Università degli studi di Milano - Bicocca
Doctorate School of Sciences
Ph.D Program in Computer Science
XXV Cycle

Service Matchmaking: Exploiting the Web

Ph.D. Thesis

Author:
Luca Panziera

Supervisor:
Prof. Flavio De Paoli
Tutor:
Prof. Francesco Tisato

April 2013

"Without music to decorate it, time is just a bunch of boring production deadlines or dates by which bills must be paid."

Frank Zappa

UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Abstract

Doctorate School of Sciences
Ph.D Program in Computer Science

Doctor of Philosophy

Service Matchmaking: Exploiting the Web

by Luca Panziera

Services are software components with standard interfaces that enable rapid and flexible development of distributed systems over the Web. The literature proved that service matchmaking is the most effective approach for finding services that fulfil user needs. Available matchmaking tools require a repository composed of semi-structured descriptions that specify service properties according to a shared model.

The diffusion of services that provide public functionalities on the Web, called Web APIs, is leading to a new scenario in which increasing Web information about services is available on dispersed sources, such as official documentations, wikis, forums, blogs and social networks.

In order to exploit Web information to support service matchmaking, several issues must be addressed. Web sources provides service information according to heterogeneous data formats, models and vocabularies. Web information is dynamic, then can change over time. Available descriptions can be invalid in terms of accuracy, currency and trustworthiness. Sources provide partial or contradictory information about same services. Finally, some relevant properties, such as service provider popularity or quality of service documentation, are not explicitly reported on the Web, then may be wrongly interpreted by users through personal subjective evaluations.

This thesis provides an overall approach for enabling effective and efficient service matchmaking on Web information by addressing the issues above. The approach constructs semantic descriptions by extracting (i) explicit property values from heterogeneous non-semantic sources; (ii) subjective property values through social media. Then, quality assessments on Web information are exploited for fusing valid descriptions extracted by several sources and performing effective matchmaking. The overall approach is implemented through a lightweight distributed architecture which focuses on scalability issues by managing big amount of Web information about services.

Acknowledgements

I would like to thank many people that supported my PhD work and taught me, during the last three years, how to be a researcher and a better person. First of all, I would like to thank my supervisor, Prof. Flavio De Paoli, for his precious advice, ideas and taught me the huge difference between software developers and doctors of philosophy. I would also like to thank my tutor, Prof. Francesco Tisato, for its pearls of wisdom and the support during the first year of PhD school.

Special thanks go to the Prof. Carlo Batini, Marco Comerio and, last but not least, Matteo Palmonari, for the big contributions to my research work. I would also like to thank my research group collogues, Gigi Viscusi and Simone Grega, for their suggestions about my PhD career.

Thanks to Prof. John Domingue, Carlos Pedrinaci and Matthew Rowe, for the supervision and their advices during my research internship at Knowledge Media Institute (KMi). Moreover, I would like to thank all the *KMiers* and my housemates for the wonderful period spent in Milton Keynes.

Thanks to my PhD colleagues which I shared the best and the worst moments of our doctorate school: Enrico, Stefano, Axel, Lorenza, Luca M., Carlo, Mauro and Anisa.

I would also like to thank my parents, Luisa and Francesco, and my sister, Marzia, for the support and understanding throughout all the years I spent at the University of Milan-Bicocca.

Thanks to all my best friends, despite the few time spent with them during last years: Brasca, Nico, Pippo, Ale and Fra Bossi, Luchino, Anna and Lore, Claudio, Skizzo, Luca S., Martina, Oriana and Alan, Marzio, Filippo, Nano, Alice Bianconiglio, Scarch, Diana (Key), Riccardo, Ventu, Gloria e Luna.

Finally, my warmest thank goes the person which most morally supported me and with whom I hope to share the best moments of my future life in UK. Thank you, Alice.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, the Internet and the World Wide Web are the *de facto* standards for the development of distributed systems over computer networks. Organizations and companies started to deploy software components accessible through the Web in order to provide functionalities to general users and business partners. The availability of these components allows for the construction of distributed applications for business as well as personal use. To give an example, banks provide software components in order to support credit card payments to e-commerce websites.

Despite the Web is based on standard communication protocols, such as HTTP, FTP and SMTP, developing distributed systems suffers from interoperability issues when integrating software components deployed by different providers, because communication messages can be defined through heterogeneous data formats, such as XML, JSON or CVS, and implemented through heterogeneous protocols. Resolving incompatibility of formats and protocols can require a huge design and development effort since providers do not adopt a common and shared approach for publishing functionalities. Moreover, these issues limit the flexibility of distributed systems, especially in a business context in which rapid and sudden change of business partners, then components, should be possible. Therefore, the time required to redesign and modify systems can become a very critical issue. Finally, distributed systems that integrate pre-existing components cannot be easily supervised and monitored in order to detect and prevent system failures.

In the early 2000, *Service-Oriented Computing* (SOC) was proposed to address these issues. SOC is a computing paradigm that defines *services* as platform-independent

software components with standard interfaces to support a rapid and flexible development of distributed applications through the Web [1]. This paradigm is based on the *Service-Oriented Architecture* (SOA) whose main characteristic is the definition of three roles: *service provider*, *service registry* and *service requester* (in figure 1.1). The service provider offers software as a service (e.g., a bank that provides functionalities for on-line credit card payments). A service registry manages service descriptions that are published by service providers and made available to service requesters. Finally, a requester performs a registry consultation, called *service discovery* in the literature, then invokes the service that better fulfils his needs.

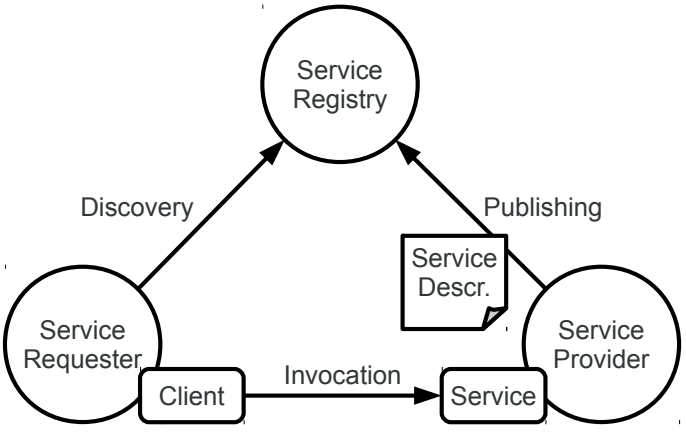

FIGURE 1.1: The Service-Oriented Architecture (SOA)

In order to implement the SOA in practice, *Web services* was proposed by *the World Wide Web Consortium* (W3C), the international organization that defines standards for the Web. According to the W3C definition [2], a Web service is a software component designed to support interoperable machine-to-machine interaction over a network. Its interface is described through the *Web Service Description Language* (WSDL), a machine-processable format based on XML and the protocol to interact with the software is the *Simple Object Access Protocol* (SOAP). Then, the *Organization for the Advancement of Structured Information Standards* (OASIS), a global consortium that drives the development, convergence, and adoption of Web service standards defined the *Universal Description, Discovery and Integration* (UDDI) and the *Web Services Business Process Execution Language* (WS-BPEL). The former is a standard that defines a set of operations that registries should implement to publish and retrieve service descriptions. Instead,

the latter is a language for describing processes composed of Web services. WS-BPEL allows for complex orchestrations between SOAP-based services.

A different paradigm for implementing services was also proposed, called *Representational State Transfer* (REST) [3], but only recently it reached a wide popularity. REST is a software design style in which functionalities are provided through *resources*. Resources represent conceptual entities, such as bank accounts, products or users. Each resource has an associated representation that must be univocally identified and contains all the information necessary to understand and modify a resource. A *RESTful service* implementation typically relies on URIs and HTTP to identify access and manipulate resources. By using directly HTTP, REST is more lightweight, simple and compliant to the Web nature compared to SOAP that adds complexity and overhead to application protocols of OSI model [4, 5].

More recently, a new wave of services, called *Web APIs*, is growing on the Web [6]. Web APIs are services that provide public functionalities on the Web, such APIs provided by Google Maps, Twitter and Amazon Web services. Actually, the term Web API is often used to comprehend services that are partially or completely compliant with REST, but also SOAP, JavaScript functions or other HTTP-based protocols, such as XML-RPC. Currently, thousands of Web APIs are made available by several providers, including Google, Facebook and Amazon [6].

1.1 Motivation

One of main research issues of SOC is the automation of service discovery [1] to support requesters for choosing among the huge number of available services. Two main approaches have been proposed for addressing this issue. A first solution historically introduced was to consider service discovery as an information retrieval issue [7]. The information retrieval approach consists of extracting significant terms from textual service descriptions. The documents that contain a high frequency of required terms are returned to users. Despite the effort spent in the information retrieval research area, these techniques are often ineffective because of potential false positive results (retrieved documents but not relevant) and false negative results (relevant documents but not retrieved). This is mainly due to the descriptions characteristics, which are often very

short (few lines or words) and generic. Therefore, requesters needs to manually check the retrieved descriptions.

A more effective solution is to perform *service matchmaking*, which means to compare automatically service descriptions in order to select the service that better fulfils constraints defined by requesters. Service matchmaking requires semi-structured descriptions in which service properties are well defined according to a model, such as WSDL. The matchmaking can be performed on three kinds of properties: *functional*, *non-functional* and *behavioural* properties. Functional properties (FPs) describe the functionality of a service, such operations or input and output parameters. Non-functional properties (NFPs) represent the description of characteristics that are not directly related to the functionality provided by a service, such as response time, data licensing, user rating. Finally, behavioural properties describe how the functionality of a service can be achieved in terms of interactions with the service. To give an example, a credit card payment service requires the identification of the user before accepting payments. This operation ordering can be considered a behavioural property.

During the last years, a lot of research effort has been spent to develop approaches, techniques and tools for automatic matchmaking. Most of them cover only functional properties and non-functional characteristics only marginally. These approaches can be classified in two categories: keyword-based and semantic matchmaking. The keyword-based approaches identify service properties that match with property constraints through a string comparison between values [8]. These techniques have several limitations. Requesters and providers must share a vocabulary which defines names and possible values that properties assume, in order to perform an effective matchmaking. Moreover, the definition of a unique vocabulary that describes properties for each service domain, such as music, geolocalization or business, is unfeasible.

To overcome the limits of keyword-based approaches, the semantic matchmaking has been proposed [9]. This approach aims to enrich services with descriptions compliant to the *Semantic Web*. The Semantic Web is a collaborative movement led by the *World Wide Web Consortium* (W3C) that proposes to enrich Web content with standard descriptions that defines its semantics. These descriptions, called *ontologies*, formally represent knowledge bases as graphs of concepts that are interconnected with semantic relations. Ontologies enable to address linguistic issues that purely syntactic descriptions can not

manage, such as synonymy, homonymy a multi-language support. To perform effective matchmaking, tools that infer semantic relations between ontologies, called *reasoners*, are required. These tools are able to automatic infer missing relations between concepts on the base of other defined relations (e.g., if $a = b$ and $b = c$, reasoners can infer that $a = c$). In this way, Semantic Web technologies enable a more effective service matchmaking compared to keyword-based approaches because requesters and providers can use their own vocabulary to define property names and values. Therefore, through the definition of few equivalence relations between concepts defined in different vocabularies and automatic reasoning, matching different terms with the same semantic is possible. Reasoners have a huge response time for big or complex descriptions that make semantic service matchmaking tools inadequate to be used as an application on the Web [10]. Moreover, despite the effectiveness of discovery tools based on semantic matchmaking, only few semantic representations of real services are available on the Web.

All matchmaking approaches provided in the literature requires a repository that contains service descriptions according to a semantic or non-semantic model. The diffusion of Web APIs opens a new scenario for service matchmaking. The increasing availability of Web APIs fosters the diffusion of documentation published by providers, Web APIs portals, wikis, forums and social networks that made available service information. The result is that the Web has become a source of disperse information about services that can be exploited for building semi-structured descriptions required to support automatic matchmaking.

Addressing the matchmaking by exploiting Web information lets emerge additional aspects and research issues that are not considered in the literature. Service information is available in heterogeneous format and models (mainly textual or semi-structured data). Moreover, Web information can be invalid in term of accuracy, currency or trustworthiness. Another aspect that must be considered is that the Web descriptions are dynamic, so the service information can change along the time. In addition, same services can be described by disperse sources that provide information that can be complementary or contradictory. Finally, the management of big amount of descriptions can introduce scalability issues due to network traffic and response time. Moreover, available matchmakers do not consider new properties that have been introduced by the Web API scenario. A survey, that has been performed by the author, shows that after an identification of Web APIs with similar functionalities, users select the service by

evaluating several kinds of NFPs which are marginally considered in the SOC research state-of-the-art. These properties, such as provider popularity or quality of the API documentation, are very important for users but are evaluated in a subjective way, with the result of potential wrong evaluations because are not explicitly provided on the Web.

1.2 Main contributions

The aim of this thesis is to propose novel techniques and tools for enabling and performing semi-automatic matchmaking that: (i) is effective and efficient by managing real data available on the Web, and (ii) supports service properties that are really considered by users. The approach is based on a process composes of the following steps:

- analysis of Web sources that provide service information to figure out the issues to be addressed for exploiting this data;
- analysis of Web API users needs in order to figure out which properties are really considered for the service discovery;
- modelling of an overall approach for enabling service matchmaking after an analysis of the advantages and limitation of the state-of-the-art approaches;
- development of techniques to implement the approach;
- design of a system architecture that implements the techniques and is adaptable to the distributed nature of the Web;
- implementation of the matchmaking system and analysis of effectiveness and efficiency.

More in detail, the following techniques have been developed:

- generation of service descriptions;
- property value extraction;
- quality assessments;
- quality-driven descriptions fusion;

- quality-driven service matchmaking;

Service descriptions are necessary to represent service properties according to a common model in order to address Web information integration. The *lightweight policy-centered meta-model* (PCM-lite) is proposed for describing services. PCM-lite main features are: (i) language independence, (ii) high expressiveness, (iii) a low complexity for efficient evaluations and (iv) compatibility with existing semantic models. PCM-lite is defined by an abstract model that allows users to define properties in any semantic or non-semantic data formats. Moreover, PCM-lite allows requesters and providers to express rich property descriptions through the specification of units of measurements and operators that assert numeric intervals and set of values. Its low complexity improves the efficiency of evaluations through semantic reasoning. Finally, PCM-lite descriptions can be easily mapped to other semantic descriptions in order to enable matchmaking through heterogeneous semantic models.

The *property value extraction* is addressed for extracting values of functional and non-functional properties from Web descriptions that have heterogeneous models and data formats. The approach adopted is building semantic descriptions according PCM-lite form Web information about service. As proven in the literature, Semantic Web technologies are able, on one hand, to address the heterogeneity of user vocabularies and, on the other hand, to enable an effective matchmaking. Techniques for semi-automatic extraction of values from semi-structured and textual information are provided.

Also techniques for value extraction of properties that are evaluated by users in subjective way are proposed. To address this issue, social media (e.g., social networks, forums) are exploited in order to figure out an objective value of the subjective properties based on the evaluation of overall community opinions or behaviours. Each technique is implemented by wrappers, services that are specialized to extract information from a specific source, then publish it through PCM-lite descriptions compliant to the Semantic Web. Wrappers are designed as flexible and customizable tools that are able to manage the dynamics of the Web source.

Quality assessment techniques are developed for estimating the validity of the property values extracted from Web sources. In this thesis, techniques for the evaluation of accuracy, currency and trustworthiness of the information are provided. The accuracy evaluation is based on the effectiveness of the techniques for the extraction of semantic

concepts that represent property values. Instead, the currency of the information is measured on the base of the temporal meta-data that can be provide by the sources. Finally, the trustworthiness is measured by monitoring the activity of users that follow Web sources profiles on social networks. The idea behind this approach is that more a source provides trustworthy information, more it is followed on social networks.

In order to manage the dispersion of service descriptions over the Web, the *description fusion* is addressed for fusing information extracted from each source in order to produce a unique description for each service. The novelty of the approach is to fuse descriptions by exploiting quality assessments in order to correctly choose between contradictory values. Some fusion techniques are proposed, such as aggregation, composition and selection of values.

In the Web API scenario, the techniques for *service matchmaking* available in the literature have several limitations. To support Web API matchmaking, an approach that combines automatic reasoning, for matching symbolic values, and mathematical functions, for the evaluation of numeric values have been proposed in this thesis. The novelty of the approach is driving the matchmaking by quality assessments. Assessments are exploited in order to give priority to valid descriptions in matchmaking results.

For making feasible service matchmaking by exploiting a big amount of dispersed Web information, design of an architecture that enable the scalability of the overall approach must be addressed. The *Policy Matchmaker and Ranker for Web* (PoliMaR-Web) is proposed for addressing this issue. PoliMaR-Web is based on a *lightweight service-based architecture* that is compliant to the Web characteristics. The architecture has two main aims: (i) to promote the adoption of wrappers in order increase the diffusion of semantic service descriptions and (ii) to be adopted by companies as an architectural model for matchmaking over the Web, in order to become a *de facto* standard.

The main part of this thesis contributions are the result of original works available trough author's publications [11–16].

1.3 Thesis outline

The thesis is composed as follows. The analysis of Web API characteristics, Web sources that describe services and properties that users consider for the matchmaking are in chapter 2. After a state of the art analysis of service matchmaking, chapter 3 shows the overall approach that enable service matchmaking by exploiting Web information, highlighting innovative aspects and research issues to be addressed. The solution proposed for the extraction of explicit property values from heterogeneous Web sources is described in chapter 5. Instead, techniques for the extraction of subjective property values from social media are shown in chapter 6. In chapter 7, methods for description fusion and matchmaking, both driven by quality assessment of the Web sources, are provided. The lightweight architecture of the tool that implements the extraction, fusion and matchmaking processes is shown in chapter 8. The evaluations of effectiveness and efficiency of the tool are in chapter 9. Finally, overall conclusions of the thesis are in chapter 10.

Chapter 2

Web APIs: a New Scenario for Service Matchmaking

Since the early 2000s, the vision of the service-oriented computing (SOC) promised the diffusion of services that provides functionalities available through the Web. According to this vision, the Web service technology was originally designed to promote the diffusion of services. Despite the effort spent for modelling Web service standards, this technology is mainly adopted to share private functionalities between business partners and only few of these services are publicly available to generic Web users.

Currently, a new generation of services, called Web APIs is becoming a *de facto* standard for providing public functionality on the Web. Despite the effort spent for implementing Web service standards, most of Web APIs adopt the REST architectural style and their diffusion is continuously increased on the Web during last years. The increasing availability of Web APIs has fostered the publication of a huge amount of Web documents and information describing and discussing these services. Such huge amount of data forms a potential knowledge base that can be exploited to support service matchmaking on real information.

This chapter focuses on analyses of the new scenario that Web APIs enabled for service matchmaking. The result of the analyses will provide a set of issues that must be addressed for making feasible matchmaking on the Web. The analyses provided by this chapter aims to highlight:

- technological characteristics of real services that provide public functionalities;
- characteristics of Web sources and information that describe services in order to highlight issues to be addressed for performing matchmaking on real data;
- current practices of actual users for service discovery in order to identify limits and challenges;
- service characteristics and properties considered by users when performing discovery.

The chapter is structured as follows. In section 2.1, technological characteristic of Web APIs will be analysed. Then, characteristics of the Web sources that provides service information are provided in section 2.2. Subsequently, an analysis of Web API discovery approaches adopted by users and which service properties they consider is addressed in section 2.3. Finally, section 2.4 shows the issues to be addressed to perform service matchmaking by exploiting Web information, that are resulting from analyses.

2.1 Web APIs: characteristics of Web-oriented services

Services and Web APIs are the key concepts of this thesis. A shared definition of both are not provided by the computer science and engineering literature. The "service" term assume different meaning in telecommunication, operating system, and software architecture contexts. According to SOC research area, the author defines a service as follows.

Definition 2.1 (Service)**.** *A service is a software component that provide functionalities according to a standard approach or protocol by exploiting internet.*

Instead, some works in literature consider Web APIs as function libraries accessible through HTTP or a synonymous of RESTful services. The author defines Web APIs as follows.

Definition 2.2 (Web API)**.** *A Web API is service that publicly provides functionalities for users of the World Wide Web.*

The proliferation of Web APIs is visible on ProgrammableWeb[1], the most popular Web portal of Web APIs, where more than 8800 service descriptions are available on March

[1]available at: `http://www.programmableweb.com/`

2013, from approximately 2000 available on February 2010 [6]. Often "Web API" is used as synonymous of RESTful service in literature, but only the 69% of APIs described in ProgrammableWeb use REST as interaction protocol. Instead, the 23% are Web services exploiting SOAP over HTTP, the 5% provides functionalities through JavaScript and the 2% exploit XML-RPC to communicate.

The technology in which the most of the research effort was spent in the 2000s are Web services based on SOAP. The Web services have been designed, according to SOC principles, to permit the development of flexible and easy-to-compose distributed application over a network [1]. To reach its aim, this technology is based on a set of standards: *Web Service Description Language* (WSDL), *Simple Object Access Protocol* (SOAP) and *Universal Description, Discovery and Integration* (UDDI) [17]. The Web Service Description Language (WSDL) is a XML-based language designed for describing service interfaces independently to the platform adopted and programming language.

WSDL descriptions mainly defines service operations, input/output parameters, interaction methods (synchronous or asynchronous) and a URI that identifies the service endpoint. The Simple Object Access Protocol (SOAP) is used to communicate with Web Services. It is based on XML and exploits the application layer protocols of the OSI model, such as HTTP, SMTP and FTP, as communication medium. However, SOAP-based Web APIs typically exploit HTTP. SOAP support two interaction methods: synchronous, via remote procedure calls (RPCs), and asynchronous, via messages. Finally, the Universal Description, Discovery and Integration (UDDI) is a standard that defines interactions and representation of service information for service registries. Registries are centralized systems in which service providers publish WSDL descriptions in order to support the discovery of services. UDDI specifies the definition of the information to provide about each service and, query and update functionalities to interact with the registry.

Moreover, to support the orchestration of distributed systems based on Web services, the *Web services business process execution language* (WS-BPEL) [18] is proposed. WS-BPEL is a XML-based standard that defines business process in which several services are composed to provide a complex functionality. These processes are represented through workflows that are executed by BPEL engines in order to orchestrate the interaction between services. In addition, WS-BPEL allows the management of potential faults during the execution of the process.

Despite their aim and the effort for developing standards, the result is that Web service are hardly Web-oriented [5]. Currently, Web service are mainly used by organizations and companies for integrating legacy systems and to interact with business partners systems. But only few services based on SOAP provide public functionalities on the Web, with the result that UDDI has become an unused standard. The reason is that Web applications are purely based on HTTP and SOAP adds complexity to a protocol that is a standard *de facto* for the Web.

Instead, RESTful services are based on the Representational State Transfer (REST) architectural style [3]. REST is based on the following practices.

Identification of resources

Functionalities are provided as resources. Each resource specifies a particular conceptual entity and is associated with a global identifier.

Manipulation of resources through representations

Requesters can hold the representation of resources provided by a service. Resource representations must contain all the information necessary for allowing requesters to modify or delete the service resources

Self-descriptive messages

Communication messages are stateless and self-descriptive, which means that each message includes enough information to describe how to process the message.

Hypermedia as the engine of application state (HATEOAS)

This principle specifies that requesters interact with RESTful resources through hypermedia provided dynamically by the service. Resources can represent a state of the service. Through actions that are dynamically identified within hypermedia, requesters are able to make state transitions.

Typically, Web APIs implement REST through HTTP protocol according to the following approach. URIs are adopted to identify resources on the Web and the HTTP methods to interact with resources. To give an example, a flight of a booking service can be considered a resource, therefore can be identified by a URI (e.g. `http://ba.com/ba0575`). Each standard HTTP method defines a specific operation that involves a resource:

- through the GET method, the description of the resource is retrieved (e.g., flight description);
- by using PUT, the description of a resource is replaced (e.g., for changing the landing time);
- POST adds a sub-resource to the considered resource (e.g. the price of the fight identified by `http://ba.com/ba0575/price`);
- finally, DELETE removes an existing resource.

Self-descriptive messages are implemented by associating messages with the *Internet media type* [19], previously known as a MIME type, that specifies the message data format. Therefore, the Internet media type indirectly specifies which parser to invoke for interpreting the message.

The last and most qualifying characteristic that distinguishes HTTP-based application and RESTful services is the *Hypermedia as the engine of application state* (HATEOAS). This principle is implemented through hyperlinks defined in resources descriptions. According to the fight example, HATEOAS can be implemented for booking a flight as follows. A user retrieves a fight representation through a GET. The representation can provide a hyperlink that allows the user to select the flight and pass to the credit card form necessary for the payment. Through the hyperlink, the user is able to change the application from "flight selection" state to "payment" state.

Compared to SOAP, REST is more Web-oriented because it directly exploits HTTP. Moreover, REST is not tied by XML, therefore it possible to adopt other data formats that are less verbose, such as JSON or CSV. In this way, the result is that RESTful services are more lightweight because concise data formats and the absence of additional protocols over HTTP reduce the service response time. However, REST presents several few limitations compared to SOAP. REST can be considered *data-oriented* because HTTP methods are used to access or modify the data associated with a resource. Instead, SOAP is *operation-oriented* because the iteration with Web services is performed calling operations with specific input and output parameters. Some functionalities can be implemented only through operations (e.g., generic mathematical functions, such as multiplication of numbers or logarithms). To implement these functions as Web service is

straightforward, instead the adoption of REST requires a huge design effort. In addition, REST support only synchronous interactions.

XML-RPC [20] is a standard proposed in 1998, therefore before REST and SOAP, and has common characteristics with both its successors. The interaction through XML-RPC services occurs via HTTP POSTs to services reachable and identified by an URI. Messages are defined in XML, requests contains the operation to be invoked and input parameters and responses contains output parameters. Therefore, XML-RPC is operation-oriented. This standard supports only synchronous iterations and standards for describing interfaces of these services do not exist. XML-RPC is considered the direct predecessor of SOAP, therefore its usage is decreasing.

Finally, JavaScript [21] is a programming language for Web applications that can be hardly considered a standard for implementing services. However, some providers make available scripts accessible via HTTP that contain several operations as JavaScript functions. These scripts can be embedded in a Web application, then programmers can exploit the provided functions that interact with a service according to the *Asynchronous JavaScript and XML* (AJAX) technique. AJAX [22] is a Web development technique that exploits JavaScript to perform asynchronous (but also synchronous) call via HTTP to a server. The data format returned by the server is typically XML, but can be also used JSON. This approach can be also adopted for invoking and composing RESTful and XML-RPC services because both categories exploit HTTP for providing functionalities.

The combination of JavaScript-based, RESTful or XML-RPC through AJAX is typically used to enrich existing Web applications (e.g., put Twitter streams in a personal blog) or for developing *mashups* [23]. Mashups are Web applications that combine resources or functions available through Web APIs in order to provide a new functionality that is not available on the Web. To give an example, it is possible to crate a mashup that localize personal pictures on a map by using Flickr RESTful service and Google Maps JavaScript APIs.

Composing classical Web services with HTTP-based services has huge interoperability issues because SOAP is an additional protocol over the application layer of the ISO model, therefore can exploit other protocols, such as FTP or SMTP. Possible solutions to address this issue are three. The first one is to develop an “adapter” that converts HTTP-based service to SOAP, then defining a composition as WS-BPEL process. The

second solution, that probably is the most complex, is developing a converter from SOAP to XML-RPC or REST, subsequently exploiting AJAX to compose services as a mashup. The latter one is to adopt JOpera [24], a toolkit that contains a visual language for modelling service compositions and an engine that is able to execute business processes or mashups that combines SOAP, REST, XML-RPC and JavaScript. Finally, BPEL4REST, a language for modelling business processes exploiting RESTful services, was proposed in [25], but execution engines able to interpret do not exist on the best of the author's knowledge.

Service approach	Web Services	REST	XML-RPC	JavaScript
Percentage available	23%	69%	2%	5%
Orientation	Operation	Data	Operation	Operation
Communication protocol	SOAP	HTTP	HTTP	HTTP
Communication type	Sync/Async	Sync	Sync	Sync/Async

TABLE 2.1: Technological characteristics of Web API approaches

Several aspects must be considered for modelling an effective matchmaking approach on the base of the Web API technological characteristics, that are summarized in table 2.1. The ability to matchmake properties that specify the technological approach adopted (Web services, REST, etc.) is mandatory. Requesters can be specialised in a specific Web API technology, therefore is important to enable a discovery approach able to highlight services that requires a low integration effort based on users skills. Moreover, a property that specifies the data format managed by RESTful services is necessary to be considered for preventing interoperability issues of these Web APIs. Finally, operations and input/output parameters, for operation-oriented Web APIs, and, resources and available HTTP methods, for RESTful services, are properties that must be also considered in a matchmaking process. In this way, a matchmaker is able to identify services that provide specific operations or resources, of which their composition reaches a specific objective as mashup or business process. By considering these service properties, the matchmaking is able to support users that integrate Web APIs manually, as well as tools that perform automatic composition.

2.2 Real service information available on the Web

WSDL is the standard proposed by W3C for describing SOAP-based services. According to the information available on ProgrammableWeb, 98% of providers of Web services publish WSDL documents. Instead, the scenario is different for RESTful services. An adopted standard that describe interfaces of these services does not exist because REST well defines the semantics of the operations through HTTP methods, self-descriptive messages and HATEOAS principle allows requesters to discover resources. However, some standard have been proposed to describe these services in order to improve their discovery and enable the automatic composition.

The first one is the *Web application description language* (WADL) [26]. It defines resource URIs, the data format adopted, and the HTTP methods enabled for each resource. However, only few provides describe their own services according WADL. Also the version 2.0 of WSDL support the descriptions of RESTful services, but also this standard is few adopted. Despite the availability of standards, only the 1% of providers adopted WADL or WSDL 2.0 for describing RESTful APIs.

Currently, the method adopted by providers to publish Web API descriptions, independently of the technology adopted, is unstructured text on common Web pages based on (X)HTML. According to an analysis performed by the author, these descriptions contains all the functional aspects, such as operations, parameters and data formats, but also additional information that is not strictly related to technological characteristics. Through Web pages, providers publish extra Web API information about:

- legal usage constraints, such as licensing on usage or provided data;
- preconditions, such as user registration and authentication before service usage;
- usage fees, such as service price or payment methods;
- usage limits, such as number of daily requests;

In this way, descriptions in natural language enrich WSDL documents, for SOAP-based services, and give information about HTTP-based services of which description standards are not adopted or do not exist. All these service properties must be considered in a matchmaking process because legal constraints are relevant aspects, especially in a

business context. Precondition and usage limits are elements that must be considered for a good design and development of mashups or business processes. The automatic extraction of these properties from free text in natural language is a hard issue that must be addressed in order to perform automatic matchmaking by exploiting providers description.

On the Web, third-party sources also provide service information. Wikipedia is a relevant Web source that provides Web API information. Known as one of the biggest general purpose information source, Wikipedia is a free collaborative encyclopaedia of which the content is generated by users. The information provided by wikis can not be considered trustworthy as well as descriptions published by service providers, but usually it is more structured therefore easier to be processed by machines for extracting Web API properties. Moreover, a publication date is associated with each Wikipedia page, therefore it is possible to measure how is current the information provided. Finally, Wikipedia do not provide additional properties compared to providers descriptions in general.

ProgrammableWeb and WebMashup[2] are Web portals specialized in Web APIs and provide structured profiles of services. Each profile provides tags that summarize Web API functionalities, in order to enable a keyword based search. In addition, these descriptions provide user ratings, and a summary of the service characteristics, such as data format, licensing and usage fees. ProgrammableWeb is the most popular and provides 8493 descriptions as Atom feeds accessible via REST, instead, WebMashup publishes 1776 XHTML-based descriptions. The information published through these descriptions is sometimes inaccurate or out-of-date [6]. However, as well as for wikis, the currency of the descriptions can be estimated through a publication date that is available on each profile.

A third party service that monitors Web APIs is API-status[3]. This service provide real time informations about response time and uptime of 48 popular Web APIs. This information is accessible through a RESTful service called Watchmouse as description defined in XML or JSON. Compared to the other sources, API-status has the characteristic to provide dynamic information that can change in every moment. Therefore, the dynamism of the information must be take in account for the design of a matchmaker

[2]Available at: `http://www.webmashup.com/`
[3]Available at : `http://api-status.com/`

that exploits real Web information. ProgrammableWeb also provides this information for several services, but Web APIs for accessing such information are not available.

In the current scenario, the *Semantic Web* [27] plays an important role. In the last years, an increasing number of Semantic Web descriptions is appearing through a phenomenon called *Linking Data Cloud* [28]. The aim of Linking Data Cloud is to promote the proliferation of interconnect datasets published according to *Linked Data* principles, that are based on four best practices:

- To identify concepts as URI;
- To use HTTP to look up concepts;
- When a URI is looked up, to use Semantic Web standards to provide useful information;
- To provide links to other concepts by URIs in order to interconnect the information among datasets.

Two main standards are adopted to enable these principles: the *Resource Description Framework* (RDF) [29] and the *SPARQL Protocol and RDF Query Language* [30]. RDF is the base model proposed by W3C to produce semantic descriptions. Instead SPARQL is a query language that easily permits to extract concepts and relations form RDF-based documents.

