Ivan Luiz Salvadori

DATA LINKING AS A SERVICE: A MICROSERVICES INFRASTRUCTURE FOR PUBLISHING LINKED DATA

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para obtenção do Grau de Doutor em Ciência da Computação.
Orientador: Prof. Dr. Frank Augusto Siqueira

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Salvadori, Ivan Luiz
   Data Linking as a Service: a microservices infrastructure for publishing linked data / Ivan Luiz Salvadori ; orientador, Frank Augusto Siqueira, 2019.
   177 p.

   Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Ciência da Computação, Florianópolis, 2019.

   Inclui referências.

   1. Ciência da Computação. 2. Microserviços. 3. Dados Conectados. 4. Web Semântica. I. Siqueira, Frank Augusto . II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Ivan Luiz Salvadori

DATA LINKING AS A SERVICE: A MICROSERVICES INFRASTRUCTURE FOR PUBLISHING LINKED DATA

Esta Tese foi julgada aprovada para a obtenção do Título de "Doutor em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 19 de Março 2019.

Prof. Dr. José Leomar Todesco

Prof. Dr. Nabor das Chagas Mendonça

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior- Brasil (CAPES) - Código de Financiamento 001 e PDSE 88881.131816/2016-01.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and PDSE 88881.131816/2016-01.

RESUMO

Empresas, governos e pessoas comuns produzem e publicam uma enorme quantidade de dados na Web. Muitos observadores estão apontando que a compreensão dos dados não pode ser feita sem ferramentas adequadas para conceituar, preparar e integrar dados. Pesquisas realizadas nos últimos anos mostram que descrever os dados semanticamente é crucial para promover soluções de integração de dados. No entanto, existem poucos padrões definidos para o desenvolvimento de serviços orientados a dados que possuem suporte a padrões da Web semântica. Nesse sentido, este trabalho propõe DLaaS, uma infraestrutura de microserviços para publicação de dados conectados, capaz de interconectar informações de várias fontes de dados. A infraestrutura proposta é composta por componentes internos capazes de conectar proativamente dados gerenciados por microserviços distintos. DLaaS busca facilitar a execução dos processos necessários para publicar adequadamente dados conectados na Web, o que inclui enriquecimento semântico, conversão de dados legados em dados conectados, processos de interlinking de dados e publicação. Seu principal objetivo é maximizar a reutilização de dados através da conexão de entidades provenientes de conjuntos de dados distintos e heterogêneos, mas que possuem um certo nível de interseção de dados ou possuem relacionamentos semânticos explicitamente definidos. O que diferencia este trabalho das demais propostas encontradas na literatura é a capacidade da infraestrutura executar a otimização da estrutura de dados e gerar links entre recursos Web e, portanto, fornecer uma visão navegável de várias fontes de dados heterogêneas e distribuídas. Além disso, este trabalho propõem um modelo de capacidade que auxilia o desenvolvimento de provedores de dados semânticos. A fim de demonstrar a viabilidade da abordagem proposta, avaliações foram conduzidas de acordo com métodos estatísticos apropriados utilizando conjuntos de dados do mundo real.
Palavras-chave: Microserviços, Dados Conectados, Web Semântica

RESUMO EXPANDIDO

Introdução

Empresas, governos e pessoas comuns produzem e publicam uma enorme quantidade de dados na Web. Muitos observadores estão apontando que a compreensão dos dados não pode ser feita sem ferramentas adequadas para conceituar, preparar e integrar dados. Pesquisas recentes mostram que descrever os dados semanticamente é crucial para promover soluções de integração de dados. No entanto, existem poucos padrões definidos para o desenvolvimento de serviços orientados a dados que possuem suporte a padrões da Web semântica. Este trabalho apresenta três abordagens para composição de serviços que podem ser adotadas para integração de dados. A primeira abordagem consiste na otimização da estrutura dos dados gerenciados pelos provedores de dados. A segunda abordagem explora as definições semânticas previamente definidas para criar links entre dados gerenciados por múltiplos microserviços. E por fim, a terceira abordagem de composição de serviços substitui valores literais associados a propriedades das informações por URLs de recursos Web, tornando as informações muito mais interligadas, significativas e que agregam mais valor aos consumidores. As abordagens de composição são implementadas através de uma infraestrutura de microserviços modelados exclusivamente como provedores de dados semânticos.

Objetivos

Este trabalho tem como objetivo facilitar os processos de integração e publicação de dados na Web. O processo de integração de dados envolve, entre outras atividades, a combinação de diferentes fontes de dados distintas a fim de criar uma visão compartilhada e complementar de seus conteúdos. Dados podem ser publicados na Web de diferentes formas. Uma das formas mais utilizadas atualmente para publicação de dados na Web é através de Web Services, direcionando o desafio de integração de dados para composição de serviços. É amplamente aceito que a adoção de técnicas de Web Semântica é um fator-chave para permitir descoberta, seleção e composição automáticas de Web Services. No entanto, não está claro como aplicar essas técnicas para implementar Web Services. Sendo assim, este trabalho tem como objetivo específico, propor um modelo de capacidade que descreve como adotar adequadamente as técnicas de Web Semântica mais apropriadas

para implementar provedores de dados semânticos. Além disso, este trabalho tem como objetivo possibilitar que recursos Web gerenciados por múltiplos provedores de dados sejam conectados. Dessa forma, os provedores de dados passam a ser capazes de realizar o processo de integração, e deixam de ser simples gerenciadores de dados previamente conectados.

Metodologia
Este trabalho adota a metodologia construtiva proposta por Kasanen, Lukka & Siitonen (1993), também conhecida como *design research*. Esta metodologia busca solucionar problemas relevantes por meio de modelos, artefatos concretos, diagramas, planos, etc., ao invés de produzir conhecimento puramente teórico. Segundo Brandt & Binder (2007), a execução de experimentos é fundamental para *design research*. Nesse sentido, Bang et al. (2012) propõem um modelo para *design research* onde a execução de experimentos assume a posição central. Neste modelo, o processo experimental é realizado a partir de uma motivação bem definida, cujo resultado auxilia a definição de hipóteses, questões de pesquisa e novas motivações. Inspirado no modelo proposto por Bang et al. (2012), este trabalho adota uma abordagem iterativa e incremental, onde as etapas que constituem a metodologia construtiva são divididas em iterações. Cada iteração pode abranger tarefas que buscam resultados parciais de algumas etapas descritas pela metodologia construtiva. Este trabalho é conduzido de acordo com uma série de iterações elaboradas nos termos anteriormente apresentados, e que através da execução de experimentos, resultam em artefatos de diversas naturezas como protótipos, ferramentas, revisões da literatura, entre outros.

Resultados e Discussão
Os experimentos realizados neste trabalho podem ser divididos em dois grupos. Os experimentos que constituem o primeiro grupo avaliam o processo de otimização da estrutura dos recursos Web gerenciados por um único microserviço. Além disso, experimentos são conduzidos para avaliar o mecanismo proposto para realização de inferência de dados. Os resultados dos experimentos mostraram a eficiência do processo de otimização proposto, através do qual padrões de dados foram reconhecidos em conjuntos de dados reais, resultando em uma reutilização significativa dos dados. Além disso, a avaliação mostrou que o mecanismo de regras disponibilizado pelo Apache Jena não é adequado para realizar inferências, mesmo para um pequeno conjunto

de axiomas. Ao reescrever as consultas SPARQL, o mecanismo de inferência proposto reduziu drasticamente o tempo de execução. Os experimentos do segundo grupo avaliam o processo de interconexão de recursos Web gerenciados por múltiplos microserviços. Experimentos mostraram que o processo de criação de links entre recursos Web gerenciados por múltiplos microserviços não influenciou significativamente o tempo de resposta ao cliente. Entretanto, o processo de alinhamento de ontologias foi responsável por um aumento significativo no processo de criação de links. Este efeito pode ser reduzido através da adoção de mecanismos de armazenamento temporário de informação e controle do número máximo de instâncias utilizadas como parâmetro de entrada para os algoritmos de alinhamento. Experimentos que avaliaram o processo de substituição de valores literais por recursos Web mostraram que a materialização resultante torna-se significativamente maior que o conjunto de dados original. No entanto, após a conversão de valores literais em recursos, obtém-se um grafo mais conectado, que segue mais adequadamente os princípios de dados conectados. Como consequência, a informação disponibilizada pelos provedores de dados proporcionam uma visão mais explorável dos dados, possibilitando aos seus consumidores navegar entre os recursos Web, e assim, encontrar mais informações relacionadas. Por fim, resultados dos experimentos realizados acerca da adoção de sistemas multi-agentes mostraram que o uso de agentes é uma solução de comunicação adequada para microsserviços. Ao adotar sistemas multi-agentes, as implementações de serviços podem focar exclusivamente nos requisitos funcionais, enquanto delegam a comunicação aos agentes.

Considerações Finais
Este trabalho propõem DLaaS, uma infraestrutura de microsserviços para publicação de dados conectados. DLaaS é capaz de interconectar recursos Web de fontes de dados distintas e publicá-los de acordo com padrões definidos pela Web Semântica. A infraestrutura proposta é composta por vários componentes internos responsáveis por realizar uma infinidade de tarefas para conectar proativamente os recursos da Web gerenciados. As contribuições resultantes desta tese são: (1) Um modelo de capacidade que descreve como adotar adequadamente técnicas da Web Semântica mais apropriadas para implementar provedores de dados conectados. (2) Um serviço especializado capaz de converter dados não semânticos em dados conectados. (3) Três abordagens de composição de Web Services orientados a dados para interligar recursos Web. (4) Uma infraestrutura de microsserviços para publicação de da-

dos conectados, que visa facilitar a execução dos processos necessários para publicar adequadamente dados conectados de alta qualidade, o que inclui enriquecimento semântico, interconexão de dados e publicação. Quanto às limitações deste trabalho, destacamos limitações quanto à otimização de dados e quanto à capacidade semântica alcançada pelo DLaaS. A implementação atual adota o algoritmo A-Close para geração de regras de associação. Entretanto, uma variedade de outros algoritmos com finalidade semelhante estão disponíveis e podem resultar em melhores resultados, dependendo das características do conjunto de dados a ser otimizado. DLaaS não suporta nenhum nível de capacidade semântica das dimensões de criação e fusão de informações. Em vez disso, DLaaS prioriza características como otimização da estrutura de recursos Web, gerenciamento de conhecimento, representação e modelos de dados. No entanto, trabalhos futuros podem considerar as demais dimensões do modelo de capacidade proposto para tornar o DLaaS ainda mais eficaz.

Palavras-chave: Microserviços, Dados Conectados, Web Semântica.

ABSTRACT

We are living in the age of big data, advanced analytics, and data science. Companies, government, and even ordinary people are producing and publishing a huge amount of data on the Web. Many observers are pointing out that making sense of data cannot be done without suitable tools for conceptualizing, preparing, and integrating data. Research in the last years has shown that taking into account the semantics of data is crucial for fostering data integration solutions. However, there is a lack of solutions for data publishing that follow the best practices for exposing, sharing and connecting data. With this regard, this work proposed DLaaS, a microservices infrastructure for publishing linked data. DLaaS is capable of interconnecting Web resources from multiple data sources. The proposed infrastructure is composed of several internal components responsible for performing a multitude of tasks for pro-actively connecting Web resources managed by the infrastructure. The proposed infrastructure aims at facilitating the execution of necessary processes to properly publish high quality linked data, which includes semantic enrichment, conversion of legacy data into linked data, data linking procedures, and publication. Its main goal is to improve the reuse of data by connecting entities based on distinct and heterogeneous datasets that share a certain level of data intersection or semantic relationship. What differentiates this work from similar proposals found in the literature is the capability of performing data structure optimization combined with the generation of links between Web resources and therefore providing a navigable view of multiple distributed heterogeneous data sources. Additionally, this work proposed a capacity model that describes how to properly adopt the most appropriