Base de Dados de Uso. Nas seções seguintes são reportados três experimentos, com base em conjuntos de dados preparados de diferentes formas. Nesses experimentos, também foram adicionados sistematicamente diferentes níveis de ruído aos dados preparados, na forma de aplicações com valores randômicos para o uso das features. As aplicações ruidosas foram inseridas nas proporções de 10% e 30% em relação ao número original de aplicações (sem ruído). Os objetos de dados foram rotulados com suas respectivas classes (perfil ou ruído), de forma que os clusters pudessem ser validados utilizando uma métrica supervisionada.

6.4 Experimentos com preparação binária

Nesta seção são apresentados dois experimentos com conjuntos de dados de uso selecionados em granularidade de operações e de tipos, utilizando preparação binária.

6.4.1 Experimentos com dados de uso de operações e transformação binária

O primeiro conjunto de dados envolve somente o uso operações, transformados de em uma representação binária. O Perfil P8 foi excluído do conjunto de dados, por ser idêntico ao Perfil P6 com relação ao uso binário das operações. Os resultados são exibidos na Tabela 6.2, a qual exibe o número de clusters (coluna C), o número de perfis distintos por eles representados (coluna P) e o valor de F-Measure (coluna F-M).

O número de clusters e perfis deveria, idealmente, ser o mesmo, ou seja, um cluster para cada perfil pré-definido. Um número de clusters maior que o número de perfis significa que as aplicações pertencentes a um mesmo perfil foram distribuídas em diferentes clusters (i.e. os clusters são redundantes). Um número de perfis identificados inferior ao esperado indica que um ou mais clusters misturam aplicações de perfis diferentes, ou aplicações de um perfil com um excesso de aplicações ruidosas, de tal forma que um ou mais perfis não são possuem um cluster que os represente. O valor de F-Measure igual a 1 indica um agrupamento perfeito das aplicações, sendo que quanto menor o valor pior a qualidade dos grupos, sendo zero o pior caso.

Em geral, o algoritmo hierárquico, utilizando ligação pela média, apresentou os melhores resultados. Os clusters corresponderam exatamente aos perfis simulados, na presença de ruído. Considerando o conjunto de dados sem ruído, 3 instâncias do perfil P2.2 foram agrupados no cluster representativo do perfil P2. Os algoritmos K-Means e EM foram os mais sensíveis ao ruído, não sendo capazes de detectar subgrupos de aplicações. Eles misturaram aplicações dos perfis P1/P1.2 e P2/P2.2, além de criar clusters para dados ruidosos. O agrupamento gerado pelo algoritmo DBSCAN correspondeu quase exatamente aos perfis simulados, porém inserindo algumas aplicações em grupos incorretos, por conta destas possuírem variações pouco significativas.

6.4.2 Experimentos com dados de uso de tipos e transformação binária

O segundo conjunto de dados diferencia-se por incluir tipos de parâmetros opcionais, em adição às operações. Dado que os logs simulados não cobrem o uso de tipos opcionais, foram injetados 2 novos perfis nos dados preparados:

- P6T, que possui as mesmas operações do perfil P6 mais 10 tipos randomicamente selecionados;

	Sem ruído			10% ruído			30% ruído		
Algoritmo	C	P	F-M	C	P	F-M	C	P	F-M
K-Means	5	5	1.00	5	5	1.00	5	4	0.94
EM	5	5	1.00	5	4	0.94	5	4	0.94
DBSCAN	5	5	0.97	5	5	0.97	5	5	0.97
Hierárquico	5	5	0.99	5	5	1.00	5	5	1.00

Tabela 6.2: Resultados de clustering para preparação binária em granularidade de operações.

- P2T, que possui as mesmas operações do perfil P2, com a adição de 10 tipos randomicamente selecionados.

Como exibido na Tabela 6.3, o algoritmo hierárquico novamente apresentou os melhores resultados. O algoritmo K-Means não foi capaz de detectar perfis com relações de subconjunto, misturando os perfis P2/P2.2/P2T, P1/P1.2 e P6/P6T. O algoritmo EM não pôde distinguir entre os perfis P2 e P2.2 na presença de ruído. O DBSCAN foi capaz de detectar todos os perfis distintos, mas agrupando no cluster representativo de P1 aplicações pertencentes a grupos vizinhos (P2, P2T, P2.2, P6 e P6T).

Os experimentos mostram evidências de que o algoritmo hierárquico aglomerativo, usando ligação pela média, apresenta bons resultados para dados binários. Em um primeiro momento, o DBSCAN também aparenta ser uma boa escolha, porém ele requer um processo extensivo de tentativa e erro para que se encontre a melhor parametrização, a qual é reportada nos resultados (mínimo de pontos = 6, epsilon = 0,5).

Esses resultados podem ser influenciados pelo número de operações em comum na intersecção entre os perfis. Não obstante, em situações reais, nossa hipótese é de que grande parte dos serviços possuem um conjunto de operações utilizado em comum por muitos perfis (se não todos eles), e que as técnicas de clustering precisam ser capazes de lidar com esse grau de similaridade entre perfis. Essa hipótese precisa ser confirmada por uma experimentação extensiva, particularmente com dados de uso de serviços reais.

	Sem ruído			10% ruído			30% ruído		
Algoritmo	C	P	F-M	C	P	F-M	C	P	F-M
K-Means	7	6	0.94	7	5	0.86	7	6	0.95
EM	7	7	1.00	7	6	0.95	7	5	0.86
DBSCAN	7	7	0.94	7	7	0.94	7	7	0.94
Hierárquico	7	7	0.99	7	7	1.00	7	7	1.00

Tabela 6.3: Resultados de clustering para preparação binária em granularidade de tipos.

6.5 Experimentos com preparação de dados ponderada

Nestes experimentos utilizou-se dados envolvendo o uso de operações, transformados de acordo com a representação ponderada. As aplicações dos perfis P6 e P8 foram utilizadas, diferentemente dos experimentos com dados binários, pois seus comportamentos diferenciam-se em relação à frequência de uso das operações. O número de interações foi normalizado utilizando o z-score, transformação que pode reduzir o efeito do ruído para alguns algoritmos de clustering. O z-score substitui o valor de um atributo de uma

instância pela distância desse valor em relação à mediana dos valores do atributo em todas as instâncias, em unidades de desvio padrão.

O algoritmo hierárquico aglomerativo utilizando ligação pela média apresentou os melhores resultados. Como exibido na Tabela 6.4, o algoritmo foi capaz de detectar os seis perfis em todos os casos, apresentando os maiores valores de F-Measure. No entanto, alguma poucas aplicações do perfil P8 foram erroneamente atribuídas aos grupos representativos dos perfis P1 e P1.2.

Os algoritmos K-Means e EM não foram capazes de distinguir os perfis que se diferem pela frequência do uso, colocando nos mesmos grupos as aplicações dos perfis P6 e P8. Eles também não puderam detectar relações de subconjunto: o K-Means uniu os perfis P1 e P1.2 em um único cluster, e o EM uniu P2 e P2.2.

O algoritmo DBSCAN, usando as duas melhores parametrizações encontradas, falhou em criar um cluster por perfil. Com mínimo de pontos = 3, o perfil P2 foi dividido em dois grupos, um dos quais misturado com aplicações dos perfis P2.2, P6 e P8. Com mínimo de pontos = 7, o algoritmo encontrou 6 perfis distintos, mas criou um cluster com as aplicações de P2 misturado a muitas aplicações próximas dos demais perfis, o que é um problema característico da abordagem baseada em densidade.

Como mencionado anteriormente, os dados simulados com base em workflows de operações e composições de workflows geram perfis semelhantes entre si, ou seja, não bem separáveis, característica enfatizada com a injeção de ruído aos dados. As consequências observadas nos agrupamentos do algoritmo K-Means são a redução do número de perfis identificados (4 dos 6 esperados) e baixo valor de F-Measure (0,52 no pior caso), evidenciando que esse algoritmo é particularmente suscetível a dados ruidosos. Com base nas premissas sobre as características dos padrões de uso de serviços, é possível afirmar que o K-Means não representa uma técnica adequada para o agrupamento de dados de uso ponderados pela frequência.

Este experimento também evidenciou que o algoritmo hierárquico aglomerativo, utilizando ligação pela média, também é capaz de produzir resultados de agrupamento consistentes para dados ponderados. É preciso notar que a diferenciação de grupos baseada em dados ponderados é um problema bem mais desafiador que o agrupamento através de dados binários.

	Sem ruído			10% ruído			30% ruído		
Algoritmo	C	P	F-M	C	P	F-M	C	P	F-M
K-Means	6	4	0.78	6	4	0.72	6	4	0.52
EM	6	5	0.88	6	5	0.95	6	5	0.95
DBSCAN (0.5,3)	7	6	0.93	7	6	0.93	7	6	0.93
DBSCAN (0.5,7)	6	6	0.89	6	6	0.89	6	6	0.89
Hierárquico	6	6	0.97	6	6	0.98	6	6	0.98

Tabela 6.4: Resultado de clustering para preparação ponderada em granularidade de operações.

6.6 Avaliação de índices de validação internos

Nas Seções 6.4 e 6.5, foram avaliados os agrupamentos criados através de diferentes algoritmos em relação aos agrupamentos esperados, de acordo com um *golden standard*, i.e. os perfis injetados. No entanto, em situações reais, dificilmente existirão expectativas

a priori a respeito dos perfis de uso a serem encontrados. Por isso, os experimentos são completados com uma comparação entre os valores de F-Measure previamente apresentados (Tabelas 6.2, 6.3 e 6.4), e os valores dos índices internos de silhueta e SD, calculados sobre os mesmos agrupamentos.

A premissa utilizada na avaliação dos resultados é de que, para ser capaz de auxiliar na escolha dos melhores algoritmos, parametrizações e agrupamentos resultantes, a avaliação do clustering segundo um índice interno precisa corresponder à avaliação segundo a F-Measure. Ou seja, os índices de silhueta e SD precisam apresentar os melhores valores para os clusterings que possuem os melhores valores de F-Measure.

Como apresentado na Seção 2.5.2, o índice de silhueta é direcionado à avaliação de agrupamentos particionais bem separados, e o índice SD é direcionado à avaliação de agrupamentos particionais baseados em protótipo. De acordo com Liu et al. (2010), os índices de silhueta e SD lidam bem com: a) ruído, o que, em nosso domínio, corresponde às aplicações cliente com comportamento randômico; b) clusters de diferentes densidades, representadas por perfis com número de aplicações muito distintos; e c) clusters de diferentes tamanhos, os quais ocorrem quando aplicações de um mesmo perfil variam muito tanto em relação às features utilizadas (dados binários) ou ao número de requisições (dados ponderados).

Os resultados são exibidos na Tabela 6.5, considerando os três conjuntos de dados previamente apresentados, com e sem adição de ruído. O valor ótimo para o coeficiente de silhueta é 1, onde 0 significa que nenhuma tendência de clustering foi detectada, ou seja, o clustering não reflete os padrões esperados. O SD index faz a correspondência contrária, quanto menor seu valor, melhor o clustering.

Observa-se que, na ausência de ruído, ambos os índices SD e silhueta (Silh.) correspondem aos valores de F-Measure (F-M) previamente discutidos, independentemente da preparação de dados utilizada (binária ou ponderada) e da granularidade dos dados. Isto significa que estas três métricas apresentam uma concordância a respeito da qualidade dos agrupamentos para conjuntos de dados sem ruído. Isso também revela que os clusters resultantes são razoavelmente bem separados, apesar das similaridades entre eles. Esses resultados, no entanto, não são observados para dados ruidosos.

Na presença de dados ruidosos, a avaliação de melhor agrupamento sob os índices de silhueta e F-Measure é diferente em todos os experimentos, o que também ocorre na comparação dos índices SD e F-measure. Por exemplo, na Tabela 6.5.B, o melhor valor de silhueta para o conjunto de dados com 30% de ruído (0,6622) corresponde ao pior agrupamento, de acordo com o valor de F-Measure (0,86). De forma inversa, o melhor agrupamento de acordo com a F-Measure (1,00) apresenta o pior valor de silhueta (0,4342) e o segundo pior valor para o índice SD (1,4519).

Esses resultados podem ser parcialmente explicados pela incapacidade dos índices utilizados de lidar com clusters muito próximos uns aos outros, i.e. que não são bem separados (LIU et al., 2010), dado que assumem a premissa de que bons clusterings devem apresentar esta característica. Com a adição de ruído, os melhores clusterings são aqueles criados pelo algoritmo hierárquico, o qual desobedece a premissa de clusters bem separados, ao distribuir as aplicações ruidosas de forma homogênea entre os clusters. Isto torna impossível para ambos os índices identificar o melhor resultado.

Para avaliar a utilidade dos índices internos para a escolha do melhor clustering, foram comparados os rankings de algoritmos derivados dos valores dos índices de silhueta, SD e F-Measure. Quanto maior a similaridade entre um ranking derivado de F-Measure e o ranking derivado de um índice interno, maior a utilidade do índice interno em questão.

A. Dados binários na granularidade de operações									
	Sem ruído			10% ruído			30% ruído		
Algoritmo	Silh.	SD	F-M	Silh.	SD	F-M	Silh.	SD	F-M
K-Means	0.9967	0.9450	1.00	0.7366	1.1552	1.00	0.5769	1.0529	0.94
EM	0.9667	0.9450	1.00	0.8541	0.8402	0.94	0.7142	0.8281	0.94
DBSCAN	0.9340	0.9618	0.97	0.6525	1.0059	0.97	0.4589	0.8226	0.97
Hierárquico	0.9576	0.9583	0.99	0.7214	1.1654	1.00	0.4588	1.5094	1.00
B. Dados binários na granularidade de tipos									
	Sem ruído			10% ruído			30% ruído		
Algoritmo	Silh.	SD	F-M	Silh.	SD	F-M	Silh.	SD	F-M
K-Means	0.9456	0.9326	0.94	0.7473	1.0015	0.86	0.5283	0.9352	0.95
EM	0.9676	0.9080	1.00	0.8504	0.7678	0.95	0.6622	1.5998	0.86
DBSCAN	0.9004	0.9495	0.94	0.6811	1.0512	0.94	0.5115	1.0814	0.94
Hierárquico	0.9605	0.9252	0.99	0.7000	1.1465	1.00	0.4342	1.4519	1.00
C. Dados ponderados na granularidade de operações									
	Sem ruído			10% ruído			30% ruído		
Algoritmo	Silh.	SD	F-M	Silh.	SD	F-M	Silh.	SD	F-M
K-Means	0.3141	1.6136	0.78	0.4787	0.6755	0.72	0.3002	1.4660	0.52
EM	0.4035	1.6986	0.88	0.4276	0.7579	0.95	0.3771	0.7158	0.95
DBSCAN (0.5,3)	0.3952	0.6576	0.93	0.3037	0.7122	0.93	0.2312	0.7144	0.93
DBSCAN (0.5,7)	0.3976	0.7030	0.89	0.3287	0.7079	0.89	0.2377	0.6906	0.89
Hierárquico	0.4662	0.6459	0.97	0.3731	0.7349	0.98	0.2553	0.9174	0.98

Tabela 6.5: Resultados dos índices de validação interna.

Para calcular essa similaridade entre rankings utilizou-se a correlação ρ de Spearman. A correlação ρ é definida pela mesma fórmula da correlação de Pearson, porém ela é calculada sobre os rankings derivados de cada variável.

Na Tabela 6.6 são apresentados os coeficientes de correlação entre os rankings derivados dos valores de F-Measure e dos rankings derivados dos índices de silhueta e SD, para os diferentes experimentos. Quando a correlação é igual a 1, há uma correspondência perfeita entre os rankings; quando igual a 0, não existe correlação entre os rankings; quando -1, existe uma correspondência inversamente perfeita entre os rankings.

Os valores de correlação corroboram com a hipótese de que os índices internos utilizados não podem ser utilizados para identificar o melhor clustering, quando o conjunto de clusters não é bem separado. Nos experimentos binários sem ruído, ambos os índices apresentam uma correspondência quase perfeita em relação à F-Measure, porém, na presença de ruído, apresentam nenhuma correlação ou uma correlação inversa. O mesmo se repete com o experimento ponderado, com a diferença de que o índice SD apresentou correlação maior que o índice de silhueta, na ausência de ruído.

Estes experimentos proveem evidências de que os índices de silhueta e SD são adequados para a avaliação de clusters bem separados, e inadequados para clusters que não possuem esta característica, o que foi exposto através de experimentos com dados ruidosos, nos quais as distâncias entre grupos foram reduzidas.

O problema supracitado pode impedir o uso dos índices de validação interna, apesar deles apresentarem bons resultados na ausência de ruído. Para alcançar conclusões mais sólidas a esse respeito, se faz necessária a realização de mais experimentos com

Experimento	Ruído	Silhueta	SD
Binário (operações)	-	0.9438798074	1
Binário (operações)	10%	-0.4045199175	-0.9438798074
Binário (operações)	30%	-0.9438798074	-0.4045199175
Binário (operações e tipos)	-	0.9438798074	0.9438798074
Binário (operações e tipos)	10%	0	-0.4
Binário (operações e tipos)	30%	-0.8	0.4
Ponderado	-	0.6	0.9
Ponderado	10%	-0.2	-0.9
Ponderado	30%	0.1	0
Média		0.0265955433	0.0661644536

Tabela 6.6: Correlação ρ de Spearman entre índices de validação internos e F-Measure.

dados de uso reais de serviços, para que se verifique a validade das premissas utilizadas na construção dos dados produzidos por simulação e utilizados nos experimentos. Mais experimentos também precisam ser realizados para verificar a utilidade dos índices na determinação do número ideal de clusters para um conjunto de dados. Segundo Halkidi, Vazirgiannis e Batistakis (2000), nesse tipo de avaliação o índice SD provê bons resultados, independentemente do algoritmo utilizado.

6.7 Considerações finais

Os objetivos dos experimentos apresentados neste capítulo foram validar a capacidade do framework proposto de produzir perfis de uso de serviço úteis, e prover indicadores sobre as melhores parametrizações a serem utilizadas nele.

Primeiramente, a implementação do protótipo e seu uso nos experimentos ratificam o framework como proposto, o qual permitiu a execução do processo completo de KDD, desde o processamento de logs de mensagens SOAP à descoberta de perfis de uso de serviços, através de tarefas pré-definidas e parametrizáveis.

Os resultados obtidos nas Seções 6.4 e 6.5 validam os perfis de uso, descobertos a partir de uma parametrização mínima. Dados os conjuntos de dados selecionados e preparados segundo os métodos propostos, verificou-se que o algoritmo hierárquico aglomerativo com ligação pela média apresentou resultados quase ótimos, tanto para dados binários quanto ponderados, o que torna essa parametrização promissora para a análise de dados reais de uso.

Outras conclusões dizem respeito aos algoritmos e parametrizações que não apresentaram bons resultados. Os experimentos indicam que o algoritmo K-Means é o menos adequado para o agrupamento de dados ponderados, o que se verifica pela baixa qualidade dos clusterings obtidos, em relação aos demais algoritmos. O DBSCAN também apresentou resultados inferiores no agrupamento de dados ponderados, além de exigir o teste de muitas combinações de parâmetros para a obtenção de um bom clustering, o que o torna uma opção pouco promissora.

Os experimentos com os índices de avaliação interna indicam que as métricas implementadas não servem para comparar a qualidade de clusterings que não apresentam boa separação. Experimentos precisam ser realizados para avaliar se esses índices servem para indicar o número de clusters ideal de um conjunto de dados de uso de serviços.

A análise dos resultados parte do princípio de que os dados de uso simulados possuem características semelhantes aos obtidos através do monitoramento de um serviço com complexidade (como uso e composição de workflows) e volume de requisições razoáveis. Os conjuntos de dados de uso reais podem, no entanto, não confirmar estas premissas, ou então apresentar uma complexidade muito superior às condições simuladas. Desta forma, os experimentos fornecem bons indicativos quanto aos melhores parâmetros a serem utilizados no framework proposto, porém esses resultados precisam ser confirmados através de experimentos extensivos, utilizando dados de uso reais.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi proposto um framework que auxilia na execução de um processo de KDD para a descoberta de perfis de uso de serviços. O framework extrai dados de uso contidos em logs de mensagens, integra e armazena os dados pré-processados em uma base de dados de uso, agrupa clientes de acordo com o uso de serviços e constrói perfis de uso a partir dos clusters gerados. O framework abstrai parte da complexidade do processo de KDD através de tarefas pré-definidas e parametrizáveis, que auxiliam em todas as etapas do processo: seleção, preparação, agrupamento, validação e geração de perfis.

Um protótipo do framework foi implementado e validado através de experimentos com dados de uso simulados, os quais ilustram sua capacidade de descobrir perfis de uso de serviços úteis. A avaliação dos resultados foi feita sobre os agrupamentos gerados durante a etapa de clustering, dado que eles determinam a qualidade dos perfis de uso resultantes. Nos experimentos avaliou-se a qualidade dos clusterings gerados por diferentes algoritmos, sob diversas parametrizações possíveis, utilizando dados preparados de forma binária ou ponderada e com diferentes níveis de ruído. Também foram feitos experimentos comparando métricas de avaliação de clustering não supervisionadas, as quais podem ser necessárias para validar resultados em situações de uso real do framework.

Os resultados do clustering foram promissores, mesmo na presença de um nível de ruído de 30%, provendo indícios sobre qual o melhor algoritmo e parametrização para o agrupamento de dados de uso de serviços. Os resultados da avaliação através de métricas não supervisionadas não foram os esperados, indicando que os índices internos de avaliação escolhidos (Silhueta e SD) não são adequados para comparar a qualidade de agrupamentos de diferentes tipos.

Os resultados obtidos são válidos assumindo que os dados simulados possuem características comuns a dados reais de uso de serviços. Apesar de os dados terem sido simulados de acordo com a especificação de um serviço real, e considerando o uso e composição de workflows de operações, é possível que dados reais de uso não apresentem estas características ou possuam uma complexidade maior. Desta forma, é necessário realizar experimentos mais extensivos, utilizando também dados reais de uso, para que os resultados obtidos tenham um maior respaldo.

É importante ressaltar que quase a totalidade dos trabalhos analisados em mineração do uso de serviços também realizam experimentos sobre dados sintéticos. Um dos maiores desafios envolvidos na mineração de serviços é a disponibilidade de dados, devido a sua natureza proprietária (NAYAK, 2008). Neste trabalho em particular, a obtenção de dados reais é ainda mais difícil, pois o serviço analisado precisa ser moderadamente complexo e possuir no mínimo algumas centenas de clientes.

O trabalho realizado resultou no protótipo construído e nos experimentos realizados,

os quais podem guiar o aprimoramento do framework e outros trabalhos futuros. Os resultados do trabalho também foram publicados em conferências de I-Restrito e periódico:

1. YAMASHITA, M. et al. Measuring change impact based on usage profiles. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 19, ICWS. Proceedings. . .* . Honolulu: IEEE, 2012. p. 226–233.

2. SILVA, E. et al. A business intelligence approach to support decision making in service evolution management. In: *INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 9, SCC. Proceedings. . .* . Honolulu: IEEE, 2012. p. 41 –48.

3. VOLLINO, B.; BECKER, K. A Framework for Web Service Usage Profiles Discovery. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 20, ICWS. Proceedings. . .* . Santa Clara Marriot: IEEE, 2013. p. 115–122.

4. VOLLINO, B.; BECKER, K. Usage Profiles: A Process for Discovering Usage Patterns over Web Services and its Application to Service Evolution. *International Journal of Web Services Research (IJWSR)*, IGI Global, v. 10, n. 1, p. 1–28, Setembro 2013.

5. *Best Student Paper Award* pelo trabalho "A framework for Web Service Usage Profiles Discovery", *2013 20th IEEE International Conference on Web Services (ICWS)*.

Em relação aos trabalhos relacionados, discutidos no Capítulo 3, o trabalho aqui proposto diferencia-se por: a) analisar padrões de uso em granularidade mais fina; b) fornecer um modelo descritivo dos dados, na forma de grupos de clientes baseados no uso; e c) pelo tipo de aplicação alvo. Em relação a estes aspectos:

- Aplicação: enquanto os demais trabalhos tratam a recomendação de serviços, análise de composição de serviços e descoberta de processos de negócio, o presente trabalho é voltado ao apoio à gestão da evolução de serviços, uma área nova. Diversas aplicações na área de análise de impacto da evolução já foram propostas no contexto do framework WS-Evolv, entre elas: a) quantificação do impacto de mudanças incompatíveis (YAMASHITA et al., 2012), b) análise da compatibilidade orientada ao uso (VOLLINO; BECKER, 2013b), e c) avaliação do impacto de mudanças em portfólios de serviços, integrando as perspectivas financeira e de uso (SILVA et al., 2012; SILVA; BECKER; GALANTE, 2013).

- Granularidade: na área de recomendação, muitas abordagens utilizam a similaridade entre usuários com base na invocação de serviços, e na descoberta de processos de negócio, são analisados os fluxos frequentes de execução de operações de um serviço. Já o framework proposto executa o agrupamento clientes de um serviço, com base no uso de serviços, operações e tipos.

- Modelo: os perfis de uso descrevem grupos de clientes e permitem quantificar o uso das features do serviço, enquanto os demais trabalhos não permitem fazer esta relação. Na recomendação de serviços são utilizados modelos de agrupamento ou de similaridade entre clientes, com base em invocações de serviços; na análise de composição, são descobertos modelos de composição de serviços, os quais não relacionam os clientes aos padrões descobertos; e na descoberta de processos de

negócio, os modelos descobertos descrevem fluxos relacionados ao uso de todos os clientes de um serviço, e não ao uso de grupos específicos.

Os provedores podem utilizar a informação de perfis de uso não só para avaliar o impacto de mudanças, decidir sobre a criação, manutenção e desativação de versões de serviço, mas também em outras aplicações, como recomendação de serviços, otimização, balanceamento de carga, redesign, etc.

Como trabalhos futuros, projetamos a implementação e experimentação de mais índices internos de validação de clusters e experimentos com dados reais, para embasar a recomendação de algoritmos e parâmetros. É necessário verificar se índices como o S_dbw (HALKIDI; VAZIRGIANNIS, 2001) são capazes de lidar com dados ruidosos e clusters não bem separáveis (LIU et al., 2010).

Outros trabalhos futuros envolvem o estudo de estratégias para reduzir a intervenção do usuário durante a parametrização do framework, especialmente no que diz respeito à escolha do melhor clustering e do número ideal de clusters, a avaliação dos custos envolvidos na coleta de dados de uso detalhados, e o desenvolvimento de experimentos com dados de uso de mais de um serviço e de versões de serviços.

REFERÊNCIAS

AALST, W. van der. Service mining: Using process mining to discover, check, and improve service behavior. *Services Computing, IEEE Transactions on*, IEEE, PP, n. 99, p. 1, 2012.

ALVES, L. J. K.; BECKER, K. Implementação de um repositório de Versões de serviços Web usando OrientDB. In: *ESCOLA REGIONAL DE BANCO DE DADOS, 9., ERBD, Anais. . .* . Camboriú: Sociedade Brasileira de Computação, 2013.

ANDRIKOPOULOS, V.; BENBERNOU, S.; PAPAZOGLOU, M. On the evolution of services. *Software Engineering, IEEE Transactions on*, IEEE, n. 99, p. 1–1, 2011.

BECKER, K. et al. Automatically determining compatibility of evolving services. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 15, ICWS. Proceedings. . .* . Viena: IEEE, 2008. p. 161–168.

BOOTH, D. et al. *Web Services Architecture*. [S.l.], 2004. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>. Acesso em: 29 nov. 2013.

BRAY, T. et al. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. [S.l.], 2008. Disponível em: <http://www.w3.org/TR/2008/REC-xml-20081126>. Acesso em: 29 nov. 2013.

CHINNICI, R. et al. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. [S.l.], 2007. Disponível em: <http://www.w3.org/TR/2007/REC-wsdl20-20070626>. Acesso em: 29 nov. 2013.

CHUVAKIN, A.; PETERSON, G. Logging in the age of web services. *Security & Privacy, IEEE*, IEEE, v. 7, n. 3, p. 82–85, 2009.

CRUZ, S. da et al. Monitoring e-business Web services usage through a log based architecture. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 11, ICWS. Proceedings. . .* . San Diego: IEEE, 2004. p. 61–69.

DALTON, L.; BALLARIN, V.; BRUN, M. Clustering algorithms: on learning, validation, performance, and applications to genomics. *Current genomics*, Bentham Science Publishers, v. 10, n. 6, p. 430, set. 2009.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Wiley, v. 39, n. 1, p. 1–38, 1977.

DUNN, J. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, Taylor & Francis, v. 4, n. 1, p. 95–104, 1974.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2., KDD. Proceedings. . .* . Oregon: AAAI Press, 1996. p. 226–231.

FANG, R. et al. A version-aware approach for web service directory. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 13, ICWS. Proceedings. . .* . Salt Lake City: IEEE, 2007. p. 406–413.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, ACM, v. 39, n. 11, p. 27–34, nov. 1996.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, 2000.

FOKAEFS, M. et al. An empirical study on web service evolution. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 18, ICWS. Proceedings. . .* . Washington DC: IEEE, 2011. p. 49–56.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. Cluster validity methods: part I. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 31, n. 2, p. 40–45, jun. 2002.

HALKIDI, M.; VAZIRGIANNIS, M. Clustering validity assessment: finding the optimal partitioning of a data set. In: *INTERNATIONAL CONFERENCE ON DATA MINING, ICDM. Proceedings. . .* . San Jose: IEEE, 2001. p. 187–194.

HALKIDI, M.; VAZIRGIANNIS, M.; BATISTAKIS, Y. Quality scheme assessment in the clustering process. In: *EUROPEAN CONFERENCE ON PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, 4, PKDD. Proceedings. . .* . Londres: Springer-Verlag, 2000. p. 265–276.

HALL, M.; FRANK, E.; HOLMES, G. The WEKA data mining software: an update. *ACM SIGKDD*, ACM, v. 11, n. 1, p. 10–18, 2009.

HAN, J.; KAMBER, M. *Data Mining: Concepts and Techniques*. 2nd. ed. San Francisco, CA: Morgan Kaufmann, 2006.

KANG, G. et al. AWSR: Active Web Service Recommendation Based on Usage History. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 19, ICWS. Proceedings. . .* . Honolulu: IEEE, 2012. p. 186–193.

LEE, K.; KANG, K. C.; LEE, J. Concepts and guidelines of feature modeling for product line software engineering. In: *INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, 7, ICSR. Proceedings. . .* . Londres: Springer-Verlag, 2002. p. 62–77.

LEITNER, P. et al. End-to-end versioning support for web services. In: *INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 5, SCC. Proceedings. . .* . Honolulu: IEEE, 2008. v. 1, p. 59–66.

LIANG, Q.; CHUNG, J. Y. Analyzing Service Usage Patterns: Methodology and Simulation. In: *INTERNATIONAL CONFERENCE ON E-BUSINESS ENGINEERING, 2007, ICEBE. Proceedings. . .* . [S.l.]: IEEE, 2007. p. 359–362.

LIANG, Q. A. et al. Service pattern discovery of web service mining in web service registry-repository. In: *INTERNATIONAL CONFERENCE ON E-BUSINESS ENGINEERING, 2006, ICEBE. Proceedings. . .* . [S.l.]: IEEE, 2006. p. 286–293.

LIU, Y. et al. Understanding of internal clustering validation measures. In: *INTERNATIONAL CONFERENCE ON DATA MINING, 10, ICDM. Proceedings. . .* . Sydney: IEEE, 2010. p. 911–916.

LO, W. et al. Collaborative Web Service QoS Prediction with Location-Based Regularization. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 19, ICWS. Proceedings. . .* . Honolulu: IEEE, 2012. p. 464–471.

MACQUEEN, J. B. Some methods for classification and analysis of multivariate observations. In: CAM, L. M. L.; NEYMAN, J. (Ed.). *BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY, 5. Proceedings. . .* . [S.l.]: University of California Press, 1967. v. 1, p. 281–297.

MOTAHARI-NEZHAD, H. et al. Event correlation for process discovery from web service interaction logs. *The VLDB Journal*, Springer-Verlag New York, Inc., v. 20, n. 3, p. 417–444, set. 2011.

NAYAK, R. Data mining in web services discovery and monitoring. *International Journal of Web Services Research*, Idea Group Publishing/Information Science Publishing, v. 5, n. 1, p. 62–80, 2008.

PAPAZOGLOU, M. P.; ANDRIKOPOULOS, V.; BENBERNOU, S. Managing evolving services. *IEEE Software*, v. 28, n. 3, p. 49–55, 2011.

PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. Restful web services vs. big'web services: making the right architectural decision. In: *INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 17. Proceedings. . .* . [S.l.]: ACM, 2008. p. 805–814.

PFITZNER, D.; LEIBBRANDT, R.; POWERS, D. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, v. 19, n. 3, p. 361–394, jul. 2009.

RONG, W.; LIU, K.; LIANG, L. Personalized Web Service Ranking via User Group Combining Association Rule. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 16, ICWS. Proceedings. . .* . Los Angeles: IEEE, 2009. p. 445–452.

ROUSSEEUW, P. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, v. 20, n. 1, p. 53–65, nov. 1987.

SILVA, E.; BECKER, K.; GALANTE, R. Supporting strategic decision making on service evolution context using business intelligence. In: *INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 10, SCC. Proceedings. . .* . Santa Clara Marriott: IEEE, 2013. p. 8p.

SILVA, E. et al. A business intelligence approach to support decision making in service evolution management. In: *INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 9, SCC. Proceedings. . .* . Honolulu: IEEE, 2012. p. 41 –48.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to data mining*. [S.l.]: Addison Wesley, 2006.

TANG, R.; ZOU, Y. An approach for mining web service composition patterns from execution logs. In: *INTERNATIONAL SYMPOSIUM ON WEB SYSTEMS EVOLUTION, 12, WSE. Proceedings. . .* . [S.l.]: IEEE, 2010. p. 53–62.

VOLLINO, B.; BECKER, K. A Framework for Web Service Usage Profiles Discovery. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 20, ICWS. Proceedings. . .* . Santa Clara Marriot: IEEE, 2013. p. 115–122.

VOLLINO, B.; BECKER, K. Usage Profiles: A Process for Discovering Usage Patterns over Web Services and its Application to Service Evolution. *International Journal of Web Services Research (IJWSR)*, IGI Global, v. 10, n. 1, p. 1–28, Setembro 2013.

W3C. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. 2007. Disponível em: <http://www.w3.org/TR/soap12-part1/>. Acesso em: 29 nov. 2013.

WANG, S.; CAPRETZ, L. Dependency and entropy based impact analysis for service-oriented system evolution. In: *INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY, 2011, WI-IAT. Proceedings. . .* . [S.l.: s.n.], 2011. v. 1, p. 412–417.

YAMASHITA, M.; BECKER, K.; GALANTE, R. Service evolution management based on usage profile. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 18, ICWS. Proceedings. . .* . Washington DC: [s.n.], 2011. p. 746–747.

YAMASHITA, M.; BECKER, K.; GALANTE, R. A feature-based versioning approach for assessing service compatibility. *Journal of Information and Data Management (JIDM)*, v. 3, n. 2, p. 120–131, 2012.

YAMASHITA, M. et al. Measuring change impact based on usage profiles. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 19, ICWS. Proceedings. . .* . Honolulu: IEEE, 2012. p. 226–233.

YAMASHITA, M. C. *Service Versioning and Compatibility at Feature Level*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, 2012.

YU, Q. Decision Tree Learning from Incomplete QoS to Bootstrap Service Recommendation. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 19, ICWS. Proceedings. . .* . Honolulu: IEEE, 2012. p. 194–201.

ZHANG, M. et al. A Web Service Recommendation Approach Based on QoS Prediction Using Fuzzy Clustering. In: *INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 9, SCC. Proceedings. . .* . Honolulu: IEEE, 2012. p. 138–145.

ZHANG, Q.; DING, C.; CHI, C.-H. Collaborative filtering based service ranking using invocation histories. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 18, ICWS. Proceedings. . .* . Washington DC: IEEE, 2011. p. 195–202.

ZHANG, X. et al. Web service community discovery based on spectrum clustering. In: *INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND SECURITY, 8, ICWS. Proceedings. . .* . Los Alamitos, CA, USA: IEEE, 2009. p. 187–191.

ZOU, Z. L. et al. On synchronizing with web service evolution. In: *INTERNATIONAL CONFERENCE ON WEB SERVICES, 15, ICWS. Proceedings. . .* . Viena: IEEE, 2008. p. 329–336.