Despite the big potentialities of Semantic Web and Linked Data, few service descriptions conforming these standards is available. The largest repository of service descriptions according to the Semantic Web is OPOSSum[4] [31]. This repository contains approximately 2851 descriptions, but only approximately 200 of them represents real services of the geographic domain. The other descriptions refers to test collection created for research proposes. Moreover, the repository is not updated since 2009. Instead, iServe [32] is a more recent platform for publishing and discovery services according Linked Data principles, but also this platform provides few amounts of real service descriptions (approximately 100). This little availability could be the combination of two factors (i) the expertise required to create descriptions or (ii) the little reward of adopting Semantic Web technologies in the service providers' opinion.

[4]Available at: `http://fusion.cs.uni-jena.de/opossum/`

With the availability of third party sources that can enrich providers descriptions, a new issue emerge: the validity of the information. ProgrammableWeb and Wikipedia descriptions are user generated content. Therefore, the information can contain potential content errors, therefore it is inaccurate. Without a periodical check and update of the documents, information can be out-of-date. Moreover, malevolent users can provide fake or untrustworthy information.

To conclude, the Web offers a plethora of Web API descriptions that can be exploited to perform service matchmaking. To use this information is necessary to deal with the following issues.

Heterogeneous formats of descriptions

Service descriptions are specified as textual documents like Web pages, semi-structured information (e.g., ProgrammableWeb), and, more recently, linked data (e.g., iServe [32]).

Quality of the provided information

Web sources can provide inaccurate, out-of-date or untrustworthy information.

Dynamic information

Web data can change frequently over time (e.g., response time, availability and user rating).

Dispersed information sources

Often, information about a single service is available from multiple sources over the Web, such as Web API portals, wikis and provider sites.

2.3 Current Web API discovery: limits and user needs

The aim of this section is: (i) to identify and analyse the limits of the current Web API discovery activity performed by users and (ii) to identify properties that are relevant for users. To achieve the goal a survey was conducted with a public on-line questionnaire[5]. The questionnaire is addressed to users with different expertise: Web developers that occasionally used Web APIs, expert mashup developers and service developers. The

[5]The survey was defined in collaboration with the Knowledge Media Institute (KMi) at the Open University (Milton Keynes, UK)

survey is started in May 2012 and is still alive. The analysis results refers to data collected until March 2013.

A first set of results report the methods that are used to perform a Web API discovery. On a first sample of 50 users, the results[6] are the following:

- the 84% of users perform the discovery manually by finding information on the Web supported by search engines (e.g., Google or Bing);
- only the 11% of exploits Web API portals (such as ProgrammableWeb);
- Stackoverflow, a Question Answering (Q&A) community of computer programming experts, is considered a relevant source for the Web API discovery by the 38% of users.

The results show that the most used way for Web API discovery is common Web search, which is time consuming, because it is based on manual Web browsing, and provides less than optimal results, because search engines are not designed for Web APIs. The lack of popularity of discovery engine can be caused by several factors. The first one can be the focus of most engines available business context and SOAP-based services, while most of Web APIs are RESTful services (see section 2.1), that refer to disparate contexts. A second reason can be related to usability problems of engine interfaces. Currently, keyword-based search with additional advanced search constraints is a *de facto* interaction standard for discovery information on the Web. Therefore, discovery engines that provide interfaces that do not follow this approach are less familiar to users. A third reason is that search engines allows users to discover any kind of information related to services that is available on the Web. Instead, discovery engines are designed for exploiting local repositories of descriptions that contains partial information. Moreover, repositories must be continuously updated in order to reflect real information. To give an example, ProgrammableWeb provides a keyword-based discovery engine that exploits its own Web API descriptions, but the information of this source some times is inaccurate or out-of-date [6]. Therefore, the result is that Web API portals, such as ProgrammableWeb, are exploited by few users, as shown by the survey.

A relevant aspect, that emerge from survey results, is that on-line communities of programming experts, such as Stackoverflow, are considered a trustworthy source for

[6]People may select more than one preference, so the sum of percentages is more than 100%

third of users. By asking to experts or browsing answers, users can discover the service that better fulfil his their requirements. This discovery activity can require a large amount of time, because users must wait for experts answers, but the result is acceptable.

A second set of results refers to the criteria that users consider to identify, in a set of services with similar functionalities (e.g., mapping and geolocalization), the best one. The survey shows that, for 67% of user sample, important characteristics are:

- provider popularity;
- vitality of Web API forums for developers support;
- quality of the answers provided in Web API forums;
- quality of the Web API documentation.

For users, popularity of the service provider is directly related to reliability, stability and with performance of Web API. Vitality of a forum measures how many users are active by posting messages. In the user prospective, the vitality of Web API forum is a metric of the community promptness for resolving developing issues. Instead, the quality of the answers available in forum is indicative of the smartness of the community that support the Web developers. Finally, a well-written documentation permits developers to prevent potential issues then decreasing risks and effort required to adopt a Web API.

A limit to perform matchmaking on these properties is that their information is not explicitly described on the Web, as opposed to operations, response time or usage fees. Therefore, users estimate them in a subjective way, on the base of personal experience, acquaintance opinions or inferring it through personal inaccurate analysis on the Web. These "subjective" properties are relevant for most users, therefore the evaluation of them is an important challenge that must be considered and marginally address in literature.

2.4 Issues for enabling service matchmaking on the Web

According to Web API characteristics, the user needs and the information available, several issues must be addressed for enabling matchmaking of real services that publish functionalities on the Web.

Evaluation of functional and non-functional properties

A matchmaker must be able to evaluate all the characteristic of technologies and approaches adopted to implement services in order to identify which one implements the functionality required by a requester. Moreover, the ability of evaluating non-functional properties is necessary to support users for choosing between services that provide similar functionalities.

Evaluation of subjective properties

The matchmaker must be able to provide an objective evaluation of property that are subjective for users, such as provider popularity or quality of the Web API documentation.

Management of information dynamism

The dynamism of the information requires an effective monitoring in order to matchmake up-to-date property values.

Property extraction from heterogeneous Web sources

The approach must be able to extract and evaluate property values defined according heterogeneous models and vocabularies.

Quality assessments on Web information

The matchmaker must be able to evaluate the validity of the information that describe a property in terms of accuracy, currency and trustworthiness.

Collection and selection of the disperse information

Collect of disperse information regarding a specific service over the Web and its selection are necessary in order to provide a unique representation of each service that is complete and valid.

Approach scalability

The management and evaluation of a big amount of disperse descriptions can require an implementation of the approach that is scalable.

User request specifications based on keywords and property constrains

Requesters must be able to define keywords to perform functional discovery and additional constraints on properties that are relevant for the matchmaking.

An effective matchmaker should consider several service properties. According to the analysis performed, service properties can be classified according four main dimensions:

functional, *value category*, *subjectivity* and *dynamism*. The *functional* dimension defines how a property is related to a service functionality. According to SOC literature [1], properties can be classified according two categories which are defined as follows.

Definition 2.3 (Functional property)**.** *Functional properties (FPs) are characteristics that define functionalities and interfaces provided by services.*

Definition 2.4 (Non-functional property)**.** *Non-functional properties (NFPs) define service characteristics which their change does not affect the service functionality.*

Functional properties include service operations, endpoints, input/output parameters, textual descriptions of functionalities and the communication protocol adopted by the Web API (e.g., SOAP, REST, XML-RPC and, AJAX). Instead, some examples of non-functional properties are usage fees, user rating, provider popularity and data licensing. NFPs include a set of service properties that in SOC literature are called *quality of service* (QoS) [33]. QoS is a set of properties that describe services performance, such as availability and response time, and security, such as message cryptography. To give an example, the response time and the availability provided by the API-status services are QoS. In a business context, these NFPs are described by *service-level agreements* (SLAs) that are contracts in which providers guarantee a level of service performances and security to users. However, QoS can be useful also in a non-business situation to support mashup developers for choosing the most reliable Web API in a set of services with similar functionalities.

Criteria to classify properties as functional or non-functional is not explicitly defined in literature, because several properties can be classified according to subjective user evaluations or the usage context. For instance, usage limits, defined as number of daily requests that a user can perform, can not be clearly considered a functional or non-functional property. On the one hand, it could be a NFP because the variation of its value do not affect the service functionality. For instance, if Twitter API decrease the number of daily request allowed, its functionality does not change. But, on the other hand, this property could be considered functional because it affects the consumption of the service, then the clients design.

The *value category* dimension refers to the kind of value that a property assumes. According to this dimension, the author proposes to classify properties as quantitative or qualitative. Quantitative and qualitative properties definitions are provided following.

Definition 2.5 (Quantitative property)**.** *A property is quantitative when it assumes numeric values and an optional unit of measurement.*

Definition 2.6 (Qualitative property)**.** *A property is qualitative when it assumes symbolic values.*

To give some examples, response time and price are quantitative properties because them assumes numeric values (e.g., 100 milliseconds or 0.99 Euros). Instead, data licensing is a qualitative properties because assumes symbolic values (e.g., Creative Commons license).

The *subjectivity* dimension defines how much is subjective the user evaluation of the properties on the base of the information available on the Web. According this second dimension, properties can be classified as explicit or subjective.

Definition 2.7 (Explicit property)**.** *A property is explicit when its values are clearly defined and interpreted without ambiguity on the Web*

Definition 2.8 (Subjective property)**.** *A property is subjective when its values are interpreted by user perception*

Usually, functional properties, such as operations and parameters are explicit in documentation published by service providers. Instead, providers popularity or quality of the answers provided in Web API forums are purely subjective because each user evaluates them according to its perception or experience. However, also properties that are explicit on the Web can be interpreted in a subjective way. To give an example, user rating is typically available as an amount of stars between 0 and 5. For some users, a good services are rated with at most 3 stars, instead, for more demanding users, 4 stars is an acceptable index.

Also the *dynamism* dimension can be considered to classify properties. The Web information is not completely static, and service properties too. Despite functional properties, such as operations and input/output parameters, are considered static in general, also these properties have a low dynamism. Web APIs are software, therefore these services can change their functionalities. For instance, new versions of operation can be released and old ones become deprecated. Otherwise, response time and provider popularity can be very dynamic. Both of them can change in every moment on the base of external events that can be unpredictable. Response time can depend on

service infrastructure workload and network traffic, instead provider popularity is related to marketing, news and user perception of who is offering services. Therefore, the management of property dynamism is necessary in order to perform matchmaking on up-to-date information.

Properties are available through heterogeneous documents on the Web. Document heterogeneity can be classified as syntactic or semantic. Syntactic heterogeneity refers to data formats adopted to describe services. The main adopted formats are HTML, XML and JSON. Instead, semantic heterogeneity refers to models and vocabularies adopted for providing service properties. Despite the availability of standard languages, such has WSDL and WADL, most of Web APIs are described according non-standard models or in natural language. In addition, few service descriptions according to Semantic Web models are available, therefore the advantages of semantic matchmaking cannot be exploited.

Service properties are published by providers and third-party sources. Compared to matchmaking approaches in literature that exploit local repositories, the quality of the information is an important aspect that must be tackled for using Web descriptions. On the Web, descriptions can be inaccurate because include inconsistent property values. Third-party sources can be composed of partially out-to-date information, such as ProgrammableWeb repository. Finally, the trustworthiness of sources is a key element that must be evaluated in order to exclude fake or slanted descriptions. Therefore, to perform matchmaking on valid data, quality assessments must be addressed.

In addition, several Web sources can provide information about same services. To give an example, API-status provides response time of Twitter APIs, which is not published by provider documentation. This information must be integrated in order to provide a unique rich document that contains all the available properties of a specific service. Sources can provide also same properties with contradictory values assigned to a service. For instance, Programmable can assert that Yahoo! maps APIs provides data free license. Instead, Wikipedia can state that the service provides copyrighted data. A technique that is able to select valid information is necessary.

The management of Web information can introduce scalability issues. The network is a bottleneck for collecting and evaluating a big number of Web descriptions. On-the-fly collection of information each time a requester requires matchmaking is unfeasible.

Therefore, a matchmaking strategy must be scalable for providing good results to users in an acceptable response time.

The adoption of common search engines for Web API discovery proves that keyword-based search is a *de facto* interface for users. However, the definition of properties as natural language keywords do not allow automatic tools to easily interpreter the requirements of users. To give an example, if the requirement are defined by the text "a mapping service with low response time", a software should be able to understand that mapping refers to the functionality and response time to a NFP. Moreover, "low response time" is subjective and depends on the domain. The following more structured requirements "Functionality: mapping; Response time: $\leq$ 100 ms." make easier the parsing of a user request. Therefore, an ideal matchmaker must be able to manage requirement definitions composed by keywords that define the functionality and an optional set of constraints regarding properties that are relevant for users.

In this chapter, the current scenario of services on the Web is described. For each analysed aspect, a set of consideration about characteristics of an effective matchmaker have been made on the base of the available information and user needs. These characteristics will be used in the next chapter for comparing service matchmakers available in literature. The aim of this comparison is to evaluate the current limitations of proposed works in order to develop an effective matchmaking approach that exploits information available on the Web.

Chapter 3

Enabling Service Matchmaking Exploiting the Web

The aim of this chapter is to provide a novel approach for enabling an effective service matchmaking by using information available on the Web. In chapter 2, an analysis of the sources that describe services and Web API user needs have been provided. The results of this deep analysis highlighted a set of issues that must be considered to reach the objectives of this thesis.

In this chapter, available service discovery tools are analysed in order to figure out how these issues introduced by the Web API scenario have been addressed. The state-of-the-art analysis provides advantages and limitations of the available approaches. After this analysis, a novel approach for service matchmaking that address the issues is modelled through an execution process. Finally, an overview of the techniques that must be developed to implement the process is shown.

3.1 Service discovery: state of the art

Since the beginning of the SOC research area, a plethora of service discovery approaches and tools have been proposed. In this section, works that are most popular and relevant for the aim of this thesis will be discussed.

First works on service discovery have been proposed approximately ten years ago. Two main approach categories have been proposed: approach based on information retrieval and matchmaking [7, 8]. The first category of solutions adopt information retrieval techniques on textual documents. Instead, the latter one propose to compare semi-structured documents that specify service properties.

After few years, the focus of discovery research moved on matchmaking of service descriptions defined according to Semantic Web standards. As introduced in the previous chapters, semantic descriptions enable a more effective matchmaking by addressing homonymy and synonymy between heterogeneous vocabularies defined requesters and providers [9].

Currently, the most cited semantic matchmaking tools in literature are: the semantic UDDI matchmaker proposed by Sycara et al. [9], METEOR-S [34], OWLS-MX [35], WSMO-MX [36], iSeM [37] and URBE [38]. The tool proposed by Sycara et al. [9] is the first one that propose to combine Web service and Semantic Web technologies. The matchmaker is modelled for SOAP-based services and manage semantic descriptions based on OWL-S model [39] (details on OWL-S are provided in section 4.1). Semantic descriptions are the result of a conversion of WSDL descriptions provided by a UDDI register. The matchmaking process is implemented through semantic reasoning which evaluates functional properties of services.

METEOR-S [34] implements an approach similar to the semantic UDDI matchmaker proposed by Sycara et al. [9]. The new aspect introduced is the evaluation of Quality of Service (QoS) [33] in terms of response time, cost and reliability. Semantic reasoning is adopted for evaluating FPs, then mathematical evaluations of QoS are exploited for ranking the results of the functional evaluation.

OWLS-MX [35] and WSMO-MX [36] performs the same matchmaking approach on semantic documents defined according to, respectively, OWL-S and WSMO model [40] (details on WSMO are provided in section 4.1). The two tools implements and hybrid approach based on semantic reasoning and information retrieval techniques, in order to exploit non-semantic information available in WSDL descriptions associated with ontologies. The two matchmaker are effective in terms of precision and recall, but they evaluate only input and output parameters of SOAP-based services.

Instead, iSeM [37] is a matchmaker that extends the approach implemented by OWLS-MX and WSMO-MX. In addition, iSeM [37] introduces the evaluation of service preconditions (e.g., user account registration to a service) and effects (e.g., shipment of product after a purchase on an e-commerce service). Moreover, iSeM exploits a more effective combination of reasoning and information retrieval techniques.

Finally, URBE [38] evaluates service interfaces defined in WSDL that are annotated with semantic descriptions. By giving an example of WSDL document as input, URBE is able to find a service that provide a similar interfaces in terms of input and output parameters and operations. The approach is based on string similarity techniques applied to operations and parameters names.

All these tools share several common characteristics. Service matchmaking is mainly performed on functional properties. Moreover, the prototypes are designed exclusively for SOAP-based services. As introduced in chapter 2, most of Web APIs are RESTful services, therefore approaches that consider only classical Web services based on SOAP are unsuitable.

To the best of author's knowledge, XAM4SWS [41] is the only semantic matchmaker for RESTful service available in literature. XAM4SWS evaluates functionalities of semantically annotated RESTful services. Its matchmaking approach computes similarity between provided functionalities by exploiting semantic reasoning.

Service matchmaking approaches for SOAP-based services that covers generic non-functional properties are presented in [42–49]. A hybrid architecture for service selection based on reasoning and constraint programming techniques is proposed in [42]. Reasoning is exploited for functional discovery, then constraint programming is adopted defining and evaluating QoS. The main limit of this approach is that advantages of Semantic Web technologies are not exploited for QoS evaluation.

An approach for the service discovery based on the normalization of QoS values is described in [43]. The approach exploits a shared ontology as vocabulary for defining QoS between requesters and providers. The evaluation is performed by combing QoS values that are normalized in numeric interval in range $[0, 1]$.

A QoS-based discovery tool is proposed in [44]. The discovery engine is based on an algebraic discovery model that evaluates QoS expressed through semantic descriptions.

Even though the tool exploits semantic descriptions, it performs only keyword-based matchmaking without reasoning activities for identify requested properties.

A NFP-based service selection approach that modifies the Logic Scoring Preference (LSP) method with fuzzy logic operators is proposed in [45]. LSP is a professional evaluation method initially designed for solving hardware selection problems. The proposed approach combines LSP with fuzzy logic for evaluating quantitative NFPs. However, qualitative properties defined through symbolic values are not supported.

[46] propose a framework for NFP-based service selection. The framework combines reasoning for identifying matching properties and evaluating qualitative properties, then optimization methods based on linear programming are exploited for evaluating numeric properties. The approach is effective, but does not support users for specification of advanced constraints, such as numeric intervals or sets of values.

The first full semantic matchmaker that has a complete support of FPs as well as NFPs is IRS-III [47]. The matchmaking is completely performed through reasoning. IRS-III is the first tool that introduce mediation ontologies that maps concepts defined through heterogeneous vocabularies defined by requesters and providers. Previous approaches suppose that requesters and providers adopts a shared ontology as vocabulary for defining property values. A limitation of this tool is the adopted model that consider properties as a name-value couples. Therefore, IRS-III suffers the same limits of [46].

In [48], another semantic matchmaker, called EASY, that considers functional aspects as well as NFPs is proposed. EASY is the first work that considers efficiency problems introduced by intensive semantic reasoning. The adopted approach is based on optimization of ontology descriptions and minimization of reasoner usage. EASY supports the definition of inequalities for numeric values (e.g., $\leq$ and $\geq$), buy does not consider bounded interval (e.g., "price between 5 and 10 euros") and the definition of symbolic value sets (e.g., "payment method: credit card and PayPal").

To the best of author's knowledge, the PoliMaR [49] is the tool that manages the most expressive NFP descriptions. This tool has the big advantage to exploit a semantic meta-model that is complete independent of the language and model used to define descriptions therefore can be easily adopted for RESTful services as well. PoliMaR is

able to evaluate property values defined as bounded or unbounded numerical intervals and symbolic value sets. However, the approach does not support functional properties.

All the above mentioned approaches, assume the availability of rich semantic descriptions in local repositories. As introduced in chapter 2, few semantic descriptions of real services are available on the Web.

In the literature, the tool that addresses service discovery by exploiting Web information is Seekda! [50]. It is a discovery engine that performs semantic matching of SOAP-based services. It uses a keyword-based semantic search method, and allows for filtering over few fixed NFPs (e.g. service availability, rating). It crawls WSDL descriptions from the Web and but not supports dynamic changes. Moreover, semantics is used only for semantically annotating functional aspects of the services. Therefore Seekda! does not consider the problem of extracting semantic values from descriptions of non-functional properties.

Three other approaches that provide matchmaking functionalities on descriptions available from Web sources have also been proposed [51–53]. These approaches do not aim to extract semantic descriptions and support only keyword-based search. The tool proposed in [51] is focused on SOAP-based services published through public UDDI registries. Today, public UDDI registers are not available on the best of author's knowledge. In the past, IBM, Microsoft and SAP provided public registries, but all the service repositories have been shuted down during 2007. Moreover, the approach does not consider that descriptions of same services can be provided by several sources.

Instead, [52] and tapia2011simplifying exploit ProgrammableWeb repository, then support keyword-based search of Web APIs. The novelty introduced by these approaches is exploiting the social information on ProgrammableWeb to evaluate Web API popularity in terms of service usage. Only [52] and tapia2011simplifying considers a purely subjective property (service popularity) in literature, to the best of author's knowledge.

Finally, [54, 55] are matchmaking tools that partially cover quality assessments of the service descriptions. These two works evaluates the trustworthiness of the descriptions published by service providers. The approaches are designed for SOAP-based services and trustworthiness is measured only on QoS through the deployment of trusted QoS monitoring services.

TABLE 3.1: Feature summary of service discovery engines in literature

Tools	[9, 34–38]	[41]	[42]	[43]	[45, 46]	[44]	[47, 48]	[49]	[50]	[51]	[52, 53]	[54, 55]
Semantic matchmaking	✓	✓	~	✓	~	~	✓	✓	~	✗	✗	✗
SOAP-based services support	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RESTful services support	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
Web APIs support	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
FP support	✓	✓	✓	✗	✗	✗	✓	✗	~	✗	~	~
NFP support	✗	✗	~	~	✓	~	✓	✓	~	✗	~	~
Keyword-based search	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗
Requester property constraints definition	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Exploiting Web descriptions	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗
Properties from social media	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	~	✗
Management of dynamic information	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Quality assessments	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	~
Management of contradictory information	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

✓= Full feature
~= Partial feature
✗= Unsupported feature

A feature summary of the discovery engines described above is provided in table 3.1. The results of the state-of-the-art analysis show that the most of the matchmaker exploits semantic descriptions and assume that these descriptions are available, but only few semantic descriptions of real services are published on the Web, as shown in chapter 2. Therefore, these tools are not applicable on heterogeneous Web descriptions. Instead, the most of the approaches that use existing Web descriptions exploits information retrieval techniques without the support of semantic technologies. In general, tools that full support NFPs perform semantic matchmaking, because ontologies are able to represent the complex nature of these properties. NFPs can cover several domains, be subjective, qualitative or quantitative, therefore requires advanced evaluations that state-of-the-art information retrieval techniques do not support. Finally, there are not tools that consider description dynamism and manage contradictory information about same services provided by different sources.

3.2 Service matchmaking exploiting the Web

The novel approach for enabling service matchmaking on Web descriptions [11], is represented in figure 3.1 by an UML 2 activity diagram[1]. Rounded rectangles represent activities of the process, instead rectangles represent data flows that are input or output of activities.

Designed as a process, the approach is composed of four main activities. This first activity extracts property values from each Web source that provides service information. After that, the property values are aggregated for each Web API and transformed in a set descriptions according to a unique shared model. Two different extraction techniques must be used. The first one for properties explicit on the Web and the latter one for subjective properties. In this process phase the management of the Web information dynamism is addressed.

The second activity is the quality evaluation of the Web sources. As introduced in chapter 2, descriptions can be provided by sources with different level of trustworthiness, currency and accuracy. The result of this evaluation is a set of quality assessments that defines the validity of each extracted property value.

[1]UML definition is available at: `http://www.uml.org/`

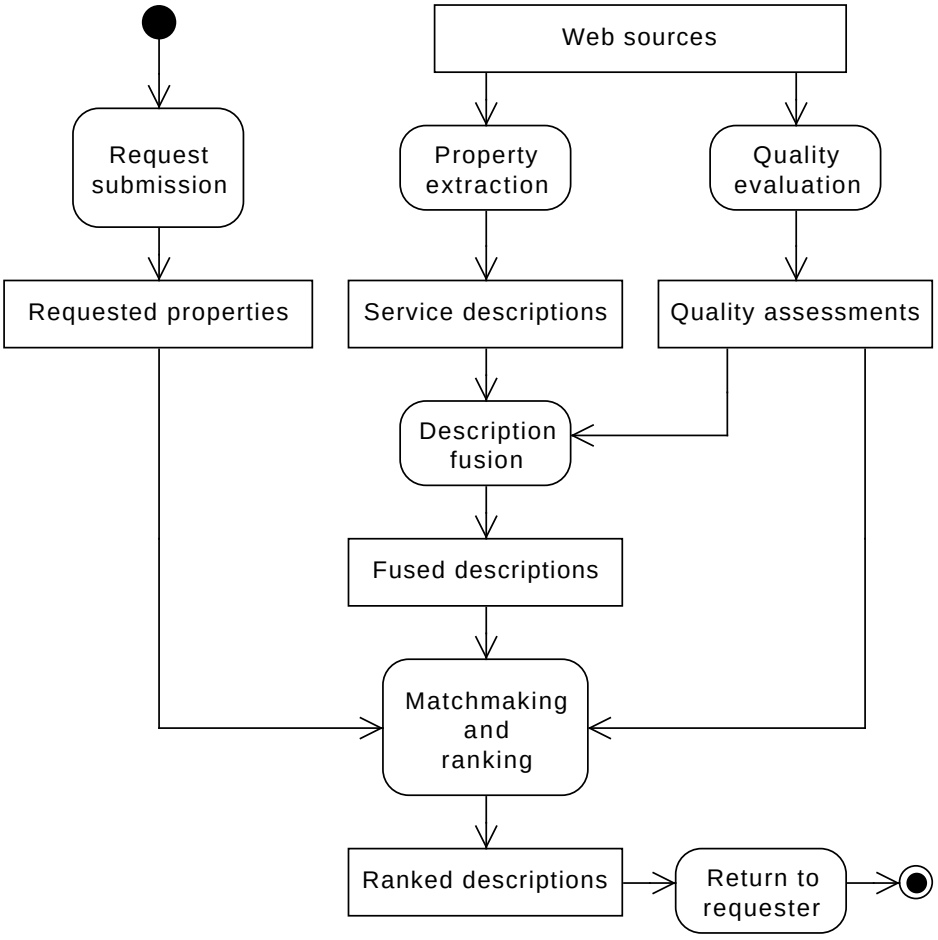

FIGURE 3.1: The process of extraction, fusion and matchmaking of service descriptions available on the Web

Afterwards, the description fusion is performed. Same properties of a single service can be described in several Web sources. In this phase, descriptions associated with the same service are fused by exploiting the quality evaluations returned by the previous phase. Depending on the property, different fusion function are adopted, such as selection of the best value or value composition. The description fusion is driven by quality assessments in order to select or compose the most valid values that are extracted from the Web.

The latter phase is the matchmaking between the request that defines an ideal representation of service that fulfils user needs and the fused descriptions. After a selection of the descriptions that contains at least one of the requested properties, the evaluation of a local matching score between values of requested and fused properties is performed. Then, local matching scores are combined with the quality assessments in order to assign a global matching score to each description. The global scoring is exploited to rank

the descriptions. Finally, a ranked list of descriptions, then services, that match the requested properties is returned to the requester. Techniques for quality assessments of Web descriptions are considered because matchmaking on unreliable information can provide wrong results to service requesters.

3.3 Implementing the overall approach

The implementation of the proposed process requires the development of techniques, models and software architectures for addressing six approach components. These components and associated techniques are described as follows.

Service description

In order to manage the heterogeneity of the Web information, the definition of a unique model for describing services is necessary. The solution proposed in this thesis is to adopt Semantic Web standards to define service descriptions according to the *Lightweight policy-centered meta-model* (PCM-lite). As previously introduced, Semantic Web technologies are able to infer relations between concepts, then to address synonymy and homonymy between terms. The PCM-lite is a meta-model for describing service properties and user requests. Its main features are: (i) language independence, (ii) high expressiveness, (iii) a low complexity for efficient evaluations and (iv) compatibility with existing models. Details of this solution are provided in chapter 4.

Property extraction from Web Sources

Service properties can be explicitly available on the Web or interpreted in a subjective way by users. Moreover, the Web information is dynamic, therefore this characteristic must be managed. For extracting explicit properties from available semantic descriptions a technique based on semantic mappings between PCM-lite and source models is defined. The proposed approach extracts properties from semi-structured documents and free text and combines: (i) the definition of *source-to-policy templates* (S2PTs), which specify portions of text that contains property values and (ii) named entity recognition, for identifying property values according to a domain ontology. The approach adopted to evaluated subjective properties exploits social media. Social media, such as social networks, blogs and forums,

contains user opinions that can be aggregated to provide a common evaluation about a service property (e.g., provider popularity). These issues are respectively addressed in chapter 5, for explicit Web information, and chapter 6 for subjective properties.

Quality assessments of service information

Web information can be inaccurate, out-of-date or untrustworthy. In chapter 7, techniques for evaluating *accuracy*, *currency* and *trustworthiness* of Web information are proposed in order to perform matchmaking on valid data. Accuracy is computed as a score that expresses the similarity between the concepts identified through named entity recognition. More the concepts are similar, then refer to the same domain or context, higher is the accuracy. Currency is evaluated by exploiting the publication date of the Web document, if available. Finally, trustworthiness is measured by exploiting user activities on social networks. More users that follow a Web source (e.g., Wikipedia) on social networks are active, implies that the information provided by the source is trustworthy.

Information fusion

Several Web sources can provide same information about a specific service. Chapter 7 also provides several techniques for fusing information about service properties to produce a unique, rich and trustworthy description for each service. The approach proposed is based on the definition of semantic mappings between properties defined in different sources. Each mapping is associated with a fusion function that depend on the nature of the property (e.g., numeric average, selection or aggregation of concepts). These functions are driven by quality assessments in order to provide valid fused information.

Service matchmaking

Reasoning is an effective tool for effective semantic matchmaking. However, reasoner are not designed for numerical evaluations and assume that semantic descriptions to be evaluated are valid. In chapter 7, a novel approach for service matchmaking is also proposed for addressing these issues. The approach combines: (i) reasoning for evaluating semantic relation between property values represented as concepts (e.g., data licenses); (ii) mathematical functions for evaluating numerical property values (e.g., usage fees or response time); (iii) quality assessments in order to rank matched services according to the user requirements and the property value validity.

Global scalability

A tool that performs extraction, quality assessments, fusion and matchmaking of service descriptions is a complex system. The Web information about services is continuously increasing. The extraction, management and evaluation of a big amount of data require a huge amount of computational resources. To perform the global matchmaking approach through a centralized system and each time an user request is submitted can require a huge response time that depends on: (i) network traffic and Web source response time, for extracting properties and (ii) reasoners that infers on semantic descriptions in exponential time according to the ontology complexity [10]. The *Policy Matchmaker and Ranker for Web* (PoliMaR-Web) is a scalable tool that implements the global approach by considering these aspects. PoliMaR-Web is based on a distributed architecture implemented by lightweight services. Each component performs extraction, fusion or matchmaking on a subset of descriptions in order to increase performances and scalability. Details on PoliMaR-Web are available in chapter 8.

Techniques and methods developed for addressing these six approach components involve different research areas in addition to the SOC, as shown in figure 3.2. The service description mainly involves Semantic Web area because ontologies are exploited to model service properties. The property value extraction touches four research areas. The identification of properties available on Web descriptions that are heterogeneous for data format is a information extraction problem. Instead, the extraction of subjective properties involves the research on Social Media Analysis. Of course, the transformation of the extracted properties to ontologies is a Semantic Web issue. Software engineering is involved to manage the dynamism of the information available on the Web. To provide quality assessment of Web information touches the Data Quality research area. The quality evaluation involves also Information Extraction, for data currency measurement, and Social Media Analysis, to estimate the trustworthiness of Web sources through social networks. The description fusion is essentially a Data Integration issue but it also touches the Semantic Web area. To be able to identify fusible properties e semantic evaluation must be performed. Moreover, an effective fusion needs the correct evaluation of the semantic equivalence or subsumption of property values. To provide an effective matchmaking on service descriptions defined through ontologies is clearly a Semantic

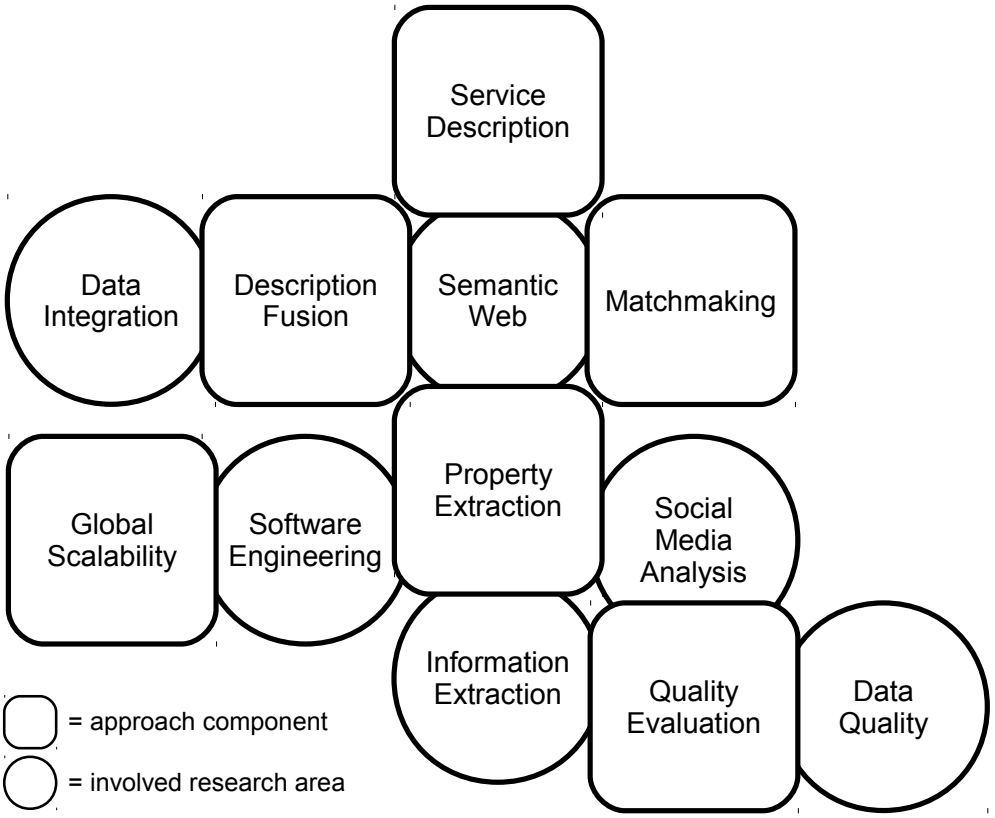

FIGURE 3.2: Research areas involved by addressing development of components of the overall approach for enabling matchmaking on the Web

Web issue. Finally, addressing the scalability of the global approach is a pure Software Engineering problem.

In next chapters, solution will be described in detail for each issue, after a deep analysis of related approaches and tools available in literature.

Chapter 4

The Representation of Real Services Available on the Web

Information about real services available on the Web is published through heterogeneous documents according several semi-structured models or free text descriptions. In order to perform effective matchmaking, the definition of unique description model that is able to represent service properties is necessary. In this chapter, the *Lightweight Policy-Centered Meta-model* (PCM-lite) is proposed for addressing description targeting any kind of service. The main features of PCM-Lite are: (i) language independence, (ii) high expressiveness, (iii) a low complexity for efficient evaluations and (iv) compatibility with existing models. After an analysis of the descriptions models available in literature, details of PCM-lite are provided.

4.1 Service description: state of the art

A plethora of standards and models have been proposed in the literature to represent service characteristics for enabling their discovery. Some of them are tied to SOAP-based or RESTful services, however they can mainly classified in models that are compliant or not with the Semantic Web standards. The most popular non-semantic languages are WSDL and WADL. Instead, SAWSDL and SA-REST have been proposed for binding, respectively, SOAP-based and RESTful service with descriptions according

to the Semantic Web. Finally, OWL-S, WSMO, MicroWSMO, and Linked-USDL are the most popular semantic model for describing services.

4.1.1 Non-semantic models

As introduced in chapter 2, the *Web Service Description Language* (WSDL) [56] is a XML-based standard proposed by *World Wide Web Consortium* (W3C), that was originally designed for describing classical Web services based on SOAP.

Despite the most recent version of this language (2.0) [57] have been designed to also support REST, the most widely adopted version is 1.1 which supports only SOAP, but is compliant with the WS-BPEL [18], the most adopted standard for orchestrating application composed of Web services.

WSDL 1.1 semantics is based on seven terms that specify functional interfaces of services.

Service

A *service* is the root concept of the language and is represented by a set of related *ports* together.

Port

Each *port* defines the Web address of a service and is typically represented by a URL. Through this address, the service is reachable and invokable.

Binding

Bindings specify the application protocol adopted for transporting SOAP messages (e.g., HTTP, SMTP, FTP, etc.) and the SOAP binding style. The binding styles define how is structured the message. Two styles are supported: (i) *RPC*, which means that SOAP messages contain parameters and return values, or (ii) *document*, which specifies that messages are defined according a user defined XML schema. Moreover, *bindings* associate *ports* with *operations*.

Port Type

The *Port Type* is a collection of abstract *operations* that represent the service interface.

Operation

An *operation* defines which message represents and input or output of a specific

functionality. *Operations* can be considered a method or function call in a traditional programming language.

Message

A *message* defines XML schemas associated with each part of a SOAP message that contains the input or output of a service *operation*.

Types

Types describe the data type associated with each input or output parameter (e.g., integer, string, etc.) of *operations* that are defined through *messages*. The XML Schema language, also known as XSD, is used as vocabulary for defining data types.

The changes introduced by WSDL 2.0 are mainly four. *Port types* are renamed to *interfaces* and *ports* are renamed to *endpoints*. The *message* term is removed. XML schemas that define input and output messages are associated directly with *operations*. A further semantics is added to the language for supporting REST description (operation can refer to HTTP methods).

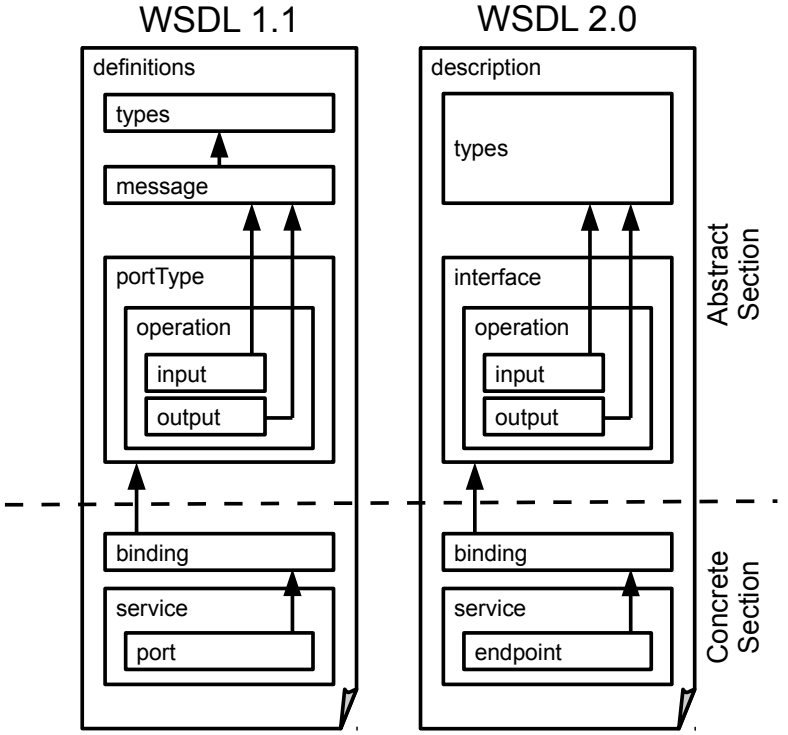

FIGURE 4.1: Comparison between WSDL 1.1. and WSDL 2.0 semantics

In figure 4.1 is provided a comparison between the two versions of the language. The abstract part defines the *interface* of the service composed of *operations*. Instead, the

concrete part provides the *endpoints* that allow requester to invoke the services that are mapped with the abstract functionality definitions through *bindings*.

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl" ... ">

<!-- Abstract type -->
   <types>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" ... >
         <xs:element name="request">
            ...
         </xs:element>
         <xs:element name="response">
            ...
         </xs:element>
      </xs:schema>
   </types>

<!-- Abstract interfaces -->
   <interface name="RESTfulInterface">
      <operation name="Get" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
      <operation name="Post" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
      <operation name="Put" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
      <operation name="Delete" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
   </interface>

<!-- Concrete Binding Over HTTP -->
   <binding name="RESTfulInterfaceHttpBinding"
      interface="tns:RESTfulInterface"
            type="http://www.w3.org/ns/wsdl/http">
      <operation ref="tns:Get" whttp:method="GET"/>
      <operation ref="tns:Post" whttp:method="POST"/>
      <operation ref="tns:Put" whttp:method="PUT"/>
      <operation ref="tns:Delete" whttp:method="DELETE"/>
   </binding>

<!-- Concrete Binding with SOAP-->
   <binding name="RESTfulInterfaceSoapBinding"
      interface="tns:RESTfulInterface"
      type="http://www.w3.org/ns/wsdl/soap"
      wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
      wsoap:mepDefault="http://www.w3.../soap/mep/request-response">
      <operation ref="tns:Get" />
      <operation ref="tns:Post" />
      <operation ref="tns:Put" />
      <operation ref="tns:Delete" />
   </binding>
```

```
<!-- Web Service offering endpoints for both bindings -->
   <service name="RESTfulService" interface="tns:RESTfulInterface">
      <endpoint name="RESTfulServiceHttpEndpoint"
                binding="tns:RESTfulInterfaceHttpBinding"
                address="http://www.example.com/rest/"/>
      <endpoint name="RESTfulServiceSoapEndpoint"
                binding="tns:RESTfulInterfaceSoapBinding"
                address="http://www.example.com/soap/"/>
   </service>
</description>
```

Listing 4.1: An example of WSDL 2.0 description

Listing 4.1 shows an example of a service that provide the same functionalities through both SOAP and REST. The service has an abstract interface composed of four operations, namely *get*, *post*, *put* and *delete*, that have inputs and outputs messages defined according the same XML schema. Two bindings are described in order to specify how the abstract interface is implemented according REST and SOAP. The REST binding maps each operation with the homonym HTTP method. Instead, the second binding define that are provided four functions trough SOAP over HTTP, specified by the attribute `protocol`, and the iteration is RPC, as reported by the attribute `mepDefault`. Two different endpoints are specified for the interactions according REST and SOAP.

Most of the providers that offer SOAP-based services publish WSDL descriptions. WSDL supports the representation of service functionalities without support for the definition of NFPs. Moreover, comparing interfaces is not enough for matchmaking service functionalities. In facts, equivalent services may support parameters with different schemas, or different services may support parameters with the same schema. To give an example, two services that respectively provide shipments and taxi bookings have both address and payment method as parameters.

For describing HTTP-based applications and RESTful services Sun Microsystems proposed the *Web Application Description Language* (WADL) [26]. WADL is based on XML, as well as WSDL, and its semantics is based on the concept of *application* that is mainly a collection of *resources* according to REST principles. Several HTTP methods can be associated with a *resource* trough the term called *method*. *Request* and *response* tags are associated with *methods*. *Requests* represents the data format of request messages and *responses* define the format of response messages. These two tags can be associated with a *representation* that defines the Internet media type [19] or the XML schema of

the message. To follow the HATEOAS principle, hyperlinks between *representations* can be defined by the term *link*.

An example of WADL description is provided in listing 4.2. The document defines a news search service that provide one resource identified by `http://example.com/newsSearch`. The resource is accessible through HTTP GETs. As specified by the "query" attribute, the resource has an input parameter that must be specified in the URI (e.g., `http://.../newsSearch?keywords=formula_one_gran_prix`). A request submission can return a HTTP 200 status (successful request) with a XML message defined by the `yn:ResultSet` or a HTTP 400 status (bad request) with another XML message defined by the `ya:Error`.

```
<application xmlns="http://wadl.dev.java.net/2009/02">
  <resources base="http://example.com/">
    <resource path="newsSearch">
      <method name="GET" id="search">
        <request>
          <param name="keywords" type="xsd:string"
                 style="query" required="true"/>
        </request>
        <response status="200">
          <representation mediaType="application/xml"
                          element="yn:ResultSet"/>
        </response>
        <response status="400">
          <representation mediaType="application/xml"
              element="ya:Error"/>
        </response>
      </method>
    </resource>
  </resources>
</application>
```

Listing 4.2: An example of WADL description

WADL was submitted to the W3C 31 August 2009, but the consortium has no plans to standardize this language and it is not yet widely supported and adopted. Moreover, like WSDL, WADL do not support the definition of non-functional characteristics. In addition, the descriptions of resources is not enough for an effective matchmaking, because same functionalities can be implemented in several ways trough REST. Despite model limits, non-semantic descriptions share several limitations for matchmaking.

The identification of different terms that refer to the same concept (synonymy) is not supported. Therefore, if two service descriptions define the same operation with different terms, a string-based matching considers the two operations different each other.

Homonym terms are also unsupported. Hence, operation with the name but implement different functionalities are considered the same. For addressing these issues, Semantic Web provides several standards and technologies.

4.1.2 Semantic models

The base model for general-purpose descriptions according to the Semantic Web is the *Resource Description Framework* (RDF) [29]. RDF is defined as a specification by the W3C and is based on a set of statements in form of subject-predicate-object expressions, also called *RDF triples*. Each element of a triple is univocally identified on the Web by an URI that can be summarized through prefixes and namespaces like XML. Several data format can be adopted to express semantic descriptions, also called ontologies, according to RDF. The most popular are XML, N3 [58], Turtle [59] and N-Triples [60]. To give an example, the sentence "cat is an animal" can be expressed through a RDF triple defined in the listing 4.3 according to N-Triples. In this example, `animals` is a prefix that summarize the namespace of an ontology of animals defined by the URI `http:/example.org/animals`.

```
@prefix animals: <http://example.org/animals>

animals:cat animals:is-a animals:animal .
```

Listing 4.3: A simple RDF document according N-Triples

Then, W3C extended RDF through the specification of *RDF Schema* (RDFS) that defines the RDF vocabulary [61]. This vocabulary permits to define additional concepts such as, classes, subclasses, instances of classes and properties of the ontology objects. To show another example, the sentences "Alice is a human" and "animal is a generalization of human" can be defined by the listing 4.4 according to RDFS. The first triple define that `animal` is an instance of class. Then, `human` is a subclass of `animal`. Finally, `Alice` is an instance of the `human` class.

During February 2004, the *Web Ontology Language* (OWL) [62] is proposed as W3C recommendation in order to additionally extend RDF/RDFS. This vocabulary extension defines new concepts, such as semantic equality between concepts, transitive and symmetric properties, enumerations and set theory concepts (union, intersection and complement).

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix animals: <http://example.org/animals>
@prefix people: <http://example.org/people>

animals:animal rdf:type rdfs:Class .
animals:human rdfs:subClassOF animals:animal .
person:Alice rdf:type animals:human .
```

LISTING 4.4: A simple RDFS document according N-Triples

The definition of concepts and relations according to RDF/OWL permits to address several issues, such as homonymy, synonymity and multi-language support (e.g., by defining equivalence relations between English and Italian terms). Moreover, the availability of tools that perform *automatic reasoning*, that are able to infer relations that are not explicitly defined, allows advanced evaluations on semantic description and a seamless integration between several ontologies. To clarify how these tools works, on the three triples defined in listing 4.4, a *reasoner* is able to automatically infer that `Alice` is an instance of the class `animal` despite the two concepts are defined in two different ontologies and the relation is not explicitly defined in the document.

The first semantic model proposed for service description is OWL-S [39]. Based on OWL, it is designed for classical SOAP-based services and each service is described can be described by three concepts:

- *service profile*, that defines functional properties (input, output and parameters) and a textual description;
- *service model*, that describe the behaviour of the service as a process;
- *service grounding*, that defines bindings between the OWL-S description and the WSDL document of the service.

This model can not be adopted to describe Web APIs because does not support REST and other interaction protocols based on HTTP. Moreover, OWL-S do not consider NFPs, therefore to perform matchmaking needs additional model for defining non-functional characteristics. However, through the modelling of the service behaviour, OWL-S allows a more rich representation of the service functionalities compared with WSDL. To perform matchmaking by exploiting OWL-S descriptions, users must define at least a *service profile* or a *service model* that represent an ideal service that fulfils his requirements.

Nevertheless, the modelling of service behaviours requires a huge effort and expertises for users that usually perform discovery of information through keyword-based searches.

Another proposed model for semantic service descriptions is the *Web Service Modelling Ontology* (WSMO) [40]. WSMO is mainly based on four concepts:

- *Web service*, that describe the functional characteristics and NFPs of a service;
- *goal*, which is the description of a service that fulfils requester needs.
- *ontology*, that defines the semantic description of a specific domain (e.g., a medical domain for hospital services);
- *mediator*, that define semantic links between goals, ontologies and Web services.

WSMO is the first model that consider the definition of non functional properties. However, it mainly support the definition of 28 kinds of NFPs that specifies QoS or document meta-data, such as author, owner and version.

One of the advantages introduced by WSMO are *Mediators*, that enable to map different ontologies adopted as vocabularies by requesters and providers. *Mediators* define relations between concepts that enable to address homonymy and synonymy through inferences of reasoners. In this way, requesters and providers are able to exploit their own vocabularies for defining, respectively, requests as *goals* and service descriptions as *Web services*. Four kind of mediators are defined by WSMO:

- *ggMediator*, that links goals in order to define state of equivalence;
- *ooMediator*, that defines cross-domain links between ontologies;
- *wgMediator*, which links services and goals in order to specify which service properties fulfils goal properties;
- *wwMediator*, that define equivalence links service properties.

Unlike OWL-S, WSMO is not based on OWL, but on the *Web Service Modelling Language* (WSML) [63]. WSML-based descriptions are based on a data format that is more human readable compared to other data format that support RDF/OWL modelling. Nevertheless, WSML semantics can be defined also through plain XML and RDF/RDFS.

```
...
<span class="domain-rel" title="sarest:method" >
  The resource representation can be retrieved through a
  <span class="domain-rel" title="http://www.w3.org/2011/http-methods#GET">
    HTTP GET
  </span> .
</span>
...
```

Listing 4.6: A portion of HTML description enriched with SA-REST

In order to bind WSDL descriptions with services described in WSML, the WSMO working group propose to adopt *Semantic Annotations for WSDL* (SAWSDL) [64]. SAWSDL is a W3C recommendation that propose to associate WSDL elements (operations, parameter, etc.) with URIs that refers to concepts of an ontology. This WSDL extension is not tied to a specific semantic service model, therefore it allows mapping with WSMO/WSML, OWL-S, or any user defined models. In the listing 4.5, a portion of SAWSDL document is provided. The example specifies a mapping between a WSDL operation with a concept identified by the URI `http://example.org/purchaseorder`.

```
...
<wsdl:operation name="order" pattern="http://www.w3.org/ns/wsdl/in-out"
      sawsdl:modelReference="http://example.org/purchaseorder">
  <wsdl:input element="OrderRequest" />
  <wsdl:output element="OrderResponse" />
</wsdl:operation>
...
```

Listing 4.5: A portion of SAWSDL description

Instead, *Semantic Annotations for REST* (SA-REST) have been proposed to map RESTful service resources with ontologies. As introduced in chapter 2, most of Web APIs, that are mainly composed of RESTful services, are described by HTML documents. In this scenario, SA-REST propose to enrich HTML pages with meta-data that maps resources descriptions with semantic concepts defined by a specific semantic models. The semantic model represent approximately the same concepts of WADL, for exceptions of links between resources. The SA-REST mappings are defined through the specification of the concept URI as a `title` attribute (according to the HTML standard) to the tag that embed the resource description. The listing 4.6 provides a portions of HTML document that contains SA-REST meta-data. The example shows how to map a portion of text that specifies the HTTP method enabled for a resource.

Another proposal for binding ontologies with HTML descriptions of RESTful services is the combination of *HTML for RESTful Services* (hRESTS) and *MicroWSMO* [65]. hRESTS is a microformat with the objective of the identification of FPs in Web pages by using the `class` attributes associated with tags that contain service URIs, operations or parameters. Listing 4.7 shows an example of HTML code enriched with hRESTS. The `div` tag classified as *service* contains the service descriptions. The service is composed of an operation identified by *op1* label. The operation is represented by a resource identified by the *address* `http://example.com/h/{id}`, where `id` is an input parameter. The resource can be retrieved through a HTTP GET and the output message is represented by a XML Schema (`ex:hotelInformation`).

```
<div class="service" id="svc">
<p>Description of the
   <span class="label">ACME Hotels</span> service:</p>
<div class="operation" id="op1"><p>
   The operation <code class="label">getHotelDetails</code> is
   invoked using the method <span class="method">GET</span>
   at <code class="address">http://example.com/h/{id}</code>,
   with <span class="input">the ID of the particular hotel replacing
        the parameter <code>id</code></span>.
   It returns <span class="output">the hotel details in an
        <code>ex:hotelInformation</code> document.</span>
</p></div></div>
```

Listing 4.7: A HTML description enriched with hRESTS

The hRESTS microformat is able to express the same information of WSDL or WADL and it is not semantic. For binding hRESTS witch semantic descriptions was introduced MicroWSMO as an extension of SAWSDL. MicroWSMO enriches hRESTS tags by adding attributes that contains URIs of WSMO-compliant ontologies.

SAWSDL, SA-REST, hRESTS and MicroWSMO are good proposals for promoting the diffusion of semantic service descriptions by binding adopted non-semantic standards. However, few descriptions according these models are available. Moreover, these proposals do not support non-functional characteristics.

The diffusion of information available through Linked Data has stimulated the design of a new semantic model called *Linked-USDL* [66]. Linked-USDL, currently in development, aim is to provide a model to describe service according Linked Data, also called *Linked Services* [67], through a remodelling of *Unified Service Description Language* (USDL) [68]. Compared to OWL-S and WSMO, this new model does not cover only software

service, but also human services (e.g., consultancy) and business services (e.g., purchase, order, requisition).

The Resource Linking Language (ReLL) [69] is a very rich model that allows providers to represent RESTful services by combing the advantages of REST and Linked Data. ReLL aim is providing a standard language for representing resources available through REST. This language is data format independent and provides a formal definition of resources and links in order to follow HATEOAS principle. Unfortunately, the model does not allow providers to specify NFPs.

RESTdesc has been also proposed in [70] to combine REST and Linked Data [71]. This approach is based on an extension of RDF/N3 descriptions that specifies the service functionalities as a set of preconditions that, combined with a user request, generate a specific post condition. A common characteristic that RESTdesc shares with our framework is the adoption of vocabularies that can be provided by the Linking Open Data Cloud. RESTdesc is able to model the behaviour of the service according to HATEOAS. However, the definition of descriptions is not straightforward for property interpretation by users and requires non-standard extensions of RDF/N3.

All proposed semantic models for service descriptions have some limitation for the definition of service properties. Proposed FPs are mainly designed to support automatic service composition and they marginally consider a human user point of view. The proposed models well define service operations and parameters, but typically a human user discovery is performed through a search based on keywords that describe the service functionality. Therefore, semantic models should permit to specify human-oriented FPs, such as semantic tags.

The main model limitations are about non-functional characteristics. OWL-S does not consider NFPs, therefore the adoption of this model needs an extension to support these properties. Instead, WSMO basically supports attribute-value descriptions of NFPs but them are not included in the logical model and thus automatic reasoning on non-functional characteristics is not possible. At the current state of the work, Linked-USDL defined models for service pricing and SLAs, but not for generic NFPs. Moreover, no one of the cited models is a *de facto* standard for semantic service descriptions.

In this scenario, there is a need of a semantic model that, on one hand, overcame the limitations of the state-of-the-art property definitions, and, on the other hand, is compliant to the existing semantic models. In [72], the *Policy-Centered Meta-model* (PCM) was proposed to address some of the highlighted limitations related to NFPs. Two main issues are addressed by the PCM in order to extend available semantic models.

Quantitative and qualitative values

NFPs can be expressed with numeric values defined in different units (e.g., price in Euro or in USD), or they can be purely qualitative (e.g., the usability is good, trust is high, the payment method is credit card).

Requester perspective

Service requester must be supported in the definition of his/her NFP requirements. Advanced mathematical operators (e.g., $\leq$, $\geq$, range) must be allowed for requested NFP specification as well as a relevance value stating if an NFP is mandatory or optional.

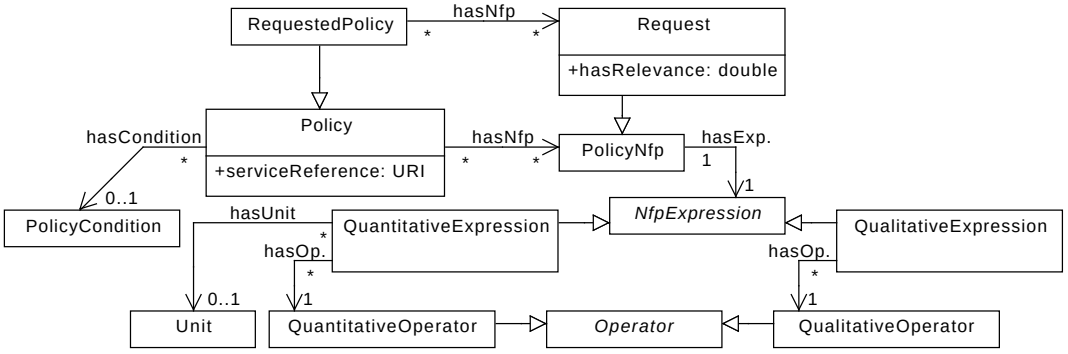

FIGURE 4.2: The Policy-Centered Meta-model (PCM)

In order to be compliant to OWL-S, WSMO and Linked-USDL, the PCM has been defined by a language-independent conceptual syntax, whose semantics is defined by the UML class diagram shown in figure 4.2. The conceptual syntax of the PCM has been defined by a BNF grammar, whose logical grounding is given by formalizations in both RDF/OWL and WSML[1].

[1]The complete formalizations are available at: `http://www.siti.disco.unimib.it/research/ontologies/`

The PCM supplies developers, providers and users with a frame for NFP descriptions that can be exploited to support service selection and ranking by matching service requests and service offers. The meta-model has the following characteristics:

- each service can be described by one or more *Policies* that is a contract that aggregates NFPs into a single entity;
- each policy is associated with a *PolicyCondition* that specify the for which user category the contract is applicable (e.g., users that subscribed an on-demand service usage);
- the explicit distinction between NFP offered (*PolicyNfp*) by providers and requested by users (*Requested Policy*)
- for each requested NFP (*Request*) the user can define a value that define how is relevant the property;
- a NFP is associated with a *QualitativeExpression* or *QuantitativeExpression* that is composed by an operator, one or more values and, for quantitative expressions, an optional unit;
- a *QualitativeOperator* is a logic quantifier ($\forall$, $\exists$) for a set of symbolic values;
- instead a *QuantitativeOperator* defines single numeric values, with $=$, unbounded intervals, with $\leq$ and $\geq$, or bounded intervals with *range*.

Through its characteristics, the PCM provides support for descriptions that reflect the user perspectives. The extensive use of constraint operators and relevance values in constraint expressions (e.g., the cost of the service must be $\leq$ 3 Euro) enhance the expressivity of the descriptions to support non-boolean matching (i.e., matching between offered and requested NFPs should be considered not crispy and degrees of satisfaction should be evaluated). Moreover, the aggregation of NFPs in policies supports the description of complex business scenarios.

Other works [73–75] propose to overcome WSMO and OWL-S limitations by supporting more sophisticated NFP descriptions. As shown in [72], even if these proposals succeed in covering some of the above mentioned aspects to be considered in NFP descriptions, none of them provides a solution able to cover all the above aspects.

4.2 PCM-lite: a lightweight meta-model for Web APIs properties

The PCM is the best candidate to be adopted for semantic description of Web APIs, but it has two main limits. The first limitation is the complexity. Each Web sources exploited to extract service information can adopt different terms to describe same property values. To address this issue for matchmaking is necessary to adopt an automatic reasoner. As shown in [10], to perform reasoning on complex models, therefore complex descriptions, require high computational resources then a high response time that is inadequate for Web applications. The letter one is the support of only NFPs. Therefore, to also adopt OWL-S, WSMO or Linked-USDL is necessary in order to implement a full matchmaking on functional and non-functional characteristics. Moreover, the result of the integration of the PCM with another model is more complex descriptions then higher response time for reasoners.

In this section, the *Lightweight Policy-Cetered Meta-model* (PCM-lite), a remodelling of the PCM that addresses these issues, is proposed in order to allow an effective matchmaking of Web APIs. PCM-lite objectives are:

- to support Web API characteristics (described in chapter 2);
- to be more simple, but expressive as well as PCM;
- to also support functional properties, in order to be independent to support functional matchmaking, but at the same time compliant to existing semantic models, in order to be able to integrate existing semantic descriptions.

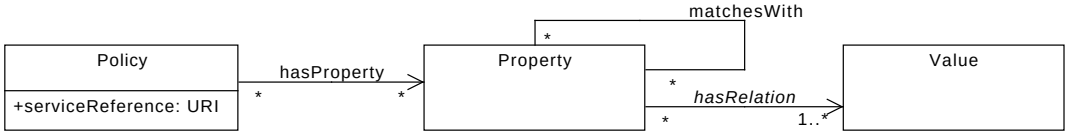

FIGURE 4.3: The PCM-lite formalization

The semantics of the PCM-lite is shown in figure 4.3 as UML class diagram. The main change of the original meta-model (highlighted in the diagram) is the substitution of the *PolicyNfp* and *NfpExpression* concepts with *Property*. The relation between NFPs

and expressions had a 1-1 cardinality, therefore it was redundant. PCM-lite does not define a distinction between functional and non functional properties. As introduced in chapter 2, the same property can be classified as functional or non-functional on the base of application domain or user context. The distinction between qualitative and quantitative properties is removed, because it is unnecessary. If properties assume numeric values are quantitative, otherwise, properties are qualitative. Another deleted concept is the *PolicyCondition*. Only few real service descriptions on the Web provide this kind of information. Moreover, the policy condition can be easily modelled as a *Property*. Several *Values* can be associated with *Properties* trough an abstract relation. Values can be semantic concepts, numbers or literal descriptions in natural language. The relation can be defined by user ontologies of models available in literature. However, PCM-lite semantics provides the following default relations that are instances of the abstract *hasRelation*.

hasValue

It represents a generic relation between a property and ad value. To give an example, "data license *hasValue* creative commons".

hasUnit

It associates an unit of measurement with a property. For instance, the "response time: 150 ms" can be represented trough two RDF triples: "response time *hasValue* 150" and "response time *hasUnit* milliseconds".

hasOperator

It associates an operator with a property. Operators can be mathematical operators, that defines bounded or unbounded numeric intervals, or set theory operators, for defining set of concepts, according to the PCM semantics.

hasRelevance

It associates a numeric value in range [0,1] with property. The value represent how is relevant the property for a requester.

The relation *matchesWith* defines which *property* instances are comparable each other. This relation has the same aim of *mediators* defined by WSMO, therefore supporting the adoptions of heterogeneous vocabularies defined by requesters and providers. The

relation *matchesWith* allow users to specify *matching mappings* according to the following definition.

Definition 4.1 (Matching mapping)**.** *A matching mapping is a relation which specifies two comparable properties. Matching mappings are transitive and symmetric relations.*

Mappings can be defined between properties offered by providers and requested by users. The second definition sentence means that:

$$a\ matchesWith\ b \implies b\ matchesWith\ a$$

and

$$a\ matchesWith\ b \wedge b\ matchesWith\ c \implies a\ matchesWith\ c$$

Matching mappings are a particular kind of semantic mappings as defined as follows.

Definition 4.2 (Semantic mapping)**.** *A semantic mapping is a relation that links two concepts defined in different ontologies that may be modelled by different authors.*

One of the main characteristics of the PCM-lite is the absence of a distinction between functional and non-functional properties. As shown in [72] and chapter 2, there is not an agreement about what properties should be classified functional or non-functional in literature. A possible reason should be that the classification of a property depends on the domain or the context. In addition, the absence of this categorization in the meta-model permits to more easily map PCM-lite descriptions with other existing models.

Another advantage of PCM-lite is the data format independence. The proposed meta-model supports the definition of descriptions in different languages, such as RDF, XML and JSON. However, in this thesis we adopt RDF/OWL for exploiting the advantages of Semantic Web technologies.

Compared with the other semantic models, PCM-lite allows describing Web APIs because is not tied to a specific service technology. Moreover, it permits to define additional FPs, such semantic tags, that enable human requester to perform a rich keyword-based discovery.

4.3 Representation according to PCM-lite

The aim of this section is providing several examples of PCM-lite descriptions in order to highlight the features of meta-model proposed in this thesis.

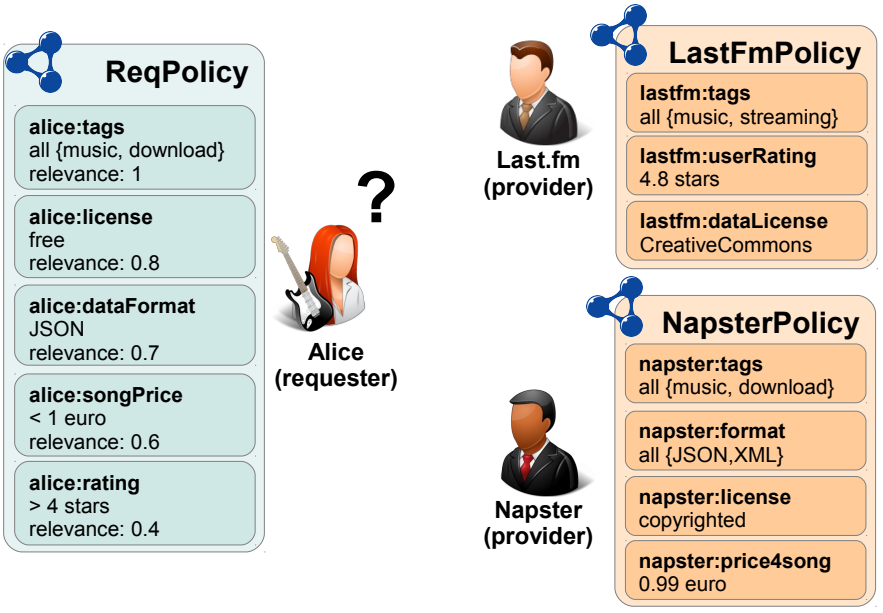

FIGURE 4.4: Example of requested and offered policies according to PCM-lite

In figure 4.4, an example of *Policy* instances according to PCM-lite is shown. The example models the following scenario. Alice, a music fan, is looking for a Web API which provides functionalities for music downloading. To develop her application, she needs an API that publishes data under a free license and she is very skilled in technologies that manage data in JSON format. Moreover, she prefers a service that permits to download music by paying less than 1 euro for song and high rated by users. Alice's requirements con be represented as an instance of a policy composed by five properties: *tags*, for the functional search, *license*, *data format*, *song price* and *rating*. The highest relevance is assigned to *tags* because is mandatory for the functional discovery. The other properties have lower relevance values according to Alice's needs. Two instances of offered policies are also shown in figure 4.4. The first one describes some characteristics of the Web APIs provided by Last.fm, a music social network that allows listening songs in streaming. Instead, the latter one represents properties of Napster a popular music download service.

Listing 4.8 provides the Alice's PCM-lite policy according to RDF/OWL. The example shows that external ontologies, such as DBpedia [76] can be exploited as shared vocabularies for defining values. Operators can be defined by adopting PCM-lite definitions as well as external ontologies. The operator `pcm-lite:all` asserts that the property assume all the values.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix alice: <http://example.org/alice>
@prefix pcm-lite: <http://pcm.disco.unimib.it/pcm-lite>
@prefix dbpedia: <http://dbpedia.org/resource/>

alice:ReqPolicy rdf:type pcm-lite:policy .

alice:ReqPolicy pcm-lite:hasProperty alice:tags .
alice:ReqPolicy pcm-lite:hasProperty alice:license .
alice:ReqPolicy pcm-lite:hasProperty alice:dataFormat .
alice:ReqPolicy pcm-lite:hasProperty alice:songPrice .
alice:ReqPolicy pcm-lite:hasProperty alice:rating

alice:tags rdf:type pcm-lite:property .
alice:tags pcm-lite:hasOperator pcm-lite:all .
alice:tags pcm-lite:hasValue dbpedia:Music .
alice:tags pcm-lite:hasValue dbpedia:Download .
alice:tags pcm-lite:hasRelevance "1" .

alice:license rdf:type pcm-lite:property .
alice:license pcm-lite:hasValue dbpedia:Free_content_licenses .
alice:license pcm-lite:hasRelevance "0.8" .

alice:dataFormat rdf:type pcm-lite:property .
alice:dataFormat pcm-lite:hasValue dbpedia:JSON .
alice:dataFormat pcm-lite:hasRelevance "0.7" .

alice:songPrice rdf:type pcm-lite:property .
alice:songPrice pcm-lite:hasOperator dbpedia:Less-than .
alice:songPrice pcm-lite:hasValue "1" .
alice:songPrice pcm-lite:hasUnit dbpedia:Euro .
alice:songPrice pcm-lite:hasRelevance "0.6" .

alice:rating rdf:type pcm-lite:property .
alice:rating pcm-lite:hasOperator dbpedia:Grater-than .
alice:rating pcm-lite:hasValue "4" .
alice:rating pcm-lite:hasUnit dbpedia:Star_(classification) .
alice:rating pcm-lite:hasRelevance "0.4" .
```

LISTING 4.8: A PCM-lite policy defined by a requester

Instead, the PCM-lite descriptions of the policy provided by Last.fm service is available in listing 4.9.

Listing 4.10 shows a mediation ontology that defines matching mappings between comparable properties defined in the requester and provider policies. The definition of all

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix lastfm: <http://last.fm/api>
@prefix pcm-lite: <http://pcm.disco.unimib.it/pcm-lite>
@prefix dbpedia: <http://dbpedia.org/resource/>

lastfm:LastFmPolicy rdf:type pcm-lite:policy .

lastfm:LastFmPolicy pcm-lite:hasProperty lastfm:tags .
lastfm:LastFmPolicy pcm-lite:hasProperty lastfm:dataLicense .
lastfm:LastFmPolicy pcm-lite:hasProperty lastfm:userRating

lastfm:tags rdf:type pcm-lite:property .
lastfm:tags pcm-lite:hasOperator pcm-lite:all .
lastfm:tags pcm-lite:hasValue dbpedia:Music .
lastfm:tags pcm-lite:hasValue dbpedia:Streaming .

lastfm:dataLicense rdf:type pcm-lite:property .
lastfm:dataLicense pcm-lite:hasValue dbpedia:Creative_Commons_license .

lastfm:userRating rdf:type pcm-lite:property .
lastfm:userRating pcm-lite:hasValue "4.8" .
lastfm:userRating pcm-lite:hasUnit dbpedia:Star_(classification) .
```

Listing 4.9: A PCM-lite policy defined by a provider

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix med: <http://example.org/mediaton>
@prefix lastfm: <http://last.fm/api>
@prefix lastfm: <http://napster.com/service>
@prefix alice: <http://example.org/alice>
@prefix pcm-lite: <http://pcm.disco.unimib.it/pcm-lite>

alice:tags pcm-lite:matchesWith lastfm:tags .
napster:tags pcm-lite:matchesWith lastfm:tags .

alice:license pcm-lite:matchesWith lastfm:dataLicense .
alice:license pcm-lite:matchesWith napster:license .

alice:dataFormat pcm-lite:matchesWith napster:format .

alice:songPrice pcm-lite:matchesWith napster:price4song .

alice:ranting pcm-lite:matchesWith lastfm:userRating .
```

Listing 4.10: A PCM-lite policy defined by a provider

mappings is not necessary, because the relation *matchesWith* is defined as symmetric and transitive. Therefore, for instance, the first 2 triples in the example description allow the inference of "`alice:tags pcm-lite:matchesWith napster:tags .`" trough automatic reasoning.

These mediation ontologies are necessary in order to enable reasoner to identify properties that match each other. These can be defined manually, by domain experts, or

automatically, by ontology matching tools, such as AgreementMaker[2].

The definition of semantic mappings is a strategy that can be also adopted for binding semantic models with PCM-lite in order to enable matchmaking between heterogeneous models. These mapping can state equivalence relations between properties defined in PCM-lite with FPs and NFPs defined in other models. The definition of semantic mappings is enabled by language independence of the PCM-lite that permits to define description compliant to RDF/OWL and WSML.

Mappings can be expressed by semantic relations in RDF/OWL (e.g., `rdf:type`) or written in a semantic rule language, such as *Semantic Web Rule Language*[3] (SWRL) or rules compliant to the *JENA framework*[4]. Through automatic reasoning, mappings between concepts are automatically inherited to instances of classes defined by model not compliant to PCM-lite.

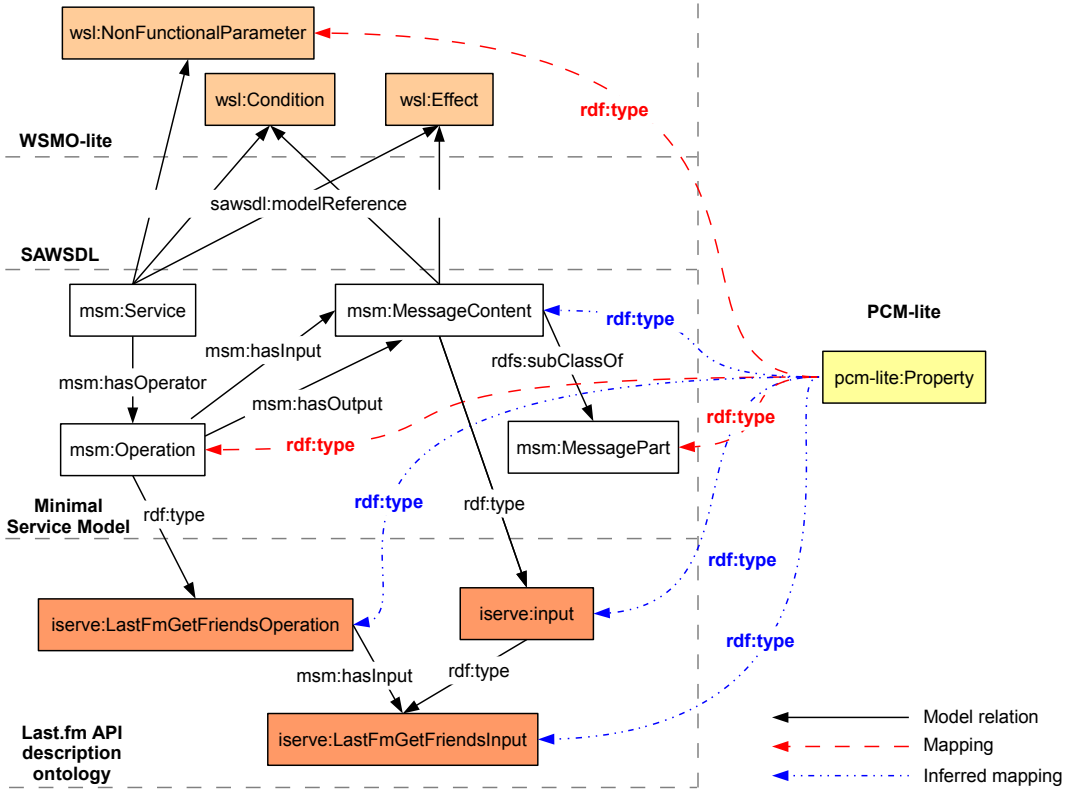

FIGURE 4.5: Example of mappings between PCM-lite and heterogeneous models

[2]Available at: `http://agreementmaker.org/`
[3]Definition available at: `http://www.w3.org/Submission/SWRL/`
[4]Definition available at: `http://jena.sourceforge.net/inference/`

Semantic mappings can be also defined with non-standard models. An example is described in figure 4.5. The diagram shows the mapping with the Minimal Service Model (MSM) adopted to represent services available in the iServe repository [32]. The MSM imports concepts defined by WSMO-lite[5], a lightweight remodelling of WSMO, and relations provided by SAWSDL model. In this example, mappings are defined as `rdf:type` relations between PCM-lite *Property* concept and *NonFunctionalParameter*, *Operation* and *MessagePart*. According to WSMO-lite, *NonFunctionalParameter* represents a generic NFP. Instead, *Operation* and *MessagePart* are FPs according to the MSM. The mappings are inherited by instances of the mapped concepts through reasoning inferences. In figure 4.5, are also represented some instances defined in an ontology that describes a Last.fm API. By performing reasoning, instances of MSM concepts became instances of *Property*. Mappings enable the transformation of heterogeneous semantic descriptions to a uniform, rich and lightweight model compliant to PCM-lite. In this way, performing effective matchmaking over heterogeneous models is possible.

In this chapter, the author introduced PCM-lite as a meta-model for addressing service description. In the following chapters PCM-lite will be adopt to represent in a unique format service properties extracted from heterogeneous sources in order to enable effective matchmaking by exploiting Semantic Web technologies.

[5]Definition available at `http://www.w3.org/Submission/WSMO-Lite/`

Chapter 5

Extraction of Explicit Property Values

The preliminary step of the matchmaking process that exploits Web descriptions is the extraction of property values that describe a specific service. As described in chapter 2, Web API information can be explicitly published over dispersed and heterogeneous Web sources or can be interpreted subjectively, such as popularity of a service provider.

Most of the explicit service descriptions can be textual, available as common Web pages based on XHTML, semi-structured data, such as XML and JSON documents. Moreover, Web sources can provide dynamic information, such as service availability, response time or user rating.

In this chapter, a novel approach is proposed for extracting explicit property values from Web documents by addressing source heterogeneity and dynamic information. The approach aims to propose a technique for building semantic descriptions from extracted data, in order to publish service information in a common, rich, machine-readable and interoperable format. Approach details will be provided after the description of techniques for construction of semantic descriptions from non-semantic data that are available in literature.

5.1 Extraction and annotation of Web descriptions: state of the art

The most popular system that performs information extraction with the aim of creation of semantic descriptions is DBpedia [76]. DBpedia is an information system that provides the semantic representation of Wikipedia contents. It exposes semantic descriptions as Linked Data and, to create them, it exploits Wikipedia database dumps. Each concept is represented by a page and the relations are built by exploiting hyperlinks defined in the Wikipedia infoboxes (typically shown on the right of wiki pages). The main lack of this system is the information currency. Sometimes, data updates depends on the frequency of the Wikipedia dumps (usually several months). Therefore, some documents can be out-of-date and the management of dynamic information is not addressed.

For addressing this issue, DBpedia Live has been proposed in [77]. This information system, adopt the same extraction implemented by DBpedia, but is able to provide up-to-date information as soon as a Wikipedia page is modified. DBpedia live does not exploit Wikipedia dumps, but information retrieved through the Wikipedia update stream.

The stream is accessible through the Wikipedia OAI-PMH live feed, a Web API that allows to pull updates in XML via HTTP. This system provides a good solution for managing dynamic information. However, this approach is applicable only on sources that provide update streams, for instances, RSS or Atom feeds.

An extended approach is adopted for building YAGO [78] ontology. YAGO is a semantic representation of Wikipedia and Wordnet [79], a large lexical database for the English language. The approach adopted is using sets of synonymous terms provided by Wordnet as classes of the ontology. Then, natural language processing techniques are used for associating instances identified in Wikipedia with Wordnet classes. This approach exploits Wikipedia dumps, as well as DBpedia, therefore the dynamic information management is not considered.

Instead, SOFIE [80] is a tool that extended existing ontologies by exploiting textual Web sources. Through natural language processing and automating reasoning, semantic concepts and relations are identified in the text and linked to an existing ontology. In the

the Web APIs scenario, this approach is not applicable because few semantic descriptions of existing services are available on the Web, as introduced in chapter 2.

ANGIE [81] is a tool that integrates a RDF database with missing information provided as XML by RESTful and SOAP-based services. The approach is based on mappings that links concepts defined in the RDF store with elements of XML data returned by the services. The management of dynamic values is supported by an on-the-fly invocation of service when users access to the database missing information. The limitation of the approach is to focus only to XML description without considering other semi-structured format (e.g., JSON) or textual description.

In the SOC research area, the first tool that aims to extract semantic description from non-semantic Web sources is Deimos [82]. On the base of a set of known sources, the tool is able to discover new services and automatically build their RDF/OWL representation. The approach requires samples of typical input and output parameters of services for a specific domain (e.g., weather services) and corresponding semantic description. Samples are used as feature of a machine learning algorithm that extract information from HTML documents. Deimos main limitations are supporting the extraction of only FPs and the definition of service descriptions samples. Moreover, experimentations in [82] proved hat approach is ineffective in some domains.

In [83], is proposed a technique for semantic annotation of RESTful services. The annotation is performed by exploiting DBpedia concepts. The input of the annotation process is a set of WADL documents. Labels of input and output parameters defined in WADL descriptions are matched with DBpedia concepts trough string similarity techniques. Then, matched concepts are associated with parameters. The main limitation of the approach is the focus on only geospatial services. Moreover, the semantic annotation is performed only on input and output parameters.

Karma [84] is a tool proposed for integrating RESTful services with Linking Open Data Cloud. The Karma approach is to provide RESTful resources defined according to XML or JSON as Linked Data. Karma extracts information from structured datasets in order to construct RDF representations. However, this approach does not consider that information about services can be dispersed over the Web and NFPs are not provided directly by services as resources. Moreover, most of the information on services is provided as textual descriptions in partially structured HTML documents. Finally, Karma does

not consider that Web documents can include information unrelated to services, which means that service properties should be extracted from portions of structured documents.

Finally, [85] is the only work that proposes an approach for extracting service information from RESTful service documentation on the best of author's knowledge. The method composes HTML structure analysis with natural language processing to deliver a complete automatic technique for extracting information, but it takes into account only service functionalities, neglecting NFPs.

5.2 Property value extraction from non-semantic data

Compared to the state of the art, this chapter proposes a novel approach for service property extraction and semantic annotation that consider the following aspects:

- extraction of functional and non-functional properties of Web APIs;
- supporting value extraction form non-semantic Web descriptions defined by heterogeneous models;
- management of dynamic properties, such as QoS or user rating.

The proposed technique supports property value extraction from semi-structured as well as textual descriptions. As introduced in chapter 2, Web API descriptions are published as semi-structured, such as JSON, XML or XML-based extensions (e.g., RSS or Atom), or textual descriptions as XHTML Web pages.

Despite Web pages appear complete unstructured in a Web browser, there is a hidden structure of the page that defines the page layout (e.g., paragraphs, titles, etc.) by XHTML standard tags. Therefore, textual descriptions can be considered as semi-structured data.

The approach proposed for the property value extraction is based on the adoption of *Source-to-policy templates* (S2PTs) that specify elements of a semi-structured documents that contain property values, then document portions identified through S2PTs are extracted. Afterwards, *named-entity recognition* techniques are adopted to identify semantic concepts in the document portions in order to create a semantic representation of the value. Finally, extracted semantic values are provided as PCM-lite descriptions.

5.2.1 Source-to-Policy Templates

A *Source-to-Policy Template* (S2PT for short) associated with a source declaratively specifies the properties that will be included in the extracted policies and, for each property, the process of extraction of property values. To extract service information a template defines several XPath[1] expressions. XPath is a W3C recommendation that allow the definition of paths that identifies portions of XML and HTML documents, that is also adaptable to JSON format.

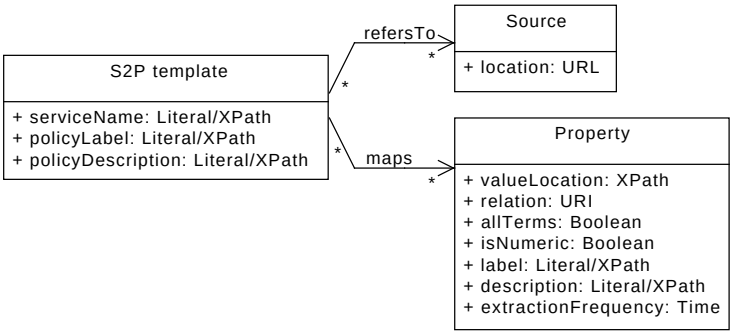

FIGURE 5.1: Formalization of Source-to-Policy Templates (S2PTs)

The S2PT formalization is represented as UML class diagram in figure 5.1. A template refers to one or more sources that are localized on the Web by an URL. Moreover, a template maps a set of properties that are represented by the following attributes:

- a XPath expression that identifies the value location in a document portion document;
- a URI that defines the relation between the property and the value according to PCM-lite (e.g., *hasValue* or *hasUnit*);
- boolean flag that specifies if the value is represented exactly by all text terms in the document portion or is defined only by a sub set of them, in order to apply different extraction techniques.
- a boolean flag that specifies if the property assumes a numeric or symbolic value.
- a natural language label that defines the property name (e.g., "data license" or "tags");

[1]Formalization available at:`http://www.w3.org/TR/xpath/`

- a natural language description of the property;
- a time interval that defines the extraction frequency, in order to support the dynamism of the information provided.

Finally, a S2PT has a service name, a label and a comment that describe the PCM-lite policy in natural language. Labels and comments for policies and properties can be defined manually as literal or extracted from documents through the definition of XPath expressions. Natural language attributes will be associated with PCM-lite policies and properties by using `rdfs:comment` and `rdfs:label` (according to RDFS vocabulary [61]), in order to increase the human readability of the extracted semantic descriptions.

According to the formalization in figure 5.1, S2PTs can be defined in a semi-structured data format, such as XML or JSON, or in RDF as an ontology. Currently, S2PTs are manually written by experts, anyway, due to the limited number of sources to be considered (e.g., API repositories and blogs) this should not be regarded as a critical limit of the approach. Once a significant set of templates will be available, it is reasonable to foresee a reuse activity to address (semi-)automatic definition of templates for new sources.

An example of a XML-based S2PT for ProgrammableWeb, that provides data as Atom feeds, is represented in the listing 5.1. The XML code shows in detail the extraction configuration for the value associate with the data format property.

5.2.2 Property value extraction process

Each time is required a value extraction, after the requested policy submission, for dynamic properties, or periodically, is executed the process shown in figure 5.2 as activity diagram according to UML 2.

The first phase is the *document retrieval*. By using the source location URL defined in the S2PT, the document that contains the the value is retrieved through a simple HTTP GET. Afterwards, the *document portion extraction* is performed by submitting the retrieved document and the XPath expression associated with the property to a XPath engine. The result returned by the engine is a textual portion of document that contains the property value. The third process step is the *named-entity recognition* (NER). NER

```
<?xml version="1.0" encoding="UTF-8"?>
<s2pt:template
   xmlns:s2pt="http://pcm.itis.disco.unimib.it/s2pt"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <s2pt:source>
      <s2pt:location>
         http://api.programmableweb.com/apis/
      </s2pt:location>
   </s2pt:source>
   <s2pt:serviceName>
      /feed/entry/content/api/name
   </s2pt:serviceName>
   <s2pt:policyLabel>
      ProgrammableWeb description
   </s2pt:policyLabel>
   <s2pt:policyDescription>
      /feed/entry/content/api/description
   </s2pt:policyDescription>
   <s2pt:Property>
      <s2pt:valueLocation>
         /feed/entry/content/api/dataFormats
      </s2pt:valueLocation>
      <s2pt:allTerms>false</s2pt:allTerms>
      <s2pt:isNumeric>false</s2pt:isNumeric>
      <s2pt:description>Data formats</s2pt:description>
      <s2pt:extractionFrequency>24 hours</s2pt:extractionFrequency>
   </s2pt:Property>
   <s2pt:Property>
      ...
   </s2pt:Property>
   ...
</s2pt:template>
```

Listing 5.1: An example of S2PT for ProgrammableWeb

tools are able to identify textual terms that refers to a specific semantic concept on a given

To give an example, a NER tool can recognize form the text "Extensible Markup Language" the concept `df:XML` defined in a data format taxonomy adopted as domain ontology. The domain ontology defines property values instances, classes and relations. Can be exploited *ad-hoc* ontologies defined by a domain expert or ontologies already available on the Web. Available ontologies can be specialized in a domain (e.g., GeoNames, a geolocalization ontology) or more general and cross-domain, such as DBpedia [76] and YAGO [78].

The NER is a popular issue of the *natural language processing* research area. To address this issue is not the aim of this thesis, therefore the NER is adopted by our approach through existing tools. For the concrete implementation of the extraction process, will be adopted DBbedia-spotlight [86] a NER tool that identifies concepts provided by DBpedia.

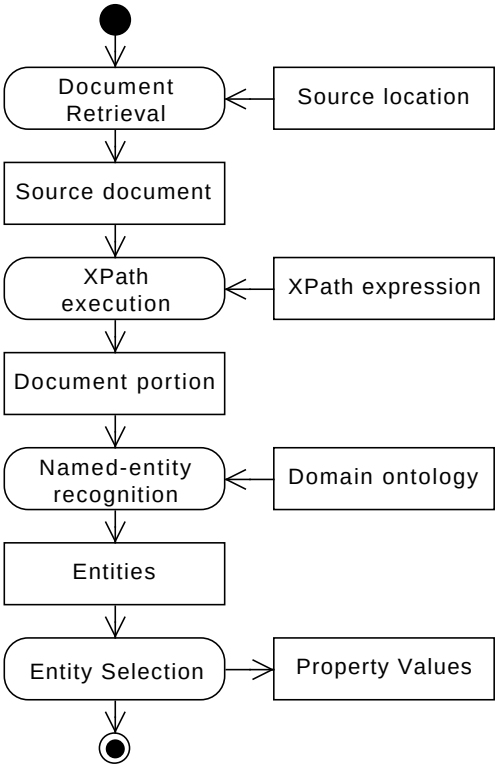

FIGURE 5.2: Property value extraction process from textual and semi-structured data

Details on the implementation will be provided in chapter 8. Instead, details on NER techniques are available in [87].

Finally, an *entity selection* is required to define which concept identified by NER refers to a property value. In a text portion that specify a value, can be identified concepts that are not related to the property. To give an example, in the text "You are expected to be able to parse the XML or JSON response payloads" the NER can identify the concepts `payload` and `parsing` that do not represent a value of the property "data formats". The techniques for the selection of entities depends of the type of property value tu be extracted. For properties that assume symbolic values, two different techniques are adopted. If the extracted text represents exactly the value, as defined by the `allTerms` S2PT flag, all the identified concepts became a property value. To give an example, form the text "music, streaming, social", can be recognized the concepts `music`, `streaming`, `social`. All three concepts will be grouped in a set that represents a value (e.g., for the property "tags").

Instead, the technique adopted for entity selection in text contains noisy terms is more complex. The preliminary step is processing the property description provided by the S2PT with the NER tool. The technique adopted is to perform NER on the property description, defined in the S2PT, and the text portion that contains the value. Then, the concept that will be considered a value is the entity in the extracted text that has the lowest *semantic distance* (under a particular threshold) with the concepts identified by the property description. For semantic distance, the lowest number of relations between not equivalent concepts defined by the domain ontology graph is considered. For instance, by referring to the previous example, `XML` and `JSON` are selected because their distance between the concept `DataFormat` identified in the property description "data formats" is lower to the other identified concepts.

Despite the extraction of numeric values is easier compared to symbolic values, units of measurement identification must be addressed for numeric properties. The author proposed to adopt the same technique for extracting symbolic values. To give an example, let consider "response time" as a property description and "the service response time is 5 ms". Moreover, let suppose that NER processing identifies respectively the concept `time`, for the property description, and `service`, `time` and `millisecond`. In the domain ontology, `time` has semantic distances 24 with `service`, 0 with `time`, because is the same concept, and 2 with `millisecond`. In this scenario, the concept selected to represent the unit is `millisecond` because has not equivalence relations with `time` and lowest semantic distance. When the extraction process is terminated, the extracted value is associated with a property instance of a policy according to the S2PT.

5.2.3 Policy creation

The creation of policies compliant to PCM-lite is performed at system start-up and periodically, in order to support potential service name modifications. The approach adopted for creating polices is modelled as the UML 2 activity diagram shown in figure 5.3. For each URL that refers to a service description in the S2PT, a policy instance is created. Each policy is labelled with the service name, policy label and the policy descriptions that are extracted by executing XPath expressions, defined in the S2PT, on the retrieved service description. After that, for each property defined in the S2P template, a property instance and an *extraction daemon* is created. Extraction daemons

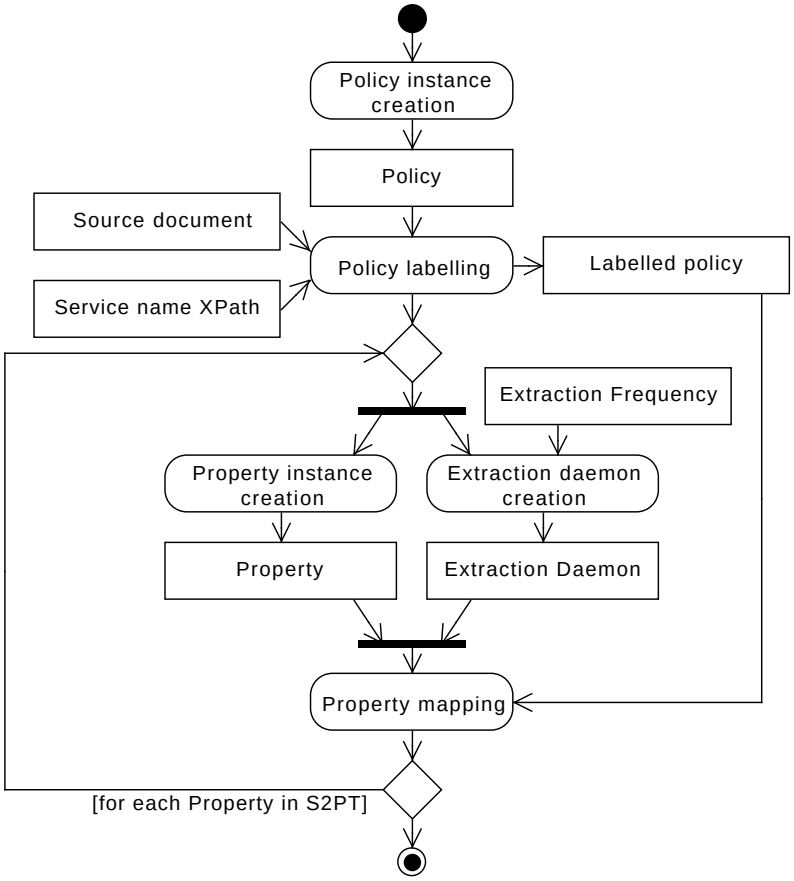

FIGURE 5.3: Property value extraction process from textual and semi-structured data

are software components that perform the extraction process and manage the value update. Each time a property value is required, the associated daemon retrieves a cached value, updated according the extraction frequency defined in the S2PT. Afterwards, each property instance is associated with the extraction daemon and the related policy.

In order to implement a real matchmaking tool, the approaches described in this chapter will be implemented by *wrappers*. A wrapper is a software component that performs property value extraction and publish PCM-lite description according to Linked Data principles. By adopting this strategy, the aim is to extend the Linked Open Data Cloud in order to promote the diffusion of semantic service descriptions and wrappers. Details on wrappers are available in chapter 8. Next chapter focus on extraction of properties that are evaluated subjectively by users.

Chapter 6

Extraction of Subjective Property Values

The survey described in chapter 2 has shown that relevant properties are interpreted subjectively by users through personal experience or collecting casual information available on the Web. Properties that can be considered "subjective" and relevant for users are popularity of a service provider, quality of the answers provided by Web API forums for supporting developers, quality of the Web API documentation. As a matter of facts, the Web does not provide explicit information about these characteristics, and it is marginally considered by the literature.

For addressing this issue, a property extraction approach based on social media analysis is proposed. According to the definition provided in [88], social media is a group of Internet-based application that are built following the Web 2.0 principles to create and exchange user-generated content. This group is manly composed of social networks (e.g., Facebook and Twitter), blogs, wikis and on-line forums. These Web sources can provide information about any existing topic, public figure, product or service.

The automatic analysis of communities behind social media allows for a global view on information that would be unreachable by manual evaluations, which are time consuming and reflects personal viewpoints. Therefore, a tool for social media analysis permits users to evaluate information with a unified prospective, that allows comparisons.

The aim of this chapter is to define techniques for value extraction of *social properties*. The concept "social property" is defined as follows.

Definition 6.1 (Social property)**.** *A social property is a subjective property that can be estimated through social media analysis.*

Since social media has completely different characteristics, to extract a specific property value an *ad-hoc* technique must be developed for each source. To give an example, the evaluation of provider popularity can not be performed in the same way on social networks and blogs, because their aims are different, then the information is provided in different ways. Typically, blogs provide long articles that can be commented. Instead, users on social networks write short posts that can be commented and re-shared by other users. Moreover, posts can be approved or disapproved (e.g., Facebook "likes") and users can follow profiles of public figures, companies or products according their interests.

In this chapter, specific techniques for evaluation of three properties are proposed as a starting point for addressing property value extraction from social media. The considered properties are: (i) provider popularity, (ii) vitality of Web API forums and (iii) Web API usage. After discussing a state-of-the-art of evaluation techniques for subjective properties, the following aspects are addressed:

- how the three properties can be extracted;
- a demonstration that the properties are correlated;
- then, the correlation can be exploited for estimating properties that are not provided by social media.

6.1 Subjective property evaluation: state of the art

Social media analysis is an issue of the *Social Computing*, a research area that broadly covers computational facilitation of social studies and human social dynamics, as well as design and use of information and communication technologies that consider social context [89]. Despite social media analysis is recently adopted in industry for business intelligence [88], this instrument is considered only in few works in the SOC research area.

On the best of author's knowledge, only [52, 53] propose Web API discovery tools that exploits service information provided by existing social media. ApiHut [52] is a Web API search engine that exploits information provided by ProgrammableWeb. As introduced in chapter 2, ProgrammableWeb is a portal for Web APIs and allows users to publish service and mashup descriptions. Moreover, users can specify a rating for each API and mashup. According to the definition provided by [88], ProgrammableWeb can be considered a social media because it follow Web 2.0 principles and permits users to publish user-generated contents. ApiHut performs a Web API keyword-based search based FPs (tags) and a set of NFPs by using information retrieval techniques. After that, it ranks APIs on the base of a social property: the popularity of the service. The popularity evaluation is measured exploiting Alexa[1] and mashups information.

Alexa is a service that provides a ranking of Web sites based on the network traffic. This service is exploited by ApiHut for evaluating the ranking of each mashup available on ProgrammableWeb. The ranking is combined with mashup user rating, then it is assigned to each API that compose the mashup. Finally, a weighted average of these parameters is computed to produce a service usage index that is considered as service popularity metric.

MashupReco [53] has been proposed as alternative of ApiHut. It computes the service usage through an API graph built exploiting mashups information. Each graph edge represents a Web API. APIs that are co-used to build mashups are linked by vertices. The number of links are used for evaluating the service usage. The more links connect an API, the more it is used, therefore is more popular.

The two proposed approaches are interesting but a very important aspect is not considered: the temporal information. With the flowing of time, services evolve by proposing new functionalities or can fall into disuse for the coming of new services. Without considering temporal information, an API that was very popular in the past could be evaluated popular also today. To give an example, Myspace was the most popular social network between 2006 and 2008. After the coming of Facebook, most of Myspace users moved to the new social network [90]. The result is that Myspace company is going to failing because of few users. In this scenario, ApiHut and MashupReco evaluate the same popularity of Myspace and Facebook APIs because these tools exploit all the existing

[1] Available at: `http://www.alexa.com/`

mashup descriptions published from 2005, despite the publication date of each mashups is available on ProgrammableWeb. In order to figure out correctly service popularity and usage, the tools must exploit the information published in the last time period (e.g., last year).

Another approach for computing service popularity is proposed in [91]. The technique adopted do not exploit social media, but monitors mashups and service invocations. The principle on the base of this work is that the more a service is invoked, the more is popular. The limitation of this approach is that can be applied only if the internal logic of each monitored mashup is known, since mashups contain several service invocations and the number of invocation can depend on the input and output of each service. For instance, let consider a mashup that implements the localization on a map of a set of music events. Music events are provided by Last.fm APIs, instead Google Maps APIs are used as a geolocalization service. For each mashup invocation, future music events are retrieved, then the geolocalization service is invoked for each retrieved event. In this example, the number of invocations to Google Maps depends on the number of events provided by Last.fm, which can change periodically. Therefore, if provided events decrease, the geolocalization service invocations decrease, but this does not imply that the Google Maps popularity is declined. Finally, the approach on large scale is unfeasible because to develop *ad-hoc* monitors for thousands of mashups and APIs available on the Web is unrealistic.

In [92], a Web API recommendation system for mashup building, called iSocialMash is proposed. The tool allows users to publish mashups descriptions and establishes social relationships among Web developers. With its characteristics, iSocialMash can be considered a social media. Social relationships are exploited for the assignment of tags that identifies each mashup or API functionalities according the following approach. Each user defines a set of tags that represent the goal for which a published mashup is developed. For instance, the tags “geolocalization”, “music” and “events” can be associated with the mashup in the example above. Tags aggregated to identify the most representative keywords according to social relations between users that collaborate for developing mashups. To give an example, tags defined by two users that, respectively, one is expert of Last.fm API and the other is expert of Google Maps, and collaborate each other for developing the above mashup are more relevant, compared to keyword associated by other users. Despite the proposed approach is promising, currently, iSocialMash is

not available on the Web. Therefore, there are not experimentations that prove the effectiveness on a real set of users that specify goals and collaborate each other.

6.2 Social property evaluation

This section provides techniques for evaluating *provider popularity*, *vitality of Web API forums* and *service usage*. According to the results of the survey described in chapter 2, the first two properties are very relevant for Web API users. Instead, the last one is the most considered in the SOC literature and sometimes is used as an estimator of the service popularity [52, 53, 91]. Provider popularity is a relevant characteristic that users exploits for choosing between similar services. In users opinion, a popular provider (e.g., Google) typically guarantee a reliable service. In addition, providers make available forums in order to support and help developers that use Web APIs. Users consider that support forums with a high community vitality are able to provide timely and sound solutions for development issues. Finally, service usage evaluation is addressed in order to provide more effective solutions compared with the literature.

6.2.1 Evaluation of provider popularity

As well know, Google is the most used a popular Web search engine. Through the Alexa[2] service, that monitors network traffic of Web sites, it is proved that approximately the 60% of access to search engines are on Google on April, 2013. For evaluating provider popularity, Google Trends[3] can be exploited. This Google service provides information about queries submitted to the Web search engine. Google Trends computes the weekly volume of searches related to every keyword since 2004 and provides this data as CSV (comma-separated values) documents. Therefore, the search volume associated with a provider name can be considered a popularity index, because it represents how much a provider is searched on the Web. Despite, Google is a Web search engine, for some aspects can be considered a social media. The submitted queries can be considered user-generated content which related search volumes provided by Google Trends can be shared between users.

[2]Available at: `http://www.alexa.com/`
[3]Available at: `http://www.google.com/trends/`

Through Google Trends, the popularity Π of a service provider p is computed as follows:

$$\Pi(p) = 1 - e^{-\mu_v^2}, \text{ where}$$

$$\mu_v = \sum_{i=0}^{n} \frac{v_i(p)}{n} \text{ and}$$

$v_i(p)$ is the search volume of the provider p during the i-th week before the current one. The formula is based on an average (μ_v) of search volumes of the n last weeks. Then, $\Pi(p)$ is computed as a normalized in range $[0, 1]$ of μ_v through a Gaussian function.

The reason of the average computation is smoothing periodical or exceptional peaks caused by particular events that may affect Google searches. To give an example, Amazon, the popular e-commerce website, is also a provider of cloud computing services. The search volume graph (in figure 6.1) shows that in December there is a high increase of searches due to Christmas gifts phenomenon. On a sample of 6178 providers, that published APIs on ProgrammableWeb, it has been detected that the 5.3% of search volumes present periodical peaks and all of them have frequency lower or equal to one per year. Therefore, the author suggest to compute the average of the search volume on a year base ($n = 52$) in order to reduce the noise generated by such exceptional or periodical peaks.

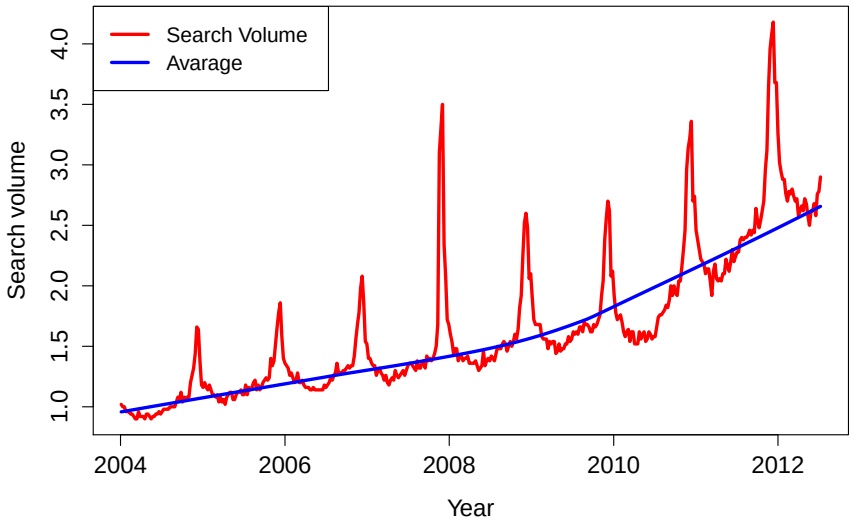

FIGURE 6.1: Amazon search volume provided by Google trends

6.2.2 Evaluation of Web API forum vitality

In this thesis, the forum vitality is defined as the volume of people that are active by contributing to the generation of content in a forum. In order to evaluate this property is necessary accessing to all the threads and posts and to be able to identify user information. A general tool that is able to retrieve this information is not possible to be modelled, because each forum disposes the required information according to different page layouts. Therefore the only solution is to develop a *ad-hoc* Web scraper for each forum.

In order to collect a representative sample of forum information, a scraper was developed for Google Groups. Google Groups is a Web platform that allows end-users to publish forums and has a default layout. The scraper supported the access to forums about 110 Web APIs to collect information about 49078 users, 532514 posts and 68815 threads.

Through Google groups data, the vitality V of a Web API forum f is computed as follows:

$$V(f) = 1 - e^{-\mu_U^2}, \text{ where}$$

$$\mu_U = \sum_{i=0}^{n} \frac{|U_i|}{n} \text{ and}$$

U_i is the set of users that posted at least one message during the i-th day before the current one. The vitality evaluation is based on the average μ_U evaluated on the number of daily active users. Then, $V(f)$ is μ_U normalized through a Gaussian function to provide a value in range $[0, 1]$, as well as provider popularity. Every day the number of active users can noticeably vary in forums. For instance, proposing topic of common interest can increase exceptionally active users for a day. As well as for provider popularity, the average is computed to smooth out the noise generated by expectational events. The author suggests to compute the average on a temporal window of one year before the current date ($n = 365$).

6.2.3 Evaluation of service usage

Sometimes, in literature, service usage is a property used as an index of service popularity. As introduced in section 6.1, the main limitation of the two state-of-the-art approaches to evaluate this property [52, 53] do not consider temporal information.

The social media exploited is ProgrammableWeb by considering mashup descriptions published by users. The estimation of usage SU of a service (s) is evaluated as follows:

$$SU(s) = 1 - e^{-\mu_{M_a}^2}, \text{ where}$$

$$\mu_{M_a} = \sum_{i=0}^{n} \frac{|M_{i,a}|}{n}.$$

μ_{M_a} represents the average number of elements in the set $M_{i,a}$ composed of mashups that invoke the API a and have been published on ProgrammableWeb the i-th day before today. Trough a Gaussian function, μ_{M_a} is normalized to provide a value in the interval [0,1]. In addition, to reduce potential noise, the average is computed in order to figure out the usage in the last period. This approach overcome the limitation of [52, 53] by considering recent information. The author suggests to consider information provided in the past year for conformity to the evaluation of the other two subjective properties.

6.3 Social properties correlations

According to the survey results provided in chapter 2, users prefer services with popular providers. Therefore, a correlation between provider popularity and service usage should exist. Moreover, it is straightforward to think that more an API is used, more its forum community is active. Obviously, the proportion depend on the API usability. A complex API should have a higher forum vitality compared to a simple one. But in general, if the usage increase, the number of potential users that need help through forums should increase too. Hence, a correlation should also exist between service usage and web API forum activity. Finally, if provider popularity and forum vitality are correlated with usage, both of them should be correlated each other for transitivity.

The aim of this section is to verify that the three social properties are correlated each other through an experimentation. If this hypothesis is true, the correlations can be exploited for estimating social property values that Web sources do not provide. This hypothesis of correlation between provider popularity, service usage and Web API forum vitality is confirmed by the following experimentation.

The experimentation dataset is composed of information about 110 Web APIs, which Google Trends, Google Groups and ProgrammableWeb provide data for estimating the

three social properties according the approaches shown in the section above. For each service, the social properties have been computed for each time period from January 2004 to May 2012.

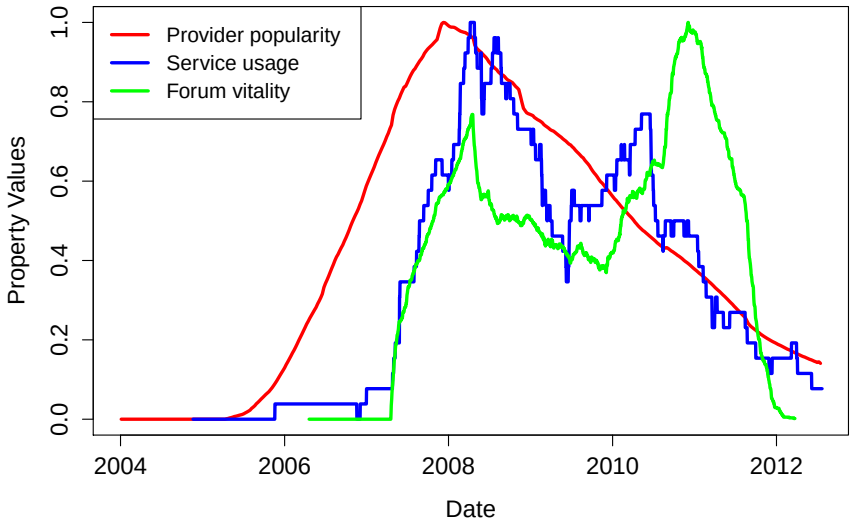

FIGURE 6.2: Provider popularity, vitality of forum and usage of Digg APIs

Figure 6.2 shows the social property values for Digg APIs. Digg[4] is a social news website which its APIs has been published at the end of 2005, then removed at begin of 2012. Therefore, the graph shows the complete life of the service. In general, the three properties increase from end of 2005 until the begin of 2008, then decrease until the end of service life, despite some peaks. Peaks can be justified as follows. From the end of 2009, Digg deployed a new version of APIs with additional functionalities, then increased the number of mashups. The forum vitality is increased twice compared to usage because forum mediators have been addressed problems of old Web developer for upgrading APIs, in addition to issues of new users. Despite the release of a new API version, Digg popularity decreased until 2012, when the service has been retired.

This analysis showed that the three social properties have the same behaviour for Digg APIs, for exception of the time period in which a new API version has been released. In order to automatise the experimentation, two correlation coefficients can be computed: Pearson's and Spearman's correlations. Pearson's coefficient [93] evaluates a linear correlation between two distributions of values. The evaluation of the Pearson's

[4]Available at: `http://digg.com/`

coefficient return a value in the interval $[-1, 1]$. More the coefficient is near 1, more the values are correlated according an increasing linear function. Instead, more the coefficient is close to -1, more the values are correlated according a decreasing linear function.

In the example provided in figure 6.2, the Pearson's coefficient is 0.89 for provider popularity and forum vitality during the 2007, because property values proportionally increase. Instead, the correlation of the same properties is -0.79 during the 2011, which means that values proportionally decrease.

Spearman's correlation evaluation [94] returns a value in the interval $[-1, 1]$ that specifies if a distribution of values is correlated according a linear or non-linear function, such as exponential, logarithm or polynomial. As well as for Pearson's correlation, a coefficient value close to 1 means that the correlation function is increasing, instead a value near -1 specifies a decreasing behaviour. According to literature, two distributions can be considered correlated if one of the both evaluations return a coefficient which its absolute value is grater than 0.75.

To prove that these correlations exist between properties for the 110 service of the experimentation sample, the approach described by algorithm 1 is adopted. The algorithm

Algorithm 1 CorrelationEvaluation (A, B, D, δ)

```
1: cp ← 0
2: for all date ∈ D do
3:     subA ← getValuesOfPeriod(A, date, δ);
4:     subB ← getValuesOfPeriod(B, date, δ);
5:     c ← computeCorrelation(subA, subB);
6:     if |c| > 0.75 then
7:         cp ← cp + 1;
8:     end if
9: end for
10: return cp.
```

takes two sets of property values A and B, a set of dates D and a number of days δ as inputs. A and B represent values of two social properties associated with a service which their correlation must be computed. The set D is composed by the dates of the period which the properties are evaluated (from January 1st, 2004 to May 31st, 2012). Finally, δ represents the time period size of value samples which the correlation is computed. The algorithm return a number periods cp in which the two properties are correlated. In line 1, cp is initialized. Then, for each $date$ in D is created a subset $subA$ of values

in A and a subset $subB$ of values in B (lines 2-4). The subsets are created through the function $getValuesOfPeriod$ that returns the values measured in the temporal interval $[date, date + \delta]$. Line 5 states that the correlation coefficient c is computed for values in $subA$ and $subB$. If $|c| > 0.75$ the values of the period are correlated then, cp is incremented. Finally, cp is returned. This approach based on evaluation subset of values allows for inferring the percentage of service life in which the social properties are correlated.

For intervals of three months ($\delta = 90$), the evaluation of 110 services defined by algorithm 1 provides the results available in table 6.1. The provided results show the percentage of correlated time period between property couples. The experimentation highlights that the most correlated properties are forum vitality and service usage. The reason is that provider popularity is the lesser related property with services, because can be easily affected by external events. To give an example, Google Glasses, devices for augmented reality, are very discussed in these months, therefore the Google popularity is increasing, but this phenomenon does not affect usage of Google Web APIs. A higher percentage of periods correlated with Spearman's correlations is attended because Spearman's coefficient includes identifies both linear and non-linear correlations.

Properties couples	Pearson's coefficient	Spearman's coefficient
Provider popularity - Forum vitality	55.6%	62.3%
Forum vitality - Service usage	76.8%	85.3%
Service usage - Provider popularity	63.2%	71.2%

TABLE 6.1: Pearson's and Spearman's correlations between social properties

6.4 Estimation of missing property values

Web sources can become temporary unavailable, removed or restructured. In this scenario, the correlations between social properties can be useful for estimating missing values. To give an example, Foursquare API forum has been moved from Google Groups to another website on November 2011. The new forum changed the layout therefore maintaining the evaluation of its vitality requires the redesign of the Web scraper used for collecting data. However, the evaluation of the Pearson's correlation between vitality of the old forum and Foursquare popularity proves that a strong linear correlation exists between

the two properties (the coefficient is 0.945). Therefore, the provide popularity can be exploited for estimating the vitality of the new forum.

The objective of this section is to provide a generic approach for estimating missing value of social properties. The approach aims to find a function f such that:

$$f(\upsilon_a) = \upsilon_m$$

where, υ_a is a property value available and υ_m is a missing value of another property. The proposed approach as two requirements: (i) the availability of past values of the property to be estimated (e.g., vitality of the old forum) and (ii) a correlation of past values with other property values.

The first phase of the approach is to identify the last time interval ι in which past values of the missing property have been correlated with the property used as estimator. For instance, the last six months before the Foursquare API forum have been moved. Then, if the property correlation is linear, according Pearson's coefficient, it implies that:

$$\upsilon_m = f(\upsilon_a) = \alpha + \beta\upsilon_a$$

According to the literature [95], the constant values α and β can be estimated through a 2-dimensional regression model based on linear least squares method. The method computes α and β as follows:

$$\beta = \frac{cov(\Upsilon_a, \Upsilon_p)}{var(\Upsilon_a)} \text{ and } \alpha = \bar{\upsilon_p} - \beta\bar{\upsilon_a}$$

where Υ_a and Υ_p are, respectively, the set of property values available and the set of past values of the property to be estimated in the time interval ι. Instead, $\bar{\upsilon_p}$ and $\bar{\upsilon_a}$ are the simple means of, respectively Υ_a and Υ_p.

For non-linear correlations, the estimation cannot be easily computed. The literature proposes the application of the Gauss-Newton algorithm for estimating a non-linear regression model [96]. The algorithm requires a model function that can be represented as:

$$\upsilon_m = f(\upsilon_a, \beta)$$

where β is a vector of variables $\beta_0, ..., \beta_n$. To give an example, if the non-linear function that defines the relation between two properties is a hyperbola, the model function is:

$$v_m = \beta_0 + \beta_1 v_a + \beta_2 v_a^2.$$

However, this algorithm is not applicable for estimating service properties because asserting a model function of future missing values is unpredictable.

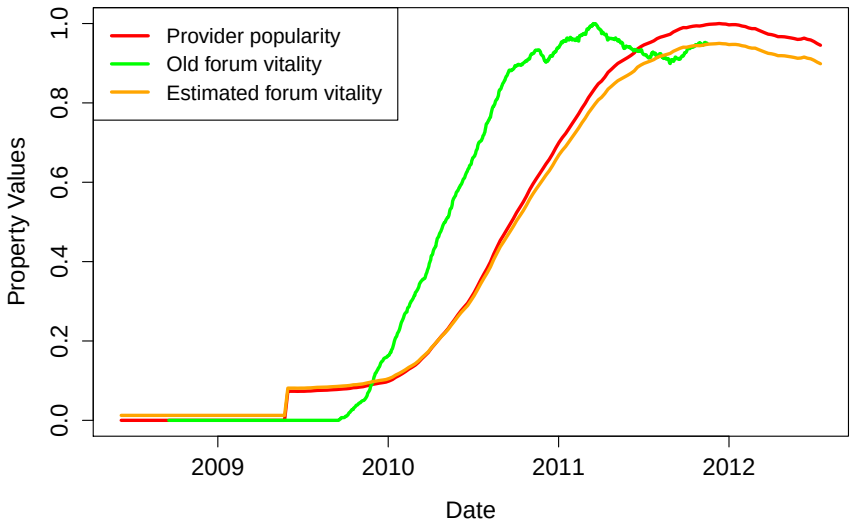

FIGURE 6.3: Estimation of forum vitality for Foursquare APIs

Figure 6.3 shows the estimation of the vitality of the Foursquare API forum. Vitality estimation is possible for this service because the correlation with provider popularity is linear, theretofore linear least squares method is adopted.

This chapter proposes a preliminary work on the evaluation on subjective properties, therefore the proposed approach presents some limitations and open issues. A general approach for addressing this issue must be provided, moreover a more advanced study on estimation of missing property values is necessary. However, some relevant results are obtained. Temporal aspects must be considered because user perception of properties can change during the service life. In addition, a relation between subjective property values can exist.

In previous and this chapters, techniques for extracting property values from Web sources are provided. Next chapter will provide techniques for evaluating the quality of extracted

values, then techniques for fusing descriptions and performing matchmaking driven by the quality assessments.

Chapter 7

Quality-driven Fusion and Matchmaking

Service information provided by different Web sources can not be considered at the same level. Web descriptions can be out-of-date or published by untrustworthy sources. Moreover, Web data can be inaccurate (e.g., can be contain orthographic errors) and NER techniques, used for the extraction of property values (chapter 5), are not always effective [87]. The result can be a wrong extracted values that affect the effectiveness of the service matchmaking.

Another aspect that must be considered is the availability of Web descriptions that provide information about same services and are dispersed over the Web. Moreover, the same properties of a specific service can be published providing potential contradictory values.

In this scenario, an effective evaluation of extracted information quality can be exploited for: (i) fusing extracted descriptions of same services in order to provide a unique and rich description for each Web API; (ii) driving the matchmaking form providing more effective results to users.

In this chapter, techniques for quality assessments, description fusion and matchmaking are provided. After a state-of-the-art analysis, techniques for quality evaluation of service descriptions available on Web sources are described. Then, the quality-driven descriptions

fusion techniques are explained. Finally, the quality-driven semantic matchmaking between fused descriptions and requester constraints is shown.

7.1 Quality assessments of properties available on the Web

The evaluation of quality dimensions to support integration of data from different sources is mainly addressed in the database research field. Several proposals exist in the area of virtual data integration architectures for quality-driven query processing, which returns an answer to a global query, by explicitly taking into account the quality of data provided by local sources. All approaches consider several quality meta-data associated with sources and the user query, which support the query processing activity. These approaches do not consider semantic descriptions and presume the knowledge of data schemas to permit the integration of several sources. A comparative description and analysis of proposals appears in [97].

The usage of quality dimensions to guide the data source selection process is addressed in [98]. The proposed approach is based on the evaluation of Web source reputation as a composite metric that considers trustworthiness and relevance of a source. Reputation is considered as a multi-dimensional quality attribute defined by extending fundamental data quality dimensions (i.e. accuracy, completeness, and time) defined in [99] with additional dimensions (i.e. interpretability, authority, and dependability) that should be considered when assessing reputation, especially for semi-structured and non-structured sources of information. This approach is applicable and effective for blogs and forums, because the source reputation is evaluated through the analysis of comments and posts provided by users. Most of the sources that provide Web API descriptions does not allow users to add comments, therefore the proposed approach is marginally applicable in the service domain.

Several papers deal with the trustworthiness of Web data and Web sources. A data provenance model which estimates the level of trustworthiness of both data and data providers by assigning trust scores to them is defined in [100]. Based on the trust scores, users can make more informed decisions on whether to use the data or not. The work is based on the hypothesis that if same information is provided by a lot of sources, the sources are trustworthy. In the Web APIs scenario, this assumption is wrong because

service information can change frequently during an API life, therefore, if most of the sources publish the same property values, but out-of-date, the information provided must be considered untrustworthy.

In the SOC research area, the evaluation of the quality of service descriptions is only on trustworthiness of QoS information published by service providers. This issue is addressed by several theorical approaches [101–105] and tools [54, 55] proposed in literature. These works adopt one of the following strategies: (i) to develop and promote the diffusion of trusted services that monitors QoS parameters, then to compare monitoring results with QoS guaranteed by providers [101, 104]; (ii) evaluation of user feedbacks and ratings about services [54, 103]; (iii) a composition of the previous two strategies [55, 102, 105]. These proposed approaches have several limitations for the current service scenario promoted by Web APIs.

First of all, only ProgrammableWeb and API-status performs QoS monitoring on approximately 40 services (details available in chapter 2). In addition, only QoS can be monitored unlike other properties such as functional descriptions, data licensing or usage fees. The evaluation of user feedbacks is a better solution to measure the trustworthiness of service descriptions, but it does not consider two aspects. The first one is that submitting comments or ratings requires an active user interaction with the system. Some users can consider this activity a waste of time, therefore the collected feedbacks could be an unrepresentative sample of opinions. The second aspect is that user opinion can change after a time period, hence old feedbacks can be unreliable. To conclude, proposed approaches, designed for classical SOAP-based services, are inadequate for real services that publish public functionalities on the Web.

7.1.1 Definition of quality assessments for service descriptions

On the base of the state-of-the-art analysis, specific quality evaluation techniques for Web sources that provide service information are proposed. Accuracy, currency and trustworthiness are the quality measures considered in this thesis. These three metrics are the most significant among the several dimensions investigated in the literature for quality assessments of information retrieved from the Web [99]. For each dimension short description and the chosen metric are provided.

Accuracy

Accuracy of a value v is defined as the closeness of v to a reference value $v*$, considered as the correct representation of the phenomenon that v aims to represent [97].

In this thesis, the accuracy is a dimension associated with each property value. Since values of numeric property values are extracted from the original sources without additional processing, values are assumed to be accurate for these properties. Instead, symbolic values undergo a process that transforms data represented in free text into ontology instances exploiting a named-entity recognizer (as shown in chapter 5). Therefore, the accuracy can be affected by both document errors and effectiveness of extraction techniques.

The proposed metric to compute accuracy is therefore based on a similarity score that is associated with every entity extracted by the adopted named-entity recognizer. Let $sm(w)$ be the similarity score associated with an entity extracted from the word w by the adopted recognizer; $sm(w)$ is computed making use of the frequency of w in the text (f_w) and the Inverse Candidate Frequency (ICF) of w according to the following function (details on the similarity measure can be found in [86]):

$$sm(w) = f_w \cdot ICF(w),$$

The similarity score is able to detect how much the extracted term is related to the text context. Based on this similarity score, the *accuracy* a_v of a property value v is computed by the following formula:

$$a_v = \begin{cases} 1 & \text{if } v \text{ is numeric} \\ \frac{1}{|EW|} \sum_{w \in EW} sm(w) & \text{if } v \text{ is symbolic} \end{cases}$$

where EW is the set of words from which entities have been extracted. The accuracy evaluation returns a value in range $[0, 1]$ that more it is higher, more the extracted value v is accurate.

To better understand how the approach works, the following examples are shown. Let the following text portion extracted from the Musicbrainz[1] API documentation:

[1] Available at: `http://musicbrainz.org/doc/About/Data_License`

> "The core data of the database is licensed under the Public Domain. This means that anyone can download and use the core data in any way they see fit. No restrictions, no worries! The remaining portions of the database are released under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 license. This allows for non-commercial use of the data as long as MusicBrainz is given credit and that derivative works (works based on the Creative Commons licensed data) are also made available under the same license."

The text provides a data licence property that assume "Public Domain" and "Creative Commons license" as values. According to the formulas above, similarity scores for 'Public Domain" and "Creative Commons license" are, respectively, 0.251 and 0.334, then the accuracy is 0.293. Let consider that our extraction process wrongly extract the "database" as an additional third value. The similarity score of the wrong value is 0.175, then the accuracy is 0.253, that is lower than the correct value extraction. The "database" value is the most out of context compared with 'Public Domain" and "Creative Commons license", therefore it assume the lowest similarity score, that decrease the accuracy.

Currency

Currency refers to the promptness of an API description published in the Web source to reflect a change of the values of a property occurred in the real world. Since changes are typically performed at description level, the currency is assessed at that level of granularity, and afterwards inherited by properties extracted from the descriptions. Let d be a source document and v a property value extracted from d; the currency for v is defined by the following formula:

$$c_v = c_d = e^{-t^2},$$

where t is a time difference computed by the difference of days between the current date and the publication date in the source document d. The result is a value in range $[0, 1]$ in which 1 represent a current information. Despite the proposed formula assigns a low *currency* value to old documents that could contain correct information, the proposed metrics is suitable in this domain because APIs change frequently and the documents

describing them should be up to date. Often, say, one-year-old descriptions are assumed to be out of date.

Publication date is not always available on Web documents, but descriptions provided by Web API repositories, such as ProgrammableWeb, or wikis contain this information. If available, publication date can be easily extracted from XML, XHTML and JSON descriptions by using XPath.

Trustworthiness

Trustworthiness of information in the Web is hardly evaluated on atomic data, such as single service properties, since their origin and provenance is frequently unknown; as a consequence, trustworthiness of data is indirectly evaluated from trustworthiness of sources. Since most of the sources have public profiles (*source profile* in the following) on social networks (e.g. ProgrammableWeb has a public page on Facebook and a Twitter profile) can be followed by several users, the trustworthiness evaluation of a source s is performed by analysing the activity of social-network users with public pages associated with s. Examples of activities that can be considered are placing a "like" to a post of a source profile on Facebook, and "retweetting" a tweet of a source profile on Twitter. In order to enable this evaluation, a specific social wrapper must be designed, developed and deployed to actually extract information on activities related to specific sources from each social network.

The activity rate ar of a source s on a social network sn is defined by the number of the followers of sn that have been active in a recent time window t, divided by the total number of followers of s in sn. Therefore, the activity rate $ar(s, sn)$ is computed as follows:

$$ar(s, sn) = \frac{|activeFollowers(s, sn, t)|}{|followers(s, sn)|}$$

The trustworthiness of a source can be defined by aggregating the active rates on several social networks. Given a set SN of social networks, the trustworthiness t_s of a source s is defined as follows:

$$t_s = 1 - e^{-\left[\sum_{sn \in SN} ar(s,sn)\right]^2}$$

Observe that we sum the active rates, since the trustworthiness of a source increases as the source is actively followed on more social networks; anyway, the trustworthiness is normalized in the range [0,1] through a Gaussian function to provide a more readable and understandable value. Therefore, in a property value v extracted from a source s, the trustworthiness score t_v is assigned the trustworthiness t_s:

$$t_v = t_s$$

As introduced previously, the most feasible approaches proposed in literature evaluate the trustworthiness by exploiting user ratings. The author choose to perform the evaluation through social networks for two reasons. The first one is that users perform a forced activity when rate a source. On one hand, if the system do not force the rating, users can consider it a waste of time, therefore the result is a rating based on an unrepresentative sample of users. On the other hand, if the rating submission is forced, the user iteration is more complex with possible recoil to the tool usability. Instead, users follow intentionally information provided by Web sources through social networks that they consider trustworthy according to their interest.

The latter reason is that ratings represent a user evaluation in a specific time period. Users can change opinion along the time and this information is aspect is not supported by rating systems. Instead, on social networks, users become inactive or stop to follow a source if change opinion about information provided.

The aggregated quality measure

In order to figure out an overall evaluation of information quality, an aggregated quality measure q_v associated with a property value v is defined as the following weighted sum:

$$q_v = w_a a_v + w_c c_v + w_t t_v, \text{ where } w_a + w_c + w_t = 1 \text{ and } w_a > w_c > w_t \ .$$

The inequality constraints on the weights reflect the different granularity levels (value, document and source) at which the quality dimensions are assessed. The idea is that the more fine grained the method to measure a quality is, the higher the contribution of the quality to the aggregated measure should be; in other words, the aggregated quality of extracted property value v depends more strongly on its accuracy than on its currency and trustworthiness, which are qualities originally associated with the document and the source respectively from which the value profile has been extracted. In chapter 9, experimental results will confirm that this hypothesis improves the effectiveness of the fusion method. The weights w_a, w_c and w_t are established by experts of Web API domains (e.g., music services or social network APIs).

7.2 Extending PCM-lite and S2PTs for quality support

In order to perform an effective quality driven fusion and matchmaking, the definition of quality dimensions must be considered in the data models proposed in previous chapters. The PCM-lite is a meta-model that has been proposed to represent service information extracted from heterogeneous sources in a uniform and interoperable format (details in chapter 4). The meta-model can support the definition of quality assessments associated with extracted properties by introducing additional relation between property e values.

Instead, Source-to-Policy Templates (S2PTs) have been modelled in order to identify section of semi-structured and textual descriptions that contains a property values (details in chapter 5). To also permit the extraction of information necessary for quality assessments, S2PTs must be extended and remodelled.

In this section, additional PCM-lite relations and *Quality-Enabled Source-to-Policy Templates* (QE-S2PTs), that extend S2PTs are proposed in order to enable the quality assessments support.

7.2.1 PCM-lite relations for quality assessments

As described in chapter 4, PCM-lite defines an abstract relation between properties and their values. This approach allows users to define specific relation instances of the

abstract one, each one with a specific semantics. PCM-lite defines four basic relations instances that are namely, *hasValue*, *hasOperator*, *hasUnit* and *hasRelevance*.

In order to associate quality assessments to PCM-lite properties the following additional four relations instances are introduced.

hasAccuracy

This relation associates an accuracy assessment with a property.

hasCurrency

It associates currency with a property

hasTrustworthiness

It defines the association between trustworthiness and properties

hasQuality

It associates properties with the quality measure composed of accuracy currency and trustworthiness assessments.

```
...
pw:lastfmPolicy rdf:type pcm:Policy .

pw:lastfmPolicy pcm-lite:hasProperty pw:lastfmTags .
pw:lastfmPolicy pcm-lite:hasProperty pw:lastfmLicense .
pw:lastfmPolicy pcm-lite:hasproperty pw:lastfmRating .
...

pw:lastfmTags rdf:type pw:tags .

pw:lastfmTags pcm-lite:hasOperator pcm:all .
pw:lastfmTags pcm-lite:hasValue dbpedia:Music .
pw:lastfmTags pcm-lite:hasValue dbpedia:Social .

# Quality assessments
pw:lastfmTags pcm-lite:hasAccuracy "0.84" .
pw:lastfmTags pcm-lite:hasCurrency "0.73" .
pw:lastfmTags pcm-lite:hasTrustworthiness "0.57" .
pw:lastfmTags pcm-lite:hasQuality "0.753" .

...
```

LISTING 7.1: Portion of PCM-lite that defines quality assessments

An example of PCM-lite *policy* that introduces the relations above is shown in listing 7.1. The service description is defined in RDF/OWL according to the N-Triples data format [60]. The listing represent in detail the property *tags* of the Last.fm API extracted from ProgrammableWeb. The values associated with the properties are `Music` and `Social`

defined in the DBpedia ontology [76]. Following the technique proposed previously, quality assessments are also associated with the property trough the four additional relations.

7.2.2 Quality-enabled Source to Policy Templates

Compared to the S2PT formalization in section 5.2, *Quality-Enabled Source to Policy Templates* (QE-S2PTs) introduces a new entity and a new attribute, as shown in figure 7.1. As described in the previous section, trustworthiness is evaluated by exploiting source profiles available on social networks. Therefore, the entity *social profile* is introduced in the model. A social profile is composed by two attributes: a *social network wrapper* and a *profile ID*. The social network wrapper, identified by a URI, is a software component that is specialized to extract information of active users from a specific social network (e.g., Twitter or Facebook). In chapter 8, will be shown that the URI is exploited to interact with the social network wrapper through REST. Instead the profile ID is a string that identifies a specific source profile on a social network. To give an example, ProgrammableWeb is identified on Twitter by `@programmableweb`.

In order to allow the extraction of information necessary to evaluate the currency, the attribute *publication date* is added to the entity *template*. This attribute is a XPath query that refers to a portion of the source document that contains the date of publication by the source.

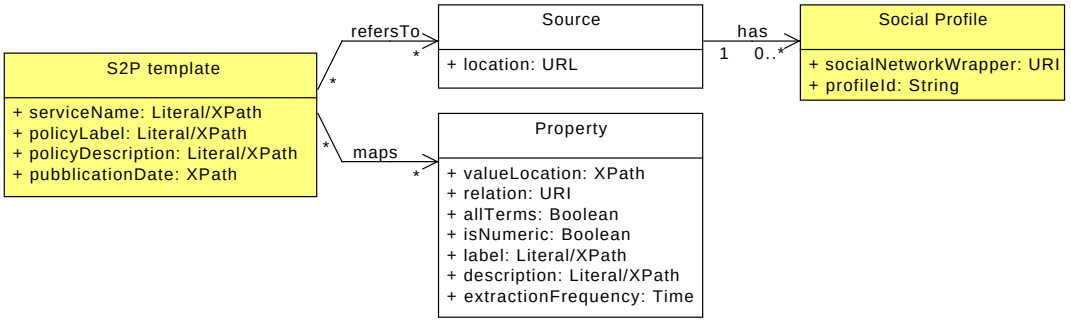

FIGURE 7.1: The QE-S2PT formalization

An example of QE-S2PT defined in XML is shown in listing 7.2. The template maps information provided by ProgrammableWeb published as Atom to policy according to PCM-lite. In the template example, the publication date of provided documents

```
<?xml version="1.0" encoding="UTF-8"?>
<s2pt:template
  xmlns:s2pt="http://pcm.itis.disco.unimib.it/q3-s2pt"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <s2pt:source>
    <s2pt:location>
      http://api.programmableweb.com/apis/
    </s2pt:location>
    <s2pt:publicationDate>
      /feed/entry/content/api/dateModified/
    </s2pt:location>
    <s2pt:SocialNetworkProfile>
      <s2pt:socialNetworkWrapper>
        http://snwrappers.itis.disco.unimib.it/facebook
      </s2pt:socialNetworkWrapper>
      <s2pt:profileID>
        programmableweb
      </s2pt:profileID>
    </s2pt:SocialNetworkProfile>
    <s2pt:SocialNetworkProfile>
      <s2pt:socialNetworkWrapper>
        http://snwrappers.itis.disco.unimib.it/twitter
      </s2pt:socialNetworkWrapper>
      <s2pt:profileID>
        @programmableweb
      </s2pt:profileID>
    </s2pt:SocialNetworkProfile>
  </s2pt:source>
  <s2pt:serviceName>
    /feed/entry/content/api/name
  </s2pt:serviceName>
  <s2pt:Property>
    ...
  </s2pt:Property>
  ...
</s2pt:template>
```

Listing 7.2: An example of QE-S2PT for ProgrammableWeb

is identified by the tag `dateModified` associated to each Web API available on ProgrammableWeb. Moreover, social network profiles are defined to extract information from Facebook and Twitter.

In the prospective of developing a tool that performs matchmaking on service properties available on the Web, software components, called *wrappers*, evaluate quality assessment. Wrappers extract property values and information for quality assessments exploiting QE-S2PTs. Adopting the techniques defined in the previous section, extracted information and related quality evaluation is made available as descriptions compliant to PCM-lite. Details of the architecture of wrappers will be provided in chapter 8.

7.3 Quality-driven description fusion

Several sources available on the Web can provide information about the same services. Moreover, the information provided can refer to the same property of a specific service. Besides, values that refers to a same property that are extracted from heterogeneous sources can be different and sometimes can provide contradictory information. To give an example, the data license of a Web API is available in provider public documentation and on ProgrammableWeb. Provider descriptions assert that the data license is free, instead information available on ProgrammableWeb report that the licensing is copyrighted.

To address this issue, a quality-driven technique for description fusion is provided in this section. The proposed approach exploits quality assessment available on PCM-lite descriptions in order to: (i) create a unique, complete and rich description for each service; and (ii) provide a correct property value by addressing potential contradictory information extracted from several sources.

The approach is based on the definition of semantic mappings between PCM-lite policies defined by the relation *matchesWith* (details are provided in chapter **??**). Each mapping is a relation of equivalence between properties defined in policies extracted from different sources. For example, values of the property *tags* can be extracted from ProgrammableWeb as well as WebMashup[2]. The definition of mapping between ProgrammableWeb *tags* and WebMashup *tags* defines that are equivalent properties therefore are fusible.

Afterwards, to each mapping is associated a *fusion function* that define the strategy that must be adopted to perform the fusion of properties. Each fusion function aim to provide a couple $< \tilde{v}, \tilde{q} >$ in which $\tilde{v}$ is the fused value of a specific property and $\tilde{q}$ is the quality associated to $\tilde{v}$. Three fusion functions are proposed: *aggregation*, *composition* and *selection*.

The aggregation function computes the fused value through a creation of a value set provided by different sources. The function can be formalized as follows. Let P a set of couples $< v_i, q_i >$, where v_i is a property value to be fused and q_i is the quality associated

[2]Available at: `http://www.webmashup.com`

with v_i. The aggregation function is defined as follows:

$$ag_P =< \tilde{v}, \tilde{q} > \text{, where } \tilde{v} = V \text{ and } \tilde{q} = \sum_{i=1}^{|P|} \frac{q_i}{|P|}.$$

The fused value $\tilde{v}$ is represented by the set V that is composed by all the property values that to be fused. Instead, $\tilde{q}$ is the fused quality that is computed as the average of the quality associated with the property that must be fused. This strategy is appropriated for symbolic property values that not assume contradictory information by aggregating values (e.g., *tags*).

The composition function computes a fused property by composing provided values and quality assessments. This strategy is modelled for numeric values. The formalization of the function is the following:

$$cp_P =< \tilde{v}, \tilde{q} > \text{, where } \tilde{v} = \sum_{i=1}^{|P|} \frac{q_i}{\sum_{j=1}^{|P|} q_j} \cdot \frac{v_i}{|P|} \text{ and } \tilde{q} = \sum_{i=1}^{|P|} \frac{q_i}{|P|}.$$

The fused value $\tilde{v}$ is computed as a weighted average based on the quality. The fused quality $\tilde{q}$ is evaluated as the average of the provided quality assessments in the same way is calculated for the aggregation function. This function can be used for numeric properties, such as *user rating* and *provider popularity*.

Finally, the selection function is based on a strategy that chooses the property value with the higher quality. This function is formalized as follows. The selection function is defined as:

$$s_P =< \tilde{v}, \tilde{q} >=< v_i, q_i > \in P \text{ such that } q_i = \max(Q)$$

This function is suitable for both symbolic and numeric property values that a composition or an aggregation can provide inconsistent results. For instance, if contradictory values are provided for *data license* (e.g., "free" and "copyrighted") an aggregation provide an ambiguous value for users. Instead, a composition on *service price* can provide wrong estimation that affect critically the adoption of the service in business context.

The choice of the best fusion function is made by domain experts. However, if domain experts are not available, the author suggests the adoption of selection function by default because the consistency of the fused values is always guaranteed.

On the base of the semantic mappings and fusion functions, the proposed *policy fusion* approach is described in Algorithm 2. The algorithm takes a set *extracted policies* associated with several services and the set of *mappings* as inputs, and returns a set of *fusedPolicies* as output. The first step is to group the extracted policies that describe a specific service into *clusters* (line 1). For each *property class* of the *policies* in a *cluster*, are identified all the *properties* that are equivalent to the selected one trough the definition of semantic mappings (lines 2-6). Subsequently, the selected *property* instances are fused according to one of the three functions presented previously (line 7). The resulting *fused property* is added to an instance of *fused policy* (line 8). Finally all the *fused policies* are aggregated and returned (lines 11-13).

Algorithm 2 PolicyFusion (*extractedPolicies*, *mappings*)

```
1: clusters ← groupByService(extractedPolicies)
2: for all c ∈ clusters do
3:    fusedPolicy ← ∅
4:    for all p ∈ policies do
5:       for all propertyClass ∈ p do
6:          properties ← getEquivalentProperties(propertyClass, mappings)
7:          fusedProperty ← fusion(properties)
8:          fusedPolicy ← fusedProperty
9:       end for
10:   end for
11:   fusedPolicies ← fusedPolicy
12: end for
13: return fusedPolicies
```

In chapter 8, *Service Matching Endpoints* (SMEs) will be introduced the as a software component of a general tool architecture for service matchmaking that exploits Web information. Each SME is specialized in a specific domain and it performs the policy fusion exploiting PCM-lite descriptions extracted by *wrappers*.

7.4 Quality-driven matchmaking and ranking

The proposed matchmaking process has the task of producing a list of ranked services according to their matching score with a given user request. The process is inspired by the one proposed in [49] and its novel characteristic is to consider the quality of information provided. Service descriptions that match with user requirements, but with a low quality assigned to property values, can not be considered trustworthy as

well as other descriptions. Therefore, information quality must affect the result of the matchmaking when the data is provided by Web sources, in order to provide a trustworthy list of matched services. On the best of author's knowledge, the matchmaking approach provided in this chapter is the first one that consider the quality of the service information compared with the literature, as shown in chapter 3.

The elements required to perform the proposed approach for quality-driven matchmaking are:

- a set of PCM-lite *policies* that are extracted from the Web and enriched by quality assessments;
- a *requested policy*, according to PCM-lite, that defines a set of properties required by a user;
- a set of *semantic mapping* that permit to identify offered properties that are comparable with requested properties.

Semantic mapping definitions are supported by PCM-lite by the *matchesWith* relation between properties. As introduced in the previous section, these mappings are also exploited for specifying fusible properties. Details an examples on semantic mappings are provided in chapter 4.

Algorithm 3 Matchmaking (*requestedPolicy*, *offeredPolicies*, *mappings*)

1: **for all** $p \in requestedPolicy$ **do**
2: $matchingProperties \leftarrow propertyMatching(offeredPolicies, mappings)$;
3: **end for**
4: **for all** $< reqp, offp > \in matchingProperties$ **do**
5: $LMS_{reqp,offp} \leftarrow localPropertyEvaluation(reqp, offp)$;
6: $Q_{offp} \leftarrow quality(offp)$;
7: $propertyProfile_{offp} \leftarrow < LMS_{reqp,offp}, Q_{offp} >$;
8: **end for**
9: **for all** $fp \in fusedPolicies$ **do**
10: $propertyProfiles(r) \leftarrow$ all the $propertyProfile_{offp}$ such that $offp \in r$;
11: $GMS_{fp} \leftarrow globalPolicyEvaluation(propertyProfiles(r))$;
12: $globalScores \leftarrow < fp, GS_{fp} >$;
13: **end for**
14: $rankedServices \leftarrow policyRanking(globalScores)$;
15: **return** $rankedServices$.

For seek of clarity, the quality-driven matchmaking approach is described through the Algorithm 3. The proposed approach consists of four main phases: *property matching*

(lines 1-3), *local property evaluation* (lines 4-8), *global policy evaluation* (lines 9-13), and *ranking* (line 14).

The *property matching* activity identifies the properties ($offp$) in the offered *policies* that match with the ones ($reqp$) in the *requested Policy* This phase is performed by using available mappings and semantic reasoning that is able to automatically identify equivalence relation between properties. The result is a set of couples $< reqp, offp >$ that can be matched.

After that, the *local property evaluation* is performed in order to compute, for each couple, a *Local Matching Score* (LMS). The LMS is a value in range [0, 1], that defines how the offered property satisfies the requested one. The techniques adopted for the evaluation are proposed in [49] and depend on the operators associated with the requested and offered properties and their type. For numeric properties, users can define single values and unbounded or bounded intervals, according to PCM-lite. Mathematical methods are exploited to measure how much the offered value is in range with the value interval defined by the requested property. On the base of this matching a specific LMS is returned.

Instead, symbolic properties can be defined by users as a set of values associated with a logical quantification operator (e.g., $\forall, \exists$) or set theory operator (e.g, $\subseteq$). Through automatic reasoning the semantic equivalence between requested and offered values is verified. On the base of the operator and the number of equivalent values between requests and offers the LMS is computed. To give an example, if the required value is {`music`,`download`}, the request operator is $\forall$ and the offered value is {`music`,`events`}, the LMS is 0.5 because only the half of elements of the requested set matches with the offered one.

This hybrid approach that combines mathematical function with reasoning is mainly adopted because reasoners are designed for inferences on ontologies, therefore they are inefficient and ineffective for advanced mathematical evaluations.

In order to consider that policies can be extracted from different Web sources, the proposed algorithm takes into account the quality associated with the offered properties. Therefore, every LMS is associated with a quality value Q defined by the overall quality

attached with each property in the offered policies. The associations are modelled through *property profiles* that are couples represented as $< LMS, Q >$.

Afterwards, property profiles are exploited by the *global policy evaluation* phase in order figure out how the requested policy globally matches with each offered policy. The third process step reach its objective by computing a *Global Matching Score* (GMS) based on the composition of LMS and Q. The GMS is calculated using the following formula:

$$GMS_P = \frac{1}{|PP|} \sum_{p \in PP} w_{LMS} \cdot r_p \cdot LMS_p + w_q \cdot q_p$$

where $w_{LMS} + w_q = 1$, PP is the property profile set associated with the policy P and r_p is the *relevance* of the property profile p for the user. The weights w_{LMS} and w_q are established by domain experts. Finally, the *ranking* phase delivery a ranked list of offered services ($rankedServices$) based on GMS computed on each policy.

In the prospective of the development of a concrete tool, the matchmaking process is implemented by distributed *Service Matching Endpoints* (SMEs) and a *Service Matching Orchestrator* that coordinates all the system components. The detailed modelling of a tool that performs extraction, quality assessments, fusion and matchmaking, by adopting the techniques described in this and previous chapters will be provided in the next chapter.

Chapter 8

A Lightweight Service Architecture for Matchmaking on the Web

In previous chapters, techniques have been proposed for (i) extraction of service property values from Web sources (chapters 5 and 6), (ii) quality assessment of the extracted information, (iii) quality-driven fusion of extracted service descriptions and (iv) quality-driven semantic matchmaking (chapter 7). These activities are components of an overall process that enable service matchmaking exploiting real descriptions available on the Web (details in chapter 3).

The main aim of this thesis is to propose a software architecture for making feasible the matchmaking on the Web. To reach this objective, this chapter shows the architecture of the *Policy Matchmaker and Ranker for Web* (PoliMaR-Web), a tool designed ad developed for implementing the overall matchmaking process that combine techniques provided in previous chapters. The proposed tool is modelled to address the following issues: (i) adaptation to each application domain enabled by a flexible and extendible architecture; (ii) to permit a scalable and efficient semantic matchmaking over the Web; (iii) promoting the diffusion of semantic descriptions of existing service available on the Web. The approach adopted for enabling these features is based on:

- a high architecture modularity that make flexible and extendible the matchmaking process;
- a lightweight service-based architecture enabled by using REST and Linked data principles;
- a distribution of components that exploits semantic reasoning in order to improve the matchmaking efficiency.

The architecture of this tool is introduced in other author's publications [11–13] and is described in detail in this thesis chapter as follows. The high level architecture of PoliMaR-Web is provided in section 8.1. After that, each main software components are described in detail: *Wrappers*, that are specialized on property extraction and quality assessments (section 8.2); *Service Matching Endpoints* (SMEs), that performs quality-driven fusion and matchmaking (section 8.3); and *Service Matching Orchestrator* (SMO), that orchestrate SMEs and return the overall matchmaking result to users (section 8.4).

8.1 High level architecture of PoliMaR-Web

The PoliMaR-Web is based on a distributed architecture composed of three kinds of components modelled as services: *wrappers*, *service matching endpoints* (SMEs) and *service matching orchestrators* (SMOs).

An architecture representation through several logical levels is provided in figure 8.1. On the base level, heterogeneous Web sources that provide service information are represented. These sources, can be documentation published by service providers, Web API repositories (e.g., ProgrammableWeb), wikis and social media (e.g., social network and forums).

The upper architectural level is composed by *wrappers*. Each wrapper is a service specialized to extract service property values from a specific source according to the techniques provided in chapters 5 and 6. Wrappers also performs quality assessments on the extracted information through techniques provided in chapter 7. These components are modelled to cope with the distributed and unstructured architecture of the Web, which provides for a huge amount of heterogeneous dispersed data.

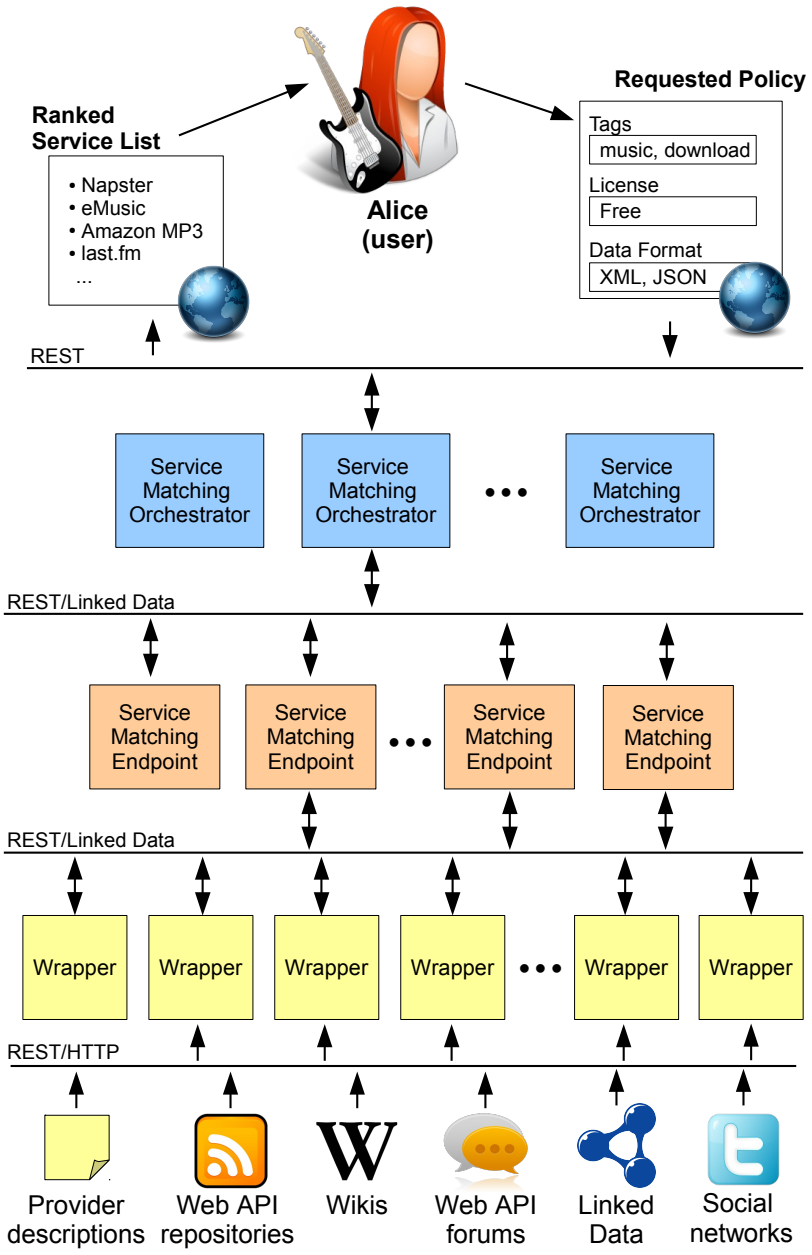

FIGURE 8.1: The PoliMaR-Web architecture (logical view)

The architectural level above wrappers is composed by *service matching endpoints* (SMEs). SMEs are services that collect semantic descriptions provided by one or more wrappers. On PCM-lite descriptions, SMEs perform the quality-driven fusion. Subsequently fused descriptions are exploited for the quality-driven semantic matchmaking according to techniques described in chapter 7. Each SME can be specialised in a specific service category (e.g., music, geolocalizzation, social networks) in order to provide an effective fusion and matchmaking for each domain.

As known in literature [10], semantic matchmaking has high response time due the adoption of reasoners. As introduced in chapter 4, reasoner are able to automatically infer logic relations that are not explicitly defined between concepts defined by ontologies. To give an example, reasoners can assert if a requested property value matches with a requested one by inferring equivalence relations. This issue can be addressed by SMEs, which can be instantiated to support parallel execution of fusion and matchmaking on several subsets of descriptions. Since such tasks may require the use of reasoners to deal with symbolic values, the possibility of concurrent execution dramatically improves the performance. With the proposed architecture, the load can be balanced among a dynamic set of SMEs, each of which has to evaluate smaller sets of descriptions. In chapter 9, experimentations that prove the efficiency due the distribution of SMEs will be described in detail.

The top level of the PoliMaR-Web architecture is composed by *service matching orchestrators* (SMOs). SMOs are the access points through which users can issue property requests, and get a ranked list of selected services. The task of SMO components is to coordinate SMEs in order to perform a distributed service matchmaking.

Wrappers, SMEs and SMOs interact each other according to a set of best practices that combine REST [3] and Linked Data [28] principles. Linked Data and REST shares some common characteristics [71].

URI as universal identifiers

Linked data proposes to identify concepts by resolvable URI, as well as, REST identifies resources. Therefore, Linked data concepts can be considered RESTful resources.

HTTP as communication protocol

Linked data concepts can be looked up via HTTP, as well as RESTful resources. In addition REST allows to modify resources through HTTP PUTs and DELETEs.

Links and hypermedia control

HATEOAS is a REST principle which specifies that resources are discoverable through hyperlinks provided by other resources. Linked Data implements HATEOAS through the definition of links between concepts in order to discover additional concepts.

In order to exploit the advantages of both REST and Linked Data, the author proposes to combine them according to the following five best practices for supporting consumption and modification of service description by human users and automatic tools as well.

Best Practice 1: Information modelling

Component services must provide information according to PCM-lite as a shared and formal model that specifies service properties.

The definition of a sound model for representing service properties is fundamental for allowing machines and humans to understand and manipulate descriptions exploiting the same syntax and semantics. As shown in chapter 4, PCM-lite is a good candidate to be adopted for representing service properties, because it is lightweight, expressive and compliant with other semantic models for service descriptions.

Best Practice 2: Semantic data model

According to Link Data principles, service descriptions need to be represented as RDF documents.

If descriptions are expressed in RDF, both syntax and semantic interoperability can be addressed. The former by adopting a shared semantic data model, the latter by enabling the use of Semantic Web tools, such as reasoners, which can make inference, and ontology matching tools [106], which are able to compare descriptions referring to different domain ontologies.

Best Practice 3: Common vocabulary

Property values should represent concepts that are linked to concepts available on the Linking Open Data Cloud.

The most popular Linked Data datasets are DBpedia [76] and Yago [78], which are the semantic representation of, respectively, Wikipedia and Wordnet [79]. Therefore, they refer to general-purpose ontologies to represent a vast portion of human knowledge. These two datasets are central hubs for hundreds of other datasets that define concepts

for specific domains (geography, music, etc.). If property values are defined as concepts available on the Linking Open Data Cloud, they represent well-known concepts, which means preventing ambiguity, synonymy and homonymy, and thus facilitating evaluations by automatic tools. Relations between datasets, defined as links, allow users and machines to discover additional information related to concepts that represent property values.

In addition, Linked Data is a medium for promoting the diffusion of semantic descriptions of existing services, that only few of them are available on the Web currently. Each year, the number of semantic descriptions that compose the Linking Data Cloud increase noticeably providing a big amount of interconnected information [28]. Following this trend, the adoption of wrappers can enable the diffusion of semantic service descriptions that originally can be provided as non-semantic documents.

Best Practice 4: Human interpretability

A natural language description, or label, must be associated with each service property.

Property definitions in pure RDF, without natural language descriptions, allow machine to manage information, but reduce human readability. To address this issue, a `rdfs:label` attribute or a `rdfs:comment` attribute should be specified for each service property.

Best Practice 5: RESTful descriptions

Component services must provide descriptions as aggregations of properties published as RESTful resources.

Linked Data supports only reading available information. By combining Linked Data with REST, adding, writing or modify descriptions through HTTP PUTs and DELETEs is enabled. Through HTTP methods human users and machines can access, evaluate and modify property values. According to HATEOAS, accessing to values is enabled by hyperlinks provided by resources that represent policy or properties linked each other by relationships.

In figure 8.2, a graph view of the PoliMaR-Web architecture is provided. As shown in the representation more than a SMO can be deployed. For instance, an orchestrator can be specialized for business users or common users.

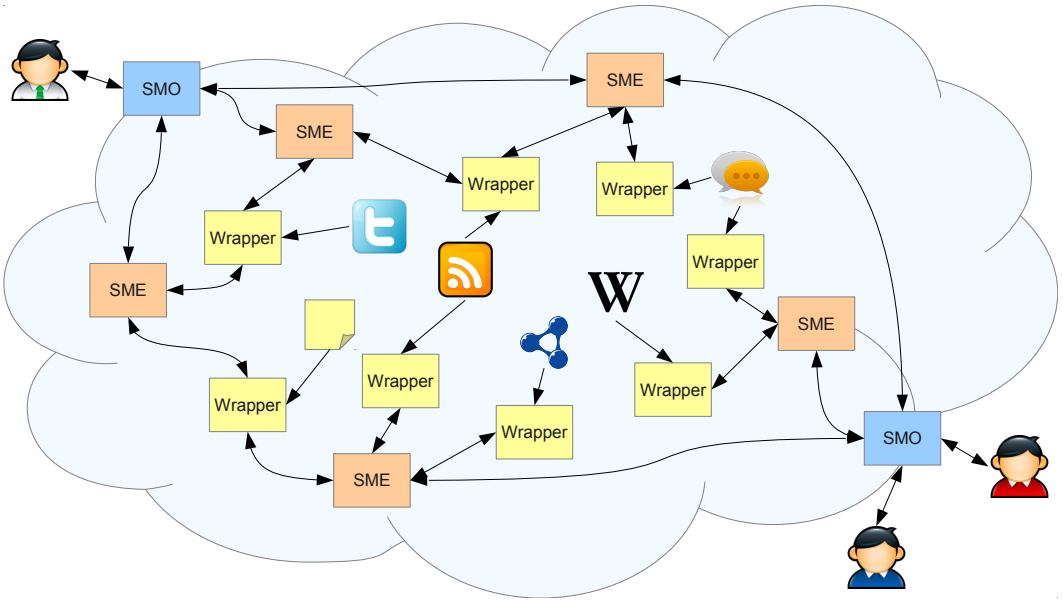

FIGURE 8.2: The PoliMaR-Web architecture (graph view)

Figure 8.2 shows that a strict relation between wrappers, SMEs and SMOs does not exist. A SME can be shared between several SMOs, in order to provide matchmaking on a particular sets of services. In addition, more SMEs, coordinated by the same orchestrator, can adopt to the same wrapper because each endpoint can be specialized in a domain, therefore it can exploit only a subset of descriptions extracted from a source.

The main advantage of modelling SMOs, SMEs and wrappers as services is allowing third-party actors to adopt functionalities provided by PoliMaR-Web components for their own tools. To give an example, a wrapper that provides semantic descriptions extracted from ProgrammableWeb can be exploited by a Web developer to build a mashup. Moreover, a user can integrate in his own application descriptions fused by a SME. The result is that PoliMaR-Web architecture is flexible and extendible allowing matchmaking improvements and user customizations. In addition, by providing functionalities and data through REST and Linked Data, the architecture is completely compliant to the Web standards.

In a real scenario, PoliMaR-Web components can be adopted as follows. Wrappers can be deployed by service providers for publishing semantic service descriptions with few efforts exploiting existing non-semantic descriptions. Other actors that can be interested to deploy wrappers are Web API repository providers (e.g., ProgrammableWeb) in order to make available their own datasets as Linked Data. Repository providers can also benefit for SMOs and SMEs deployment for providing a more effective Web API search engine for

their users. SMOs and SMEs can be also deployed by providers that offer a big amount of services (e.g., Google, Amazon, Yahoo!) in order support an effective matchmaking on their Web APIs. Following this prospective, PoliMaR-Web can be considered a first step towards a vision of a new cloud computing paradigm, called *Matching as a Service*, that provides flexible and extendible services for general purpose matchmaking.

After this high level description of the PoliMaR-Web architecture and its advantages, details on wrappers, service matching endpoints and service matching orchestrators will be provided in the next sections.

8.2 The wrapper service

A general architecture of wrappers is shown in figure 8.3. In order to provide semantic descriptions extracted from the Web and enriched with quality assessments, an architecture composed by seven modules is proposed. The interaction with the wrapper is implemented through a *REST/Liked Data interface* according to the practices in section 8.1. *URI solver* component aims to retrieve the correct policy instance on the base of the requested policy URI. The policy extraction and construction is performed by the *policy factory*. The implementation of the policy factory components depends on the source category. Specific techniques must be adopted to extract information from non-semantic descriptions and social media (as shown in chapters 5 and 6), therefore a specific implementation must be modelled.

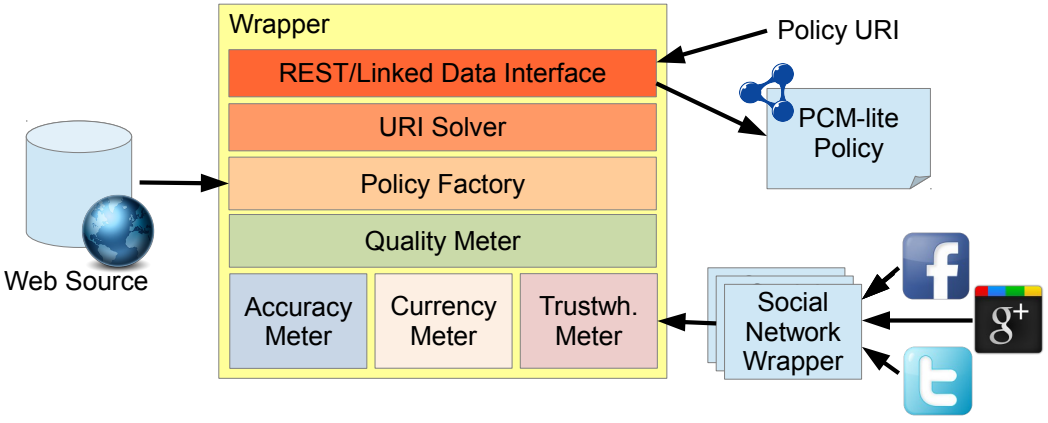

FIGURE 8.3: Wrappers architecture

To associate quality assessments with extracted property values the policy factory interact with a *quality meter* that performs quality evaluations exploiting an *accuracy meter*, a *currency meter* and a *trustworthiness meter*. The accuracy meter measure how is accurate the semantic value extracted on the base of the extraction technique adopted. Instead, the currency meter evaluates how is current the information extracted. Finally, the trustworthiness meter exploits social network information in order to figure out how the source is trustworthy. The trustworthiness meter uses *social network wrappers* that are specialized to provide activity of users that follows source profiles from a specific network (e.g., Twitter, Facebook or Google Plus). In this architecture, modules that provide quality assessments are optional, because sometimes provided information is not available (e.g., source profiles on social network) or evaluation are not necessary (e.g., dynamic property values are always current). Details on the techniques adopted for quality assessments are provided in section 7.1.

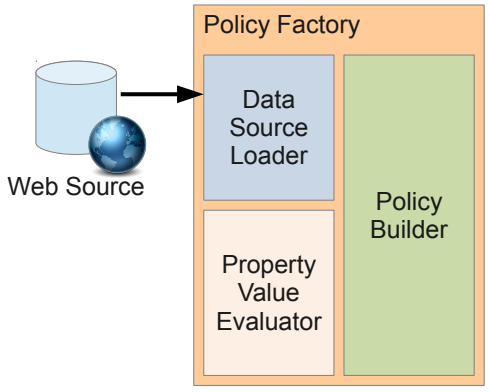

FIGURE 8.4: Generic policy factory architecture

Despite the implementation of the policy factory depends on the category of source in which service information is extracted, can be modelled its generic architecture. The representation of a generic policy factory is shown in figure 8.4. The generic architecture is composed by three basic elements: *data source manager*, *property value evaluator* and *policy builder*.

The *data source manager* is a component that focuses to retrieve source documents and related information. A second function of this module is to signal potential source faults due temporary unavailability or removed documents. Instead, the *property value evaluator* provides functionalities to extract property values and represent them as

semantic concepts. Finally, the *policy builder* orchestrates *data source manager* and *property value evaluator* in order to construct policies according to PCM-lite.

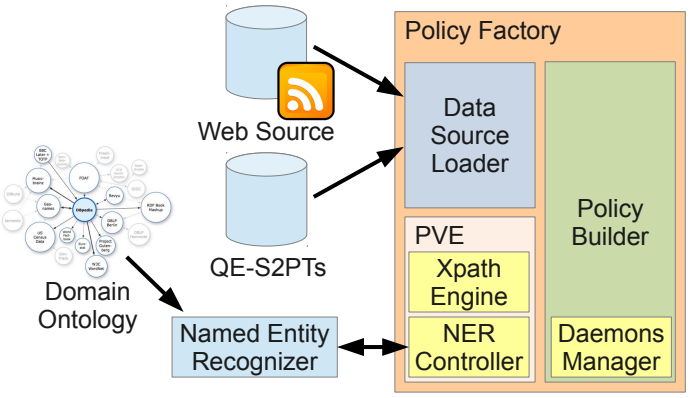

FIGURE 8.5: Policy Factory for non-semantic description

The architecture of policy factory focus on non-semantic descriptions is shown is figure 8.5. To perform the property value extraction, the factory exploits QE-S2PTs according to the techniques provided in chapter 5. QE-S2PTs defines XPath expression that localise text portions of Web documents, such as HTML and XML, that contains property values. For this reason, the *property value evaluator* includes a *XPath engine*.

For the value evaluation a *named entity recognizer* is necessary. This tool is able to identify concepts defined in a domain ontology (e.g., DBpedia [76] or YAGO [78]) in a textual descriptions. Therefore, through a *named entity recognizer*, it is possible to define semantic values provided by non-semantic sources that are defined in the reference domain ontology.

Several recognizers are available. Some of them are specialized for specific domain ontology (e.g., DBpedia spotlight [86]) or more general porpoise (e.g., AlchemyAPI[1]). Moreover each tool adopt several techniques which their effectiveness can depend on the domain [87]. To address this issue, the *named entity recognizer controller* is introduced as a component to provide a common interface to each *named entity recognizer*. In this way, recognizers can be easily substituted on the base of the application domain. The efficiency of these tools also depends on the technique adopted. In order to improve performances, a caching of NER evaluation results can be implemented.

[1]Available at: `http://www.alchemyapi.com`

Another relevant issue to be addressed is the management of Web information dynamism (details in chapter 2). To give an example, the Web API response time provided by API-status[2] service can change in every moment. In order to tackle this issue, it is introduced the *daemons manager* as a module of a *policy builder*. This component manages a set of *daemons* that are specialized on the extraction of a specific property value according to the QE-S2PT definition. Each *daemon* is a thread that performs a periodical extraction, then save the result in a cache memory. In this way, each time an extracted policy is requested, the wrapper returns always a service description with fresh and updated property values by accessing the cache.

A general policy factory that performs property extraction form social media can not be modelled. Values of social properties are computed by composing opinions and behaviour of users in order to figure out objective value. Each social media has specific characteristics that cannot be easily generalized. However, the generic architecture in figure 8.4 can be used as a design pattern to develop policy factories for each social media.

8.3 The service matching endpoint

Service matching endpoints (SMEs) perform quality-driven fusion and matchmaking according to the techniques provided in chapter 7. These two activities are performed by two main architectural components: the *policy fuser* and the *semantic matchmaking engine*.

As shown in figure 8.6, these components are accessible via a REST/Linked Data interface, according to the five best practices proposed in section 8.1. Through this interface, the SME give as input a requested policy. Subsequently, the policy fusion activity is performed by the *policy fuser*. In order to reach its aim, the *policy fuser* interacts with the *data manager* that is able to retrieve policies built by wrappers and infer semantic equivalences between properties to be fused. The policy retrieval is performed through a *wrapper invoker* that interacts with a set of wrappers identified by a list of URIs. After the wrapper invocation, the *data manager* exploits a *reasoner* to infer equivalent properties to be fused by exploiting semantic mapping that defines equivalence relations. Reasoner is

[2]Available at: `http://api-status.com/`

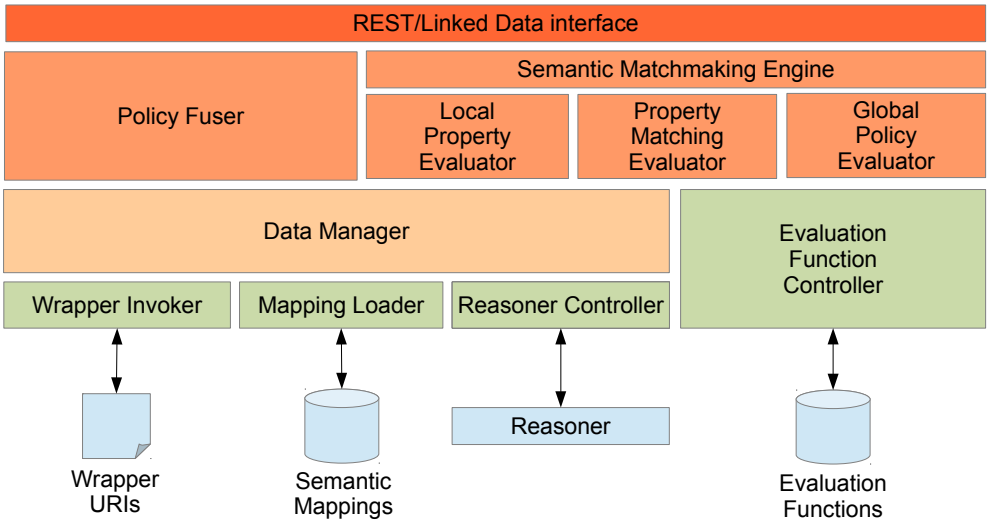

FIGURE 8.6: Service matching endpoint (SME) architecture

considered as an external component managed by a *reasoner controller*. Several reasoners are available in literature, which their efficiency and effectiveness depends on ontology complexity and characteristics [107]. Through specific implementations of the controller, the reasoner can be easily changed in order to perform effective evaluations on service descriptions.

Afterwards, the quality-driven matchmaking can be performed by the *semantic matchmaking engine* according to the approach proposed in section 7.4. The engine exploits the following three modules.

Property matching evaluator

This module implements the process necessary to perform the property matching phase. Through the *data manager*, this component submits the *matching mappings*, fused policies and the requested policy to the *reasoner* and receives a set of matching property couples as results.

Local property evaluator

The component implements the process necessary to perform the local property evaluation phase. For each matching couple produced by the *property matching evaluator*, the local matching score (LMS) evaluation is performed exploiting a specific function retrieved from the *evaluation function controller* that interacts with a *functions repository*.

Global policy evaluator

It implements the process necessary to perform the global evaluation phase. This component retrieves from the *evaluation function controller* the function to be used for the evaluation of the global matching score (GMS) based on the LMS and quality assessments.

The final output that is returned is a document that contains matched policies and related matching scores.

Through REST/Linked Data interface the SME is able to provide the following additional functionalities:

- retrieving fused descriptions provided by wrappers;
- policy fusion on user defined PCM-lite descriptions;
- matchmaking on policies defined by users.

These functionalities are not mandatory for the general approach, but are introduced in order to make flexible and customizable the overall matchmaking process. The retrieval of fused descriptions provided by wrappers can be performed through a HTTP GET that contains a fused policy URI. The interface interact directly with the *policy fuser*.

Instead, for fusing user defined PCM-lite descriptions, the interaction with the *policy fuser* is performed by submitting additional policies via HTTP PUTs. The fuser forwards policies to the *data manager* to identify fusible properties through reasoning, then performing the policy fusion. After that, fused policies can be retrieved through GETs.

Finally, through the REST/Linked Data interface, it is possible to interact directly with the *semantic matchmaking engine* in order to matchmake additional policies do not provided by wrappers. This functionality gives a list of additional offered policies and a requested policy through PUTs. During the same HTTP session opened by the requester, the matchmaking is performed. After that, matched policies with related matching scores are accessible via GETs.

8.4 The service matching orchestrator

The service matching orchestrator (SMO) is a PoliMaR-Web component that coordinate several SMEs in order to provide an efficient and effective matchmaking for users. Its architecture is shown in figure 8.7.

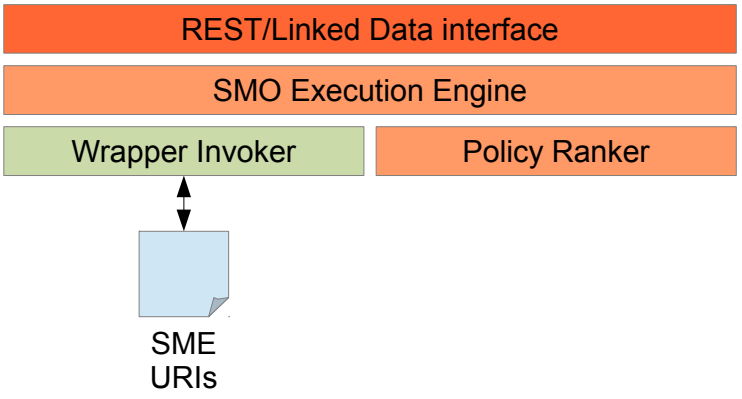

FIGURE 8.7: Service matching orchestrator (SMO) architecture

A SMO receives a requested policy through its REST/Linked Data interface. The requested policy can be defined by users through a form provided by a Web browser client that converts the filled form in a PCM-lite document, then forward it to the SMO. When the requested policy is received, the *SMO execution engine*, that implement the strategies of iteration with SMEs, start to execute the overall process. The first step performed by the engine is forwarding the requested policy to the *SME invoker*. The invoker distributes in parallel the requested policy to a set of SMEs identified by a set of URI defined in a list. Subsequently, the invoker waits that each endpoint performs a fusion and matchmaking, then it collects the provided results. After that, the SMO engine aggregates the matched policies and related matching score provided by each SME. The aggregated result is forwarded to the *policy ranker* that exploits the matching scores to sort the matched policies. The output is a ranked service list according to users constraints defined through the requested policy. The resulting list is finally returned to the user who submitted the request.

In this and previous chapters, techniques and a tool architecture for matchmaking exploiting service information distributed over the Web are provided. The next step necessary to prove the added value of the research work proposed in this thesis is to prove the effectiveness of the overall matchmaking process and the efficiency of a concrete

implementation of PoliMaR-Web. Effectiveness and efficiency evaluation will be provided in detail in the next chapter.

Chapter 9

Evaluation of Effectiveness and Efficiency

The *Policy Matchmaker and Ranker for Web* (PoliMaR-Web) is a tool designed for service matchmaking on heterogeneous service description available on the Web. In the previous chapter, the advantages of the PoliMaR-Web architecture are described but, to prove that the tool is able to make feasible the service matchmaking on the Web, some experimentation are needed. Two main characteristics must be verified: (i) effectivenesses of the overall matchmaking approach and (ii) efficiency of the tool. Proving the effectiveness is necessary for demonstrating that the matchmaking results provided are correct. Instead, to provide a tool that has a good response time means that the approach is Web scalable and can be adopted in a real context.

This chapters focus to describe several tests that demonstrate that PoliMaR-Web is effective and efficient. The first section shows in detail all the PoliMaR-Web prototype implementation and setup necessary to reproduce the experimentations (e.g., source exploited and external tools adopted). After that, the second section shows the effectiveness evaluation tests. Subsequently, the experimentation performed to measure the tool efficiency is described in the last section.

9.1 Experimentation setup

In order to allow the reader for reproducing the complete set of experimentations, the implementation of the PoliMaR-Web prototype adopted for the tests and setup detail are described in this section.

All the components of PoliMaR-Web are implemented by using Java in version 7 as programming language. Java is currently a standard for the development of each kind of application (e.g., web, businesses or mobile) and its main advantage is to be platform independent. Moreover, most of the tool available for managing Semantic Web descriptions are developed in this language. Several APIs are used to develop some PoliMaR-Web components. Restlet[1] (version 2.0.10) is the library used to implement the REST/Linked Data interfaces of wrappers, service matching endpoints (SMEs) and service matching orchestrators (SMOs) according to the practices introduced in chapter 8.

The semantic language which the prototype is able to manage is OWL [62], because most of the domain ontologies available on the Web are defined according to the RDF/OWL vocabulary. The management of ontologies is implemented exploiting the Jena[2] framework that permits to easily read and modify RDF/OWL ontologies. Through Jena APIs, modules that dynamic construct PCM-lite policies and interfaces with reasoners are implemented.

For parsing JSON documents provided external sources, the Google GSON[3] APIs are exploited. The management of XML and formats based on it (e.g., XHTML, RSS, Atom) is performed through the base APIs provided by Java. Also the adopted XPath engine, that is necessary to extract property values from semi-structured documents, is available in the Java libraries.

A named entity recognizer is a tool required by wrappers to identify semantic values from textual portion in semi-structured documents. For these experimentations, DBpedia spotlight [86] is adopted as named entity recognizer. This choice has been made because the domain ontology exploited is DBpedia [76], a semantic representation of Wikipedia. On one hand, this dataset is general-purpose, therefore is able to represent cross domain

[1] Available at: `http://www.restlet.org/`
[2] Available at: `http://jena.apache.org/`
[3] Available at `http://code.google.com/p/google-gson/`

concepts. On the other hand, DBpedia is the nucleus of the Linking Data Cloud [28]. In this way, extracted PCM-lite descriptions are linked with the Linking Data Cloud through concepts available in DBpedia as property values. DBpedia spotlight is available on the Web as RESTful service, therefore the PoliMaR-Web interacts whit the named entity recognizer through HTTP.

Instead, to perform fusion and matchmaking of semantic service descriptions extracted from the Web a reasoner is needed. PoliMaR-Web architecture permits to easily interchange reasoners. However, for the tool experimentations, Pellet[4] (version 2.3.0) is adopted to perform reasoning on RDF/OWL descriptions. Pellet is chosen because it is not one of the most efficient OWL reasoners available [107], therefore potential performance improvements, due to reasoning parallelization through distributed SMEs, are more visible.

Currently, source code of this PoliMaR-Web implementation is available on *SourceForge*[5] under GPL open source license. Moreover, a working demo of this matchmaking tool is also available on the Web[6]

The sources involved for the tests are ProgrammableWeb[7], WebMashup[8] and Wikipedia[9]. As introduced in chapter 2, ProgrammableWeb is a Web API portal that provides more that 7500 service descriptions defined by users. Its dataset is accessible via REST and provided service information as Atom feeds. Instead, WebMashup is an on-line Web API repository. On this Website approximately 1700 service descriptions are available as XHTML pages. Finally, a page of Wikipedia that provides a comparison between several Music services[10] is considered as a data source for these tests. This XHTML page provides information about properties for 60 services specialized in the music domain.

Three semi-structure sources have been considered for the following reasons. First of all, semi-structured description need the utilization of both reasoner and named entity recognizer. Therefore, it is possible to evaluate how much these component affects the effectiveness of the overall approach. Moreover, these sources provide information about a big amount of services. A big dataset is necessary to figure out the tool performances and

[4]Available at: `http://clarkparsia.com/pellet`
[5]Source code available at: `http://sourceforge.net/projects/polimar/`
[6]Demo available at: `http://jeeg.siti.disco.unimib.it:8080/polimar/discovery.jsp`
[7]Available at: `http://www.programmableweb.com/`
[8]Available at: `http://www.webmashup.com/`
[9]Available at: `http://www.wikipedia.org/`
[10]Available at: `http://en.wikipedia.org/wiki/List_of_online_music_databases`

response time. Another reason is the availability of approximately 1500 descriptions that are fusible. Hence, it is possible to evaluate effectiveness and efficiency of the description fusion.

For these evaluation tests, we selected five properties that are addressed by at least two sources: *tags*, *interaction protocol*, *data formats*, *licensing* and *user rating*. *Tags* represent a set of keywords that describe the service domain and functionalities (e.g., "music", "download", "events", "social"). As shown in chapter 2, most of Web APIs provide functionalities through REST, but some of them adopt SOAP, XML-RPC or JavaScript. The *interaction protocol* refers to the kind of service approach is used to implement a specific API. Instead, *data formats* describe which formats are adopted by the service to provide data (e.g., XML, JSON, RDF). The *licensing* defines the usage license of the data provided by the service. The data can be copyrighted, free distributed, under a Creative Commons license or other hundreds of licenses. This property is important to be considered in order to prevent legal issues. Finally the *user rating* represents a quality evaluation provided by a set of users. The first four properties are represented by symbolic values, instead *user rating* is the only numeric property. Moreover, the latter one is the most dynamic property. Most of the properties values considered in this scenario are symbolic to stress named entity recognizer and reasoner in order to measure the PoliMaR-Web scalability.

These properties are provided by each source as shown in table 9.1. Values of the five considered properties can be extracted from ProgrammableWeb. Instead, WebMashup does not provide only the service licensing information. As complete opposite, the Wikipedia page offers only values of licensing property. In this way, description fusion can be performed on all the considered properties.

Web source	Tags	Int. protocol	Data formats	Licensing	User Rating
ProgrammableWeb	✓	✓	✓	✓	✓
WebMashup	✓	✓	✓	✗	✓
Wikipedia	✗	✗	✗	✓	✗

TABLE 9.1: Web API properties described in the considered sources

9.2 Effectiveness evaluation

The aim of a service matchmaking tool is to identify a set of service that fulfil constraints defined by a requester. If the results are wrong, the functionalities of the matchmaker are not satisfactory for the users, therefore the tool is useless. This is the reason of effectiveness evaluations: to verify the quality of the results returned by PoliMaR-Web.

PoliMaR-Web implements a complex process that performs extraction, quality assessments, fusion and matchmaking of service descriptions available on the Web. Therefore, the effectiveness of the overall approach depends on each component activity. Moreover, some of these activities exploit external tools, such as named entity recognizer and reasoner. Hence, these tools can affect the overall effectiveness of the proposed approach. In the next subsections, effectiveness evaluations of each component phase and of the overall approach is are performed.

9.2.1 Extraction effectiveness

In the experimentation scenario proposed in the previous section, three semi-structured sources are exploited to extract policies according to PCM-lite. In the extraction process the elements that can affect the effectiveness are QE-S2PT definitions and the named entity recognizer. As introduced in chapters 5 and 7, *Quality-enabled source to policy templates* (QE-S2PTs) associate each property class with a XPath expression that identifies the portion of structured document that contains the property value. These expressions are interpreted by a XPath engine that is able to extract the portion of document that contains the value. These templates are defined manually, therefore, except human errors in XPath expression definitions, the extraction is always correct.

Instead, a correct semantic representation of extracted property values depends on named entity recognizers. The effectiveness of the recognizer can depend on the adopted techniques and the domain [87]. In order to figure out how a named entity recognizer affect the property value extraction, a precision and recall evaluation is performed. The precision p_e and recall r_e are evaluated as follows:

$$p_e = \frac{|C_c \cap I_c|}{|I_c|} \text{ and } r_e = \frac{|C_c \cap I_c|}{|C_c|},$$

where C_c is the set of correct concepts that represent the extracted values and I_c is the set of concepts extracted. These measures are performed on the extraction of 1816 property values available in 500 descriptions provided by the three considered sources.

For this test, the recognizer adopted is DBpedia Spotlight as introduced in the previous section. Precision and recall are computed for the four qualitative properties considered for test scenario. *User rating*, is not considered because the extraction of numeric values is straightforward and provide always correct evaluations. Moreover, the numeric property extraction do not involve the named entity recognizer.

Properties	p_e	r_e
Tags	0.65	0.68
Int. protocol	0.93	0.94
Data formats	0.92	0.91
Licensing	0.79	0.78
Average	0.83	0.83

TABLE 9.2: Effectiveness of property value extraction

The results of the effectiveness evaluations of property value extraction are shown in table 9.2. For each property, the average of precision and recall for each value that refers to a specific properties have been computed. On *interaction protocol* and *data formats*, the extraction has a good precision and recall. For *licensing*, the extraction effectiveness is a bit lower. Instead, the extraction of *tags* sometimes in not very effective. The cause of these results are strictly related to the recognition approach adopted by DBpedia Spotlight. This tool identifies concepts on the base of terms context in the textual description. For terms of the same context (e.g., licensing), the recognizer is able to identify better related concepts. Compared to the other properties, *tags* provide values that can not easily localized in specific context (e.g., "social", "music", "events"). For this reason, values of this property are more hard to be identified.

9.2.2 Implementation of quality assessments techniques

Quality assessments of information extracted from Web sources are exploited for fusing descriptions of same services and performing the semantic matchmaking. Therefore, the effectiveness of quality assessments is indirectly estimated by evaluating the fusion and matchmaking. In order to allow the reader to reproduce these experimentations on

quality-driven fusion and matchmaking, this subsection describes how quality assessment techniques are implemented.

Techniques implemented for quality assessments of extracted property values are described in section 7.1. *Accuracy* evaluation depends on the effectiveness of the named entity recognizer. This metric is computed through a similarity score between the concepts identified by DBpedia Spotlight. More the concepts are similar each other and with the property name, higher is probability of a good identification, then a good accuracy.

Instead, to compute the *currency* the date of publication of the document source must be provided. For the three properties considered, this information is defined in specific source document portions. Therefore, in order to extract this information a XPath query the refers to the section is defined in the QE-S2PT of each source. To measure the currency, wrappers execute the XPath query, parse the date, then on the base of the distance with the current day evaluate the quality metric.

Finally, *Trustworthiness* is measured by exploiting the activity of users that follows the three Web sources on Twitter and Facebook. All sources have a Twitter profile, but only ProgrammableWeb and Wikipedia have a Facebook public page. The activity measured in the following way: on Twitter, by counting the number of followers that retweet a source public profile; instead, on Facebook, by counting the number of users that like the public page and like posts of the source. This information is retrieved by social network wrappers that that are implemented exploiting the Twitter[11] and Facebook[12] Web APIs to collect data of the last three days of activity.

9.2.3 Quality-driven fusion evaluation

According to the techniques provided in section 7.3, the elements that can affect the effectiveness of the quality-driven fusion can be semantic mappings and fusion functions. The mappings, that identifies fusible properties, are defined manually in this scenario, therefore, except human errors, the identification of fusible property values is always correct. If mappings produce the expected results, the effectiveness of the fusion depends on the strategies implemented by the fusion functions.

[11] Available at: `https://dev.twitter.com/docs/api`
[12] Available at: `http://developers.facebook.com/docs/`

The fusion functions, that implements the strategy adopted for fusing values, follow one of these approaches: aggregation of symbolic values, composition of numeric values or selection of the value with highest quality. For each property, the following fusion approaches are adopted.

Two fusion approaches can be used for *tags*: aggregation and selection. On one hand, the aggregation of tags do not provide an inconsistent value. Moreover, it enriches the property description by providing additional terms that allow users to better find the service. On the other hand, the section prevent the aggregation of wrong values identified by the named entity recognition. For this reason will be evaluate which of the two strategies is more effective.

The value composition is performed for the *user rating*, The composition is computed through a quality-weighted average of the quantitative values. In this way, the fused value is only marginally affected by low quality extracted values. By computing the average of provided values, the fused value does not provide wrong or contradictory values. Therefore, this property is not considered to evaluate the effectiveness of policy fusion.

Instead, the other three properties are fused using the selection strategy. Aggregation of values for these properties can provide contradictory information. To give an example, the result of a licensing values aggregation can provide a licence that defines copyright restrictions and free distribution at the same time. Therefore, to provide a consisted value, the selection is the best fusion function.

In order to measure the effectiveness of the policy fusion, the precision and recall have been measured. For the fusion, the precision p_f and the recall r_f are computed as follows:

$$p_f = \frac{|D_c \cap F_c|}{|F_c|} \text{ and } r_f = \frac{|D_c \cap F_c|}{|D_c|},$$

where F_c is the set of concepts that represents a fused property value and D_c is the set of concepts that correctly describe extracted values. The set D_c is defined by value the are correctly extracted from the source despite the result obtained by the named entity recognizer. In this way, if results obtained are higher compared to the extraction precision and recall, it means that the fusion increases the effectiveness of the extraction. For this

reason, the descriptions considered for this test are the same used for the extraction tests (subsection 9.2.1).

Property	Source			Effective fusions	Total fused values
	Prog.Web	WebMashup	Wikipedia		
Tags	250	190	-	190	250
Int. Protocol	250	173	-	173	250
Data formats	245	105	-	105	245
Licensing	103	-	60	60	103
User rating	250	190	-	190	250
Total	1098	658	60	718	1098

TABLE 9.3: Amount of fused values for each property

In table 9.3, is shown the amount of property values that are fused according to the considered dataset. The result of the policy fusion provides 1098 fused values, but only 718 are effectively fused between several values. The remaining 380 values are provided by only one source, therefore are not affected by the fusion.

To evaluate the fusion effectiveness, three tests with different configurations of the weights have been run to compute the aggregate quality measure as defined in section 7.1. The experiments have the goal to analyse the correlation between the weights associated with the individual qualities and the effectiveness of the fusion process.

Tests	Weights			Tags (ag.)		Tags (sel.)		Int. Prot.		Data For.		Licensing	
	w_a	w_c	w_t	p_f	r_f	p_f	r_f	p_f	r_f	p_f	r_f	p_f	r_f
Test A	0.33	0.33	0.33	0.73	1	0.72	0.93	0.93	0.93	0.95	0.88	0.33	0.75
Test B	0.5	0.25	0.25	0.73	1	0.79	0.94	0.94	0.93	0.97	0.89	0.68	0.80
Test C	0.6	0.3	0.1	0.73	1	0.84	0.94	0.95	0.95	0.98	0.93	0.98	0.93

TABLE 9.4: Effectiveness of the fusion process

The overall results of the experiments are shown in Table 9.4, where w_a, w_c and w_t represent respectively the accuracy, currency and trustworthiness weight. These results confirm the hypothesis that the fusion is more effective if the constraint $w_a > w_c > w_t$ is enforced, as anticipated in section 7.3. The *licensing* property is the one that is most affected by a violation of the constraint because the fusion is performed between Wikipedia, which presents high trustworthiness and low accuracy, and ProgrammableWeb, which presents medium trustworthiness and good accuracy. Therefore, the high trustworthiness of Wikipedia has negative impact on the fusion accuracy.

Source	ProgrammableWeb	Wikipedia
Source value	"Under Creative Commons license"	"CC"
Extracted semantic value	Creative Commons (license)	Comedy Central (tv channel)
Accuracy	0.25	0.1
Currency	0.52	0.57
Trustworthiness	0.5	0.85
Aggregate (Test A)	0.423	**0.506**
Aggregate (Test B)	0.38	**0.405**
Aggregate (Test C)	**0.356**	0.316

TABLE 9.5: Quality assessment of the license values extracted for Last.fm Web API

Table 9.5 shows the impact of different weight configurations (Tests A, B and C defined in Table 9.4)) on the fusion of two *licensing* values extracted from ProgrammableWeb and Wikipedia. By parsing the text provided, DBpedia spotlight returns the concept "Creative commons" from the text "Under Creative Commons license", instead, "Comedy Central", a popular TV channel, from the text "CC". The fusion process, which selects the value with highest aggregate quality (highlighted in boldface in the table), selects the wrong value under the configurations A and B (Comedy Central is not a license term); when accuracy is highly weighted, the correct value is selected. The accuracy of the value extracted from ProgrammableWeb is higher than the accuracy of the value extracted from Wikipedia because the source value is a longer text, which makes DBpedia Spotlight perform better.

Other relevant evaluation results are related to the technique adopted for the fusion of the property *tags*. Precision and recall of the fusion based on the aggregation function is stable despite the variation of weights. This result was attended because the aggregation approach is not driven by the quality. Moreover, the recall is always 1, because the adopted strategy aggregate all the values extracted from each source.

A more interesting result is obtained by comparing aggregation and selection strategies. Despite the adoption of the first strategy is straightforward for the *tags* property, the latter strategy provides a more effective fusion. The reason of this experimental result is due to the effectivenesses of the named entity recognizer. Previous tests shown a low effectiveness in the extraction of this property, because DBpedia spotlight often identifies wrong values. The aggregation of a relevant amount of wrong values affects dramatically the precision. Therefore, the adoption of the selection strategy, that choose the value

with highest quality, provides a better fused value result. The lesson learned through this result is: for property values that are identified with difficulties by named entity recognizers, a more effective fusion is provided by adopting the selection strategy.

The most relevant result of these evaluations is that the property fusion increase the precision and recall of the values extracted thought named entity recognition, by comparing previous results available in table 9.2. The overall improvements computed on the average of each property is 11% of higher precision and 10% of higher recall. The property most affected is *tags* with an improvement of the 29% of precision and 38% of recall. This result is due to the adoption of the selection function that provides returns a values with higher quality, then more accurate, with the result of an increase of effectiveness.

9.2.4 Overall matchmaking effectiveness

The final step of the overall approach implemented by PoliMaR-Web is the quality-driven semantic matchmaking. The matchmaking is modelled trough a processes that combines automatic reasoning (for symbolic properties), mathematical evaluations (for numeric properties) and quality assessments, as described in section 7.4. By using service description that are extracted from Web source and sequentially fused, the effectiveness of this last activity corresponds to the effectiveness of the overall approach.

According to proposed matchmaking approach, the elements that can affect the effectiveness are: semantic mapping, quality assessments, mathematical evaluations, effectivenesses of the policy fusion. As introduced in the previous sections, mappings and QE-S2PTs are defined manually, therefore, except human mistakes, they are correct. If mappings are accurate, also the inferences of the reasoning are correct, hence, the matching of properties and the evaluation of symbolic values is effective.

To verify what exactly affect the overall effectiveness of the overall matchmaking, also for the last phase, precision and recall have been measured. The precision p_m and the recall r_m of the quality-driven matchmaking are defined as follows:

$$p_m = \frac{|C_v \cap M_v|}{|M_v|} \text{ and } r_m = \frac{|C_v \cap M_v|}{|C_v|},$$

where, M_v is a set of matched property values and C_v is the set of values that are correctly matched and represents correctly the property after the extraction and fusion.

Therefore, if a value is correctly matched, but it is wrongly extracted, it is not in the set C_v. By considering also the effects of the extraction and fusion, the evaluation is performed on the overall approach.

The exploited dataset is composed by fused policies returned by the description fusion performed during previous tests. In this way, it is possible to verify how the final step of the overall approach is affected by the previous phases. Moreover, to measure the precision and recall, 20 realistic requested policies are given as input to PoliMaR-Web. These requests define user constrains on 5 service domain: music (e.g., download and streaming services), geolocalization, social media, enterprise (e.g., CRM services) and internet (e.g., DNS and cloud computing services).

Properties	p_e	r_e
Tags	0.84	0.94
Int. protocol	0.95	0.95
Data formats	0.98	0.93
Licensing	0.98	0.93
User rating	1	1
Average	0.95	0.95

TABLE 9.6: Effectiveness of the overall matchmaking approach

The results of the evaluation are shown in table 9.6. Each row represents the precision and recall averages for each property. A first aspect that can be considered is that the matchmaking on *user rating* is always correct. As explained in the previous subsections, also the extraction and fusion of quantitative values is always correct. It means that quality assessments and mathematical functions, exploited for the matchmaking of numeric values, are correct too. Therefore, the final result is that the overall approach is always effective for numeric properties.

By comparing the result of this test and the previous test (in table 9.4), interesting aspect emerge: precisions and recalls of the process after the fusion and after the matchmaking are the same for qualitative properties. The result of this comparison shows that quality assessments and matchmaking are always effective for qualitative values because the precision and recall is not decreased.

To conclude, the effectiveness of the overall approach that performs matchmaking on Web descriptions mainly depends on: (i) the techniques adopted for the named entity

recognition, necessary for the property value extraction and (ii) the strategy adopted for the description fusion.

9.3 Efficiency evaluations

In order to make feasible the proposed matchmaking approach on the Web in necessary to prove that PoliMaR-Web has an acceptable response time. The aim of this section is to show that the transmission overheads introduced by distributing the matchmaking and extraction components are largely compensated by faster semantic matchmaking.

For efficiency evaluations, the PoliMaR-Web prototype components were deployed as follows:

- three wrappers, each one specialized for a specific source are hosted by an Intel Core2 Q6700 quad-core @ 2.67 GHz with 2GB RAM;
- twenty SMEs are deployed on eight Intel Xeon X5550 quad-core CPUs @ 2.67GHz with 2 GB of RAM;
- an Intel Core2 T5500 CPU @ 1.66GHz with 2 GB of RAM hosts a SMO.

Each node is equipped with 64-bit Linux operating system (kernel version 3.0.15) with 100 Mbps network connection.

For measuring the efficiency, two different tests have been performed. *Test A* to measure the execution time to perform the selection of 500 *offered policies* fused from ProgrammableWeb, WebMashup and Wikipedia repositories. The *requested policy* used for this test defines constraints on the five properties of the test scenario described in section 9.1. To stress the evaluation of performance, it is chosen to extract properties with symbolic values. Symbolic properties requires intensive reasoning activities for fusion and matchmaking, therefore it is possible to better highlight potential performance improvements introduced by the distributed architecture.

The results computed as the average of twenty measurements using the same *requested policy* are shown in Figure 9.1. The time measurements have been obtained by varying the number of involved SMEs. *Test A* highlights that the execution time behaves like

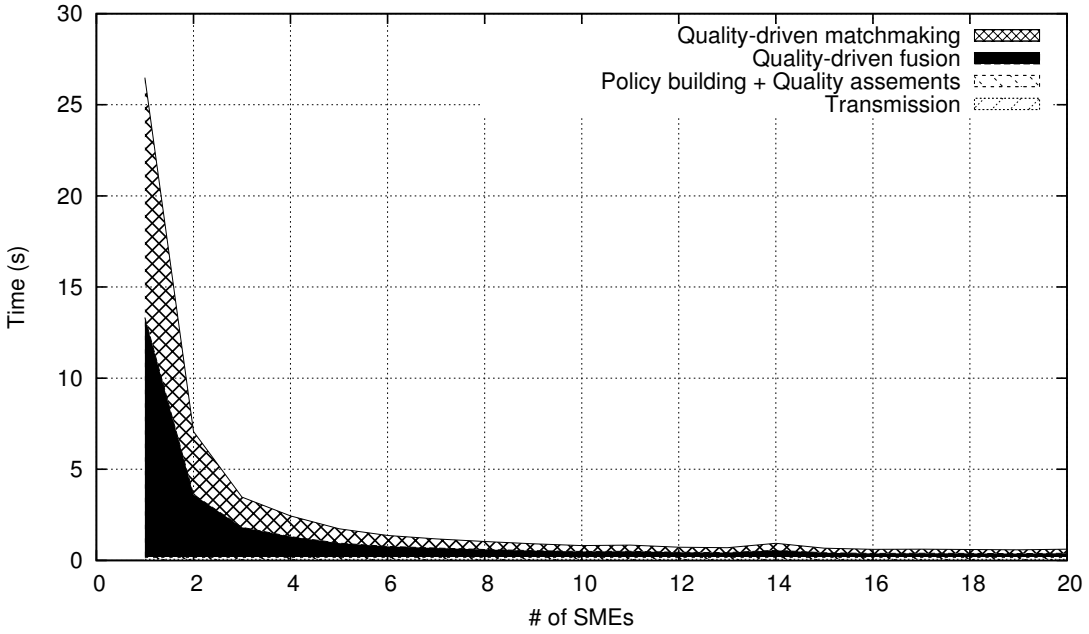

FIGURE 9.1: Test A: Execution time for 500 *policies*

a hyperbola arm. The results show that there is a relevant performance improvement already with few SMEs, resulting an abatement approximately of an 8.6 factor with 4 SMEs, and no relevant improvement with more than 10 SMEs.

Making use of the threshold identified by *Test A*, *Test B* is performed with the aim of analysing the execution time of the most relevant process phases with 10 SMEs, with different number of fused descriptions.

The results of *Test B*, reported in Figure 9.2, show that the most time consuming phases are the matchmaking and the policy fusion. This result was attended because fusion and matchmaking requires reasoning. The response time of the two phases is almost the same because the reasoning is performed on similar knowledge bases. The exponential increment of time consumption for reasoning intensive activities is a well-known problem reported in the literature [10].

The policy building and quality assessments are the less time consuming of the overall process. The effectiveness of the caching strategy implemented by wrappers (described in chapter 5) is proved by this test. For each text portion processed by the DBpedia spotlight, the average time required is approximately 1 second. As shown in table 9.3, for 500 descriptions of the considered dataset, 1276 qualitative values must be extracted with a potential wrapper response time of approximately 1300 seconds. Moreover,

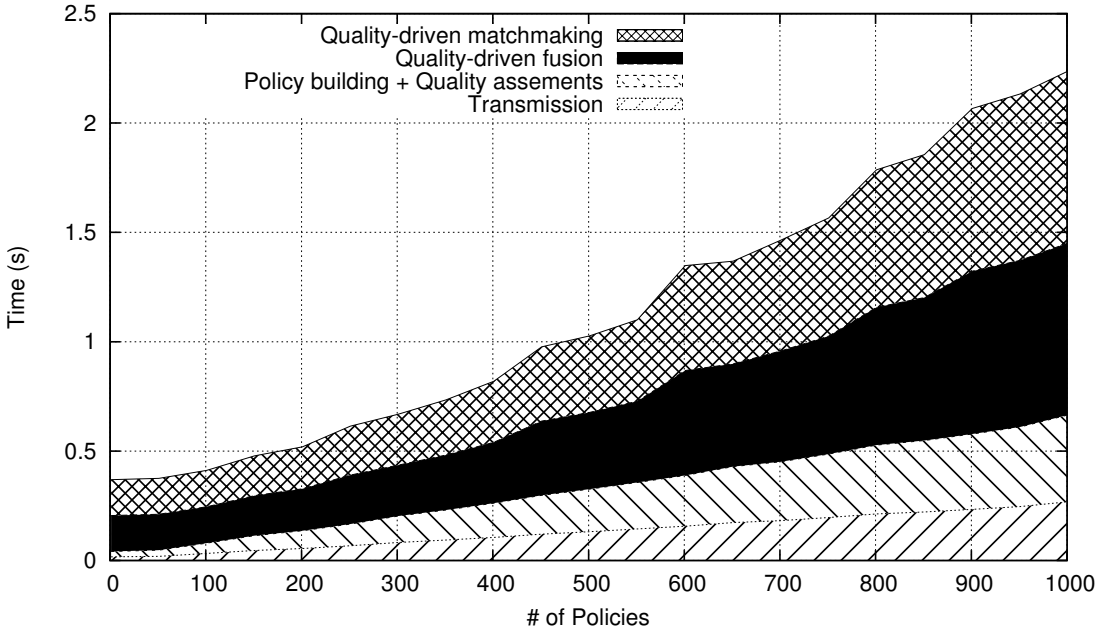

FIGURE 9.2: Test B: Execution time for 10 SMEs

wrappers response time can be also dramatically affected by performing an on-the-fly extraction of the all the properties as a solution for the management of dynamic properties. ProgrammableWeb and WebMashup have high response time, moreover these sources are able to manage only a limited amount of parallel requests. It is measured that the on the on-the-fly retrieval of 500 descriptions require approximately 23 seconds by maximising the possible parallelism of requests to Web sources. The approach of caching of extracted values, with a frequent refresh for dynamic properties (e.g., 60 seconds for user rating), has the following advantages: (i) reduce the wrapper response time to few milliseconds; (ii) support dynamic properties without overload of both sources and wrappers due to network traffic; (iii) management of temporary periods of source unavailability by providing cached values.

The measured transmission time between the orchestrator, the matching endpoints and wrappers is acceptable for the performance improvements obtained by performing parallel reasoning (approximately 250 milliseconds for 1000 extracted descriptions). For this experimentation data exchanged between components is in RDF based on XML format. Therefore, a further reduction of the transmission time can be performed by adopting data format for RDF that are less verbose, such as N-Triples [60], Notation3 [58] and Turtle [59]. Finally, notice that the *policy ranking* activities do not appear in the charts

since it execution time is negligible, because its implementation is based on a quicksort algorithm that has a polynomial execution time.

Several experimentations, provided in this chapter, proved: (i) the effectiveness of the overall matchmaking approach proposed in this thesis and (ii) the efficiency of its implementation through the PoliMaR-Web distributed architecture. Next chapter will draw conclusions of the issues addressed in this thesis and will discuss open issues and future extensions.

Chapter 10

Conclusions and Future Extensions

Today, the number of available services is continuously increasing on the Web thanks to a phenomenon called Web APIs. Web APIs popularity leads to a new scenario in which increasing amount of information about services is available on the Web. All this information can be exploited to support users for discovering services.

The SOC literature proved that the most effective discovery technique is service matchmaking, which requires the availability of semi-structured documents defined according a common model to specify service properties. However, the Web scenario is different. Information is mainly provided as textual or semi-structured document defined through heterogeneous models and vocabularies. Web sources can publish invalid or contradictory data. Web information is dynamic, then can change along the time. Moreover, the management of dispersed information can introduce scalability issues due to network response time and traffic. Therefore, literature approaches can not be applied on service information available on the Web.

In the current scenario, this thesis proposes an overall approach per enabling and effective and efficient service matchmaking by exploiting Web information. The approach is composed of four phases: (i) extraction of service property values; (ii) quality assessments of the extracted information; (iii) quality-driven fusion of service descriptions; and (iv) quality-driven matchmaking. Through the design and implementation of the proposed

approach, several issues have been addressed. Solution proposed and related lessons learned for addressing these issues are provided as follows.

Representation of heterogeneous service information

A requirement of service matchmaking is the representation of service properties according to a common model. For representing heterogeneous service information available on the Web, the *Lightweight policy-centered meta-model* (PCM-lite) is proposed. PCM-lite is a meta-model that allows users to express rich service properties in order to specify requests and services descriptions according to Semantic Web standards. As provided in the literature, the adoption of semantic descriptions enables for an effective matchmaking by addressing homonymy and synonymy between heterogeneous vocabularies.

PCM-lite is composed of three concepts: *policy*, *properties* and *values*. Each policy represents a collection of property associated with a service, then each property is composed of values. A novel aspect introduced by PCM-lite is the proposal of a unique syntax to express functional and non-functional properties. This feature allows users to classify properties as FP or NFP according to the context, domain or personal perception.

PCM-lite is defined as a lightweight abstract model that can be easily mapped to other semantic models, such as OWL-S [39] and WSMO [40], in order to matchmake descriptions defined by heterogeneous models. Moreover, the low complexity of PCM-lite contains the computational resources necessary for the evaluation of semantic descriptions [10, 12, 16].

Extraction of explicit property values from heterogeneous Web sources

Most of service information is published through HTML webpages or semi-structured data, such as XML or JSON documents. Sometimes only parts of Web documents provides informations about service properties.

For addressing property value extraction from HTML and semi-structured documents, an approach based on *source-to-policy templates* (S2PTs) and named entity recognition is adopted. S2PTs map properties with portions of documents that contain specific property values to be extracted. The identification of portions is performed through the specification of XPath expressions. By executing these expressions through a XPath

engine, document portions are extracted. Then, portions are processed by a named entity recognizer which identifies, in a text, semantic concepts that represents property values according a domain ontology. Finally, semantic values are collected and represented as PCM-lite descriptions.

The proposed approach is the first in literature that is able to extract both functional and non-functional properties. However, this solution suffers of some limits. S2PTs are defined manually by users, moreover they must be modified every time the structure of a Web documents is modified. In addition, effectiveness of named entity recognition strongly depends on the property domain and techniques adopted [87].

Extraction and evaluation of subjective properties

Social media, such as social networks, blogs or forums, provides opinions about services that can be collected for evaluating properties. The results of a survey performed by the author highlighted that some of those properties are subjectively interpreted by users.

In this thesis, techniques are provided for evaluating the following three properties: provider popularity, vitality of Web API forums and service usage. Provider popularity is evaluated through search query volumes of providers on Google trends. Vitality of Web API forums is computed as number of daily users that posted messages. Finally, service usage is evaluated through the number of mashups published on ProgrammableWeb that adopt a specific service.

Compared with the state of the art, the novelty introduced in the proposed approaches to consider the temporal information. Properties are evaluated through information published in the last period because opinions of users can change during the time.

An interesting aspect is that the three properties are correlated each other for most services. By exploiting this correlation, techniques for estimating missing values can be adopted.

A current limit of the proposed techniques is the lack of a general approach because each social media has particular characteristics that cannot be easily generalized. However, there are not better solutions in the literature, at least to the best of author's knowledge.

Quality assessments of service information

Service information can be published by service providers and third-party sources. This information can be invalid in terms of accuracy, currency and trustworthiness, therefore techniques for evaluation of these three quality measures are proposed in this thesis.

Accuracy is computed by evaluating the context of terms in the text portions exploited for extracting property values. If terms are of the same context , is probable that the information and extracted values are accurate. Currency is measured through the publication date that can be associated with Web documents. Finally, trustworthiness of Web sources is evaluated by exploiting social networks. The proposed technique provides a trustworthiness value by assuming that the more Web sources profiles on social network are followed by users, the more the source information provided is valid.

Fusion of disperse service information

Descriptions about the same services and properties can be provided by different sources could be contradictory or might need to be integrated for providing a unique rich description for each service.

For addressing this issue, an approach based on quality-driven fusion of PCM-lite descriptions is proposed. The approach associates a specific fusion function with semantic mapping that specifies fusible properties.

Three fusion functions that exploit quality assessments can be associated with mappings: aggregation, composition and selection. The aggregation function returns a fused value composed of all the values of properties to be fused. Instead, the composition function provides a unique value that represents the average of several extracted values. Finally, the selection function, choose the value with the higher quality.

Experimental results showed that the proposed technique performs an effective fusion.

Service matchmaking

Matchmaking approaches available in the literature exploits local repositories composed of valid descriptions, moreover the quality of information is not addressed by current

matchmakers. This thesis introduces a novel approach for service matchmaking that exploits quality assessments of Web information.

The proposed approach exploits user requests and service descriptions defined according to PCM-lite. The approach combines semantic reasoning, for evaluating symbolic values, and mathematical functions, for comparing numeric properties. Quality assessments associated with service properties drive the ranking of matchmaking results in order to give priority to valid and dependable service descriptions. Experimental results showed the effectiveness of the quality-driven matchmaking on service properties extracted from real service descriptions available on the Web.

Global scalability

The management of big amount of data that is dispersed over the Web and the relevant amount of computational resources required by semantic reasoning introduce scalability issues for implementing a feasible the process.

The *Policy Matchmaker and Ranker for Web* (PoliMaR-Web) is designed trough a lightweight distributed architecture based on services. The architecture is based on three components: *wrappers*, *service matching endpoints* (SMEs) and *service matching orchestrators* (SMOs). Each wrapper is a component specialised on the extraction of service properties and quality assessments for a specific Web source, to publish PCM-lite descriptions. SMEs collect descriptions from several wrappers, therefore perform quality-driven fusion and matchmaking. Finally, SMOs support the interaction with requesters and orchestrate SMEs in order to distribute the computation: SMOs receive user requests, then forward them to SMEs. When each SME terminates the matchmaking phase, SMOs collect and rank the results that are returned to the requester.

Architecture components interact each other with a communication protocol that combines REST [3] and Linked Data [28] practices. The Linking Data Cloud, that represent interconnected datasets that provide information as Linked Data, is continuously increasing on the Web. Following this trend, publishing service descriptions through Linked Data promotes the diffusion of semantic documents that represents services. Instead, the adoption of REST allows users to modify service descriptions, a functionality not supported by Linked Data. Moreover, REST and Linked Data, that directly exploit

HTTP, result more efficient and compliant with the nature of the Web compared with SOAP [4, 5].

Performance experimentation shows that response time is dramatically reduced by distributing wrappers and SMEs. These results prove that PoliMaR-Web architecture is scalable, therefore the overall approach is feasible and applicable on real context.

10.1 Future directions

Despite relevant results obtained by modelling the overall approach for matchmaking on Web information and the implementation of PoliMaR-Web, several possible extensions for the solutions proposed in this thesis can be identified.

Experimentation with users

Precision and recall measurements have been used for proving the effectiveness of the overall approach. However, an experimentation that involves users is necessary to confirm the correctness of results returned by matchmaking. As introduced in chapter 2, users can have a subjective interpretation also for values that are explicit on the Web. Experimentations with humans can highlight if the evaluation of property values respect the user perception.

Increase of automation

The proposed approach for property values extraction requires the definition of S2PTs. This activity requires a relevant human effort, especially if structure of Web documents changes frequently. The definition of techniques for building S2PTs based on the automatic identification of documents portions that contains property values is still an open issue that must be addressed.

Generalization of subjective property evaluations

In this chapter, the evaluation of subjective properties through social media represents a preliminary work. Several aspects of this issues must be addressed. The modelling of a general approach for evaluating subjective property must be defined. Moreover, experimentation with users must be addressed also for evaluating the effectiveness of the techniques.

Support of human services

This thesis proposes an approach focused on services as software components. The proposed approach can be remodelled and reimplemented for matchmaking of services that direct interact with users, such as public transportations, mobile applications (apps) and tourist services. For addressing this issue, studies of interaction design are required to allow users to define PCM-lite documents by hiding the complexity of semantic descriptions. Moreover, the diffusion of smartphones and tablets enabled a new scenario in which mobile devices can be exploited as sensors for inferring user contexts. User context is an additional aspect that can be considered for improving service matchmaking results.

Development of the Matching-as-a-Service paradigm

The *cloud computing* is a model for providing on-demand computing resources (e.g., networks, servers, storage, applications) through services [108]. According this model, wrappers, SMEs and SMOs can be redesigned for implementing a Matching-as-a-Service (MaaS) paradigm, in which service provides on-demand resources for general purpose semantic matchmaking. The implementation of MaaS paradigm can become a solution for performing scalable evaluations on the continuous increasing amount of information available as Linked Data.

Bibliography

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: a Research Roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255, 2008.

[2] H. Haas and A. Brown. Web Services Glossary. Technical report, W3C, February 2004. Available at `http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/`.

[3] R.T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California - Irvine, 2000.

[4] C. Pautasso, O. Zimmermann, and F. Leymann. RESTful Web Services vs. "big" Web Services: Making the Right Architectural Decision. In *Proceedings of the 17th international conference on World Wide Web*, WWW, pages 805–814, 2008.

[5] S. Vinoski. Putting the "Web" into Web services: interaction models, part 2. *Internet Computing, IEEE*, 6(4):90–92, 2002.

[6] M. Maleshkova, C. Pedrinaci, and J. Domingue. Investigating Web APIs on the World Wide Web. In *Proceedings of the IEEE 8th European Conference on Web Services*, ECOWS, pages 107 –114, 2010.

[7] A. Bernstein and M. Klein. Towards high-precision service retrieval. In *Proceedings of the Iternationa Semantic Web Conference*, ISWC, pages 84–101. 2002.

[8] Yiqiao W. and E. Stroulia. Flexible interface matching for web-service discovery. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, WISE, pages 147–156, 2003.

[9] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1), 2003.

[10] S.B. Mokhtar, A. Kaul, N. Georgantas, and V. Issarny. Towards efficient matching of semantic Web service capabilities. In *Proceedings of the International Workshop on Web Services Modeling and Testing*, WS-MATE, 2006.

[11] L. Panziera, M. Comerio, M. Palmonari, F. De Paoli, and C. Batini. Quality-driven Extraction, Fusion and Matchmaking of Semantic Web API Descriptions. *Journal of Web Engineering*, 11(3):247–268, 2012.

[12] L. Panziera, M. Comerio, M. Palmonari, and F. De Paoli. Distributed matchmaking and ranking of web apis exploiting descriptions from web sources. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, SOCA, 2011.

[13] L. Panziera, M. Comerio, M. Palmonari, C. Batini, and F. De Paoli. PoliMaR-Web: multi-source semantic matchmaking of Web APIs. In *Proceedings of 13th Internationa Conference on Web Information Systems Engineering*, WISE, pages 812–814, 2012.

[14] M. Comerio, F. De Paoli, M. Palmonari, and L. Panziera. Web service contracts: Specification and matchmaking. *Advanced Web Services*, 2013. To appear on June, 30th.

[15] L. Panziera and F. De Paoli. A Framework for Self-descriptive RESTful Services. In *Proceedings of Fourth International Workshop on RESTful Design*, WS-REST, 2013. To appear.

[16] L. Panziera, M. Palmonari, M. Comerio, and F. De Paoli. WSML or OWL? A lesson learned by addressing NFP-based selection of semantic Web services. In *NFPSLAM-SOC '10 Workshop Proceedings*, 2010.

[17] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *Internet Computing, IEEE*, 6(2):86–93, 2002.

[18] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, et al. Web services business process execution language version 2.0. Technical report, OASIS, April 2007. Available at `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[19] T. Bray. Internet Media Type registration, consistency of use. Technical report, W3C, June 2002. Available at `http://www.w3.org/2001/tag/2002/0129-mime`.

[20] S. St. Laurent, J. Johnston, and E. Dumbill, editors. *Programming Web Services with XML-RPC*. O'Reilly, 2001.

[21] D. Flanagan. *JavaScript: the definitive guide*. O'Reilly Media, Incorporated, 2006.

[22] L.D. Paulson. Building rich web applications with AJAX. *Computer*, 38(10):14–17, 2005.

[23] D. Benslimane, S. Dustdar, and A. Sheth. Services mashups: the new generation of Web applications. *Internet Computing, IEEE*, 12(5):13–15, 2008.

[24] C. Pautasso. Composing RESTful services with JOpera. In Alexandre Bergel and Johan Fabry, editors, *Software Composition*, volume 5634 of *Lecture Notes in Computer Science*, pages 142–159. Springer Berlin Heidelberg, 2009.

[25] C. Pautasso. BPEL for REST. In *Proceedings of 7th International Conference on Business Process Management*, BPM, pages 278–293, 2008.

[26] M.J. Hadley. Web application description language (WADL). Technical report, W3C, August 2009. Available at `http://www.w3.org/Submission/2009/SUBM-wadl-20090831/`.

[27] T. Berners-Lee, J. Hendler, O. Lassila, et al. The Semantic Web. *Scientific american*, 284(5):28–37, 2001.

[28] T. Heath and C. Bizer. Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1(1):1–136, 2011.

[29] G. Klyne, J.J. Carroll, and B. McBride. Resource description framework (RDF): Concepts and abstract syntax. Technical report, W3C, February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

[30] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, W3C, January 2008. Available at `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.

[31] U. Küster and B. König-Ries. Towards standard test collections for the empirical evaluation of semantic web service approaches. *International Journal of Semantic Computing*, 2(03):381–402, 2008.

[32] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopeckỳ, and J. Domingue. iServe: a linked services publishing platform. In *Proceedings of the workshop on Ontology Repositories and Editors for the Semantic Web*, ORES, 2010.

[33] Shuping Ran. A model for Web services discovery with QoS. *SIGecom Exchanges*, 4(1):1–10, March 2003.

[34] P. Rajasekaran, J. Miller, K. Verma, and A. Sheth. Enhancing Web services description and discovery to facilitate composition. *Semantic Web Services and Web Process Composition*, pages 55–68, 2005.

[35] M. Klusch, B. Fries, and K. Sycara. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):121–133, 2009.

[36] M. Klusch and F. Kaufer. Wsmo-mx: A hybrid semantic web service matchmaker. *Web Intelligence and Agent Systems*, 7(1):23–42, 2009.

[37] M. Klusch and P. Kapahnke. The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2012.

[38] P. Plebani and B. Pernici. Urbe: Web service retrieval based on similarity evaluation. *Knowledge and Data Engineering, IEEE Transactions on*, 21(11):1629–1642, 2009.

[39] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, et al. OWL-S: Semantic markup for web services. Technical report, W3C, November 2004. Available at `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`.

[40] D. Fensel, H. Lausen, A. Polleres, J. De Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services – The Web Service Modeling Ontology*. Springer, 2006.

[41] U. Lampe, S. Schulte, M. Siebenhaar, D. Schuller, and R. Steinmetz. Adaptive matchmaking for RESTful services based on hRESTS and MicroWSMO. In

Proceedings of the 5th Workshop on Emerging Web Services Technology, WEWST, pages 10–17, 2010.

[42] J. M. Garcia, D. Ruiz, A. Ruiz-Cortes, O. Martin-Diaz, and M. Resinas. An Hybrid, QoS-Aware Discovery of Semantic Web Services Using Constraint Programming. In *Proceedings of the International Conference on Service-Oriented Computings*, ICSOC, pages 69–80, 2007.

[43] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A QoS-aware Selection Model for Semantic Web Services. In *Proceedings of the International Conference on Service Oriented Computing*, ICSOC, 2006.

[44] L.H. Vu, M. Hauswirth, F. Porto, and K. Aberer. A search engine for QoS-enabled discovery of semantic web services. *International Journal of Business Process Integration and Management*, 1(4):244–255, 2006.

[45] H.Q. Yu and S. Reiff-Marganiec. A method for automated web service selection. In *Proceedings of the Congress on Services*, SERVICES, pages 513–520, 2008.

[46] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm. Preference-based Selection of Highly Configurable Web Services. In *Proceedings of the International Conference on World Wide Web*, WWW, pages 1013–1022, 2007.

[47] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci. IRS-III: A broker-based approach to semantic Web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(2):109–132, 2008.

[48] S.B. Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers. Easy: Efficient semantic service discovery in pervasive computing environments with qos and context support. *Journal of Systems and Software*, 81(5):785–808, 2008.

[49] M. Palmonari, M. Comerio, and F. De Paoli. Effective and Flexible NFP-Based Ranking of Web Services. In *Proceedings of Service-Oriented Computing, 7th International Joint Conference, ICSOC-ServiceWave 2009*, pages 546–560, 2009.

[50] N. Steinmetz, H. Lausen, and M. Brunner. Web Service Search on Large Scale. In *Proceedings of Service-Oriented Computing, 7th International Joint Conference, ICSOC-ServiceWave 2009*, pages 437–444, 2009.

[51] E. Al-Masri and Q.H. Mahmoud. A framework for efficient discovery of Web services across heterogeneous registries. In *Proceedings of the IEEE International Conference on Consumer Communications and Networking*, CCNC, pages 415–419, 2007.

[52] K. Gomadam, A. Ranabahu, M. Nagarajan, A.P. Sheth, and K. Verma. A faceted classification based approach to search and rank Web APIs. In *Proceedings of the IEEE International Conference on Web Services*, ICWS, pages 177–184, 2008.

[53] B. Tapia, R. Torres, and H. Astudillo. Simplifying mashup component selection with a combined similarity-and social-based technique. In *Proceedings of the 5th International Workshop on Web APIs and Service Mashups*, Mashups, page 8. ACM, 2011.

[54] Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung. Ux-an architecture providing qos-aware and federated support for uddi. In *Proceedings of the IEEE International Conference on Web Services*, ICWS, 2003.

[55] Y. Liu, A. Ngu, and L. Zeng. QoS computation and policing in dynamic web service selection. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW, pages 66–73. ACM, 2004.

[56] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. Technical report, W3C, March 2001. Available at `http://www.w3.org/TR/wsdl`.

[57] R. Chinnici, J.J. Moreau, A. Ryman, and S. Weerawarana. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. Technical report, W3C, June 2007. Available at `http://www.w3.org/TR/wsdl20/`.

[58] T. Berners-Lee and D. Connolly. Notation3 (N3): A readable RDF syntax. Technical report, W3C, March 2011. Available at `http://www.w3.org/TeamSubmission/n3/`.

[59] D. Beckett and T. Berners-Lee. Turtle - Terse RDF Triple Language. Technical report, W3C, March 2011. Available at `http://www.w3.org/TeamSubmission/turtle/`.

[60] J. Grant and D. Beckett. RDF Test Cases. Technical report, W3C, February 2004. Available at `http://www.w3.org/TR/rdf-testcases/#ntriples`.

[61] D. Brickley, R.V. Guha, and B. McBride. RDF vocabulary description language 1.0: RDF Schema. Technical report, W3C, February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.

[62] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). Technical report, W3C, December 2012. Available at `http://www.w3.org/TR/2012/REC-owl2-overview-20121211/`.

[63] ESSI WSML working group. The Web Service Modeling Language - WSML. Technical report, ESSI WSMO, August 2008. Available at `http://www.wsmo.org/TR/d16/d16.1/v1.0/`.

[64] J. Kopeckỳ, T. Vitvar, C. Bournez, and J. Farrell. SAWSDL: Semantic annotations for WSDL and XML schema. *IEEE Internet Computing*, 11:60–67, 2007.

[65] J. Kopeckỳ, K. Gomadam, and T. Vitvar. hRESTS: An HTML microformat for describing RESTful Web services. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, WI-IAT, pages 619–625. IEEE, 2008.

[66] J. Cardoso, C. Pedrinaci, T. Leidig, P. Rupino, and P. De Leenheer. Open semantic service networks. In *Proceedings of the International Symposium on Services Science*, ISSS, 2012. Available at `http://linked-usdl.org/`.

[67] C. Pedrinaci and J. Domingue. Toward the next wave of services: Linked Services for the Web of data. *Journal of Universal Computer Science*, 16(13):1694–1719, 2010.

[68] A. Barros and D. Oberle, editors. *Handbook of Service Description: USDL and its Methods*. Springer, 2011.

[69] R. Alarcón and E. Wilde. From RESTful services to RDF: Connecting the Web and the Semantic Web. *CoRR*, abs/1006.2718, 2010.

[70] R. Verborgh, S. Coppens, T. Steiner, J.G. Vallés, D. Van Deursen, and R. Van de Walle. Functional descriptions as the bridge between hypermedia apis and the

semantic web. In *Proceedings of the Third International Workshop on RESTful Design*, WS-REST, pages 33–40. ACM, 2012.

[71] K.R. Page, D.C. De Roure, and K. Martinez. REST and Linked Data: a match made for domain driven development? In *Proceedings of the Second International Workshop on RESTful Design*, WS-REST, pages 22–25. ACM, 2011.

[72] F. De Paoli, M. Palmonari, M. Comerio, and A. Maurino. A Meta-model for Non-functional Property Descriptions of Web Services. In *Proceedings of 6th IEEE International Conference on Web Services*, ICWS, pages 393–400, 2008.

[73] I. Toma, D. Roman, and D. Fensel. On describing and ranking services based on non-functional properties. In *Proceedings of the Third International Conference on Next Generation Web Services Practices*, NWESP, pages 61–66. IEEE Computer Society, 2007.

[74] K. Kritikos and D. Plexousakis. Semantic qos metric matching. In *Proceedings of the European Conference on Web Services*, ECOWS, pages 265–274. IEEE Computer Society, 2006.

[75] E. Giallonardo and E. Zimeo. More semantics in qos matching. In *Proceedings of the International Conference on Service-Oriented Computing and Application*, SOCA, pages 163–171, 2007.

[76] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[77] M. Morsey, J. Lehmann, S. Auer, C. Stadler, and S. Hellmann. Dbpedia and the live extraction of structured data from Wikipedia. *Program: electronic library and information systems*, 46(2):157–181, 2012.

[78] F.M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217, 2008.

[79] C. Fellbaum. Wordnet. *Theory and Applications of Ontology: Computer Applications*, pages 231–243, 2010.

[80] F.M. Suchanek, M. Sozio, and G. Weikum. SOFIE: a self-organizing framework for information extraction. In *Proceedings of the 18th International Conference on World Wide Web*, WWW, pages 631–640, 2009.

[81] N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, M. Ramanath, and G. Weikum. ANGIE: active knowledge for interactive exploration. *Proceedings of Very Large Database Endowment (PVLDB)*, 2(2):1570–1573, August 2009.

[82] J.L. Ambite, S. Darbha, A. Goel, C.A. Knoblock, K. Lerman, R. Parundekar, and T.A. Russ. Automatically constructing semantic web services from online sources. In *Proceedings of the 8th International Semantic Web Conference*, ISWC, pages 17–32, 2009.

[83] V. Saquicela, L.M. Vilches-Blázquez, and O. Corcho. Lightweight semantic annotation of geospatial RESTful services. In *Proceedings of the 8th Extended Semantic Web Conference*, ESWC, pages 330–344, 2011.

[84] M. Taheriyan, C. Knoblock, P. Szekely, and J. Ambite. Rapidly Integrating Services into the Linked Data Cloud. In *Proceedings of 11th International Semantic Web Conference*, ISWC, pages 559–574, 2012.

[85] P.A. Ly, C. Pedrinaci, and J. Domingue. Automated information extraction from Web APIs documentation. In *Proceedings of 13th Internationa Conference on Web Information Systems Engineering*, WISE, pages 497–511, 2012.

[86] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics, pages 1–8, 2011.

[87] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[88] A.M. Kaplan and M. Haenlein. Users of the world, unite! The challenges and opportunities of Social Media. *Business horizons*, 53(1):59–68, 2010.

[89] F.Y. Wang, K.M. Carley, D. Zeng, and W. Mao. Social computing: From social informatics to social intelligence. *Intelligent Systems, IEEE*, 22(2):79 –83, march-april 2007.

[90] A. Ahmad. Social network sites and its popularity. *International Journal of Research and Reviews in Computer Science (IJRRCS)*, 2(2), 2011.

[91] Q. Wu, A. Iyengar, R. Subramanian, I. Rouvellou, I. Silva-Lepe, and T. Mikalsen. Combining quality of service and social information for ranking services. In *Proceedings of Service-Oriented Computing, 7th International Joint Conference, ICSOC-ServiceWave 2009*, pages 561–575, 2009.

[92] X. Liu, N. Jiang, Q. Zhao, and G. Huang. iSocialMash: Convergence of social networks and services composition on a mashup framework. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, SOCA, 2011.

[93] J.L. Rodgers and W.A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):pp. 59–66, 1988.

[94] W. Pirie. Spearman rank correlation coefficient. *Encyclopedia of statistical sciences*, 1988.

[95] C.R. Rao. *Linear statistical inference and its applications*, volume 22. Wiley-Interscience, 2009.

[96] A. Bjorck. *Numerical methods for least squares problems*. Number 51. Society for Industrial and Applied Mathematics, 1996.

[97] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[98] D. Barbagallo, C. Cappiello, C. Francalanci, and M. Matera. Enhancing the Selection of Web Sources: A Reputation Based Approach. In *Proceedings of the 12th International Conference on Enterprise Information Systems*, ICEIS, pages 464–476, 2010.

[99] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM Comput. Surv.*, 41:16:1–16:52, July 2009.

[100] C. Dai, D. Lin, E. Bertino, and M. Kantarcioglu. An Approach to Evaluate Data Trustworthiness Based on Data Provenance. In *Proceedings of the 5th VLDB*

workshop on Secure Data Management (SDM 2008), SDM, pages 82–98. Springer-Verlag, 2008.

[101] E. M. Maximilien and M. P. Singh. Reputation and endorsement for web services. *ACM SIGecom Exchanges - Chains of commitment*, 3(1):24–31, December 2001.

[102] F. Emekci, O.D. Sahin, D. Agrawal, and A. El Abbadi. A peer-to-peer framework for web service discovery with ranking. In *Proceedings of the IEEE International Conference on Web Services*, ICWS, pages 192–199. IEEE, 2004.

[103] S. Kalepu, S. Krishnaswamy, and S.W. Loke. Reputation= f (user ranking, compliance, verity). In *Proceedings of the IEEE International Conference on Web Services*, ICWS, pages 200–207. IEEE, 2004.

[104] L. Vu, F. Porto, K. Aberer, and M. Hauswirth. An extensible and personalized approach to qos-enabled service discovery. In *Proceedings of the International Database Engineering and Applications Symposium*, IDEAS, pages 37–45, 2007.

[105] Z. Noorian, M. Fleming, and S. Marsh. Preference-oriented QoS-based service discovery with dynamic trust and reputation management. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC, pages 2014–2021. ACM, 2012.

[106] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158 –176, January 2013.

[107] J. Bock, P. Haase, Q. Ji, and R. Volz. Benchmarking owl reasoners. In *Proceedings of Workshop on Advancing Reasoning on the Web*, ARea, 2008.

[108] P. Mell and T. Grance. The NIST definition of cloud computing (draft). *NIST special publication*, 800:145, 2011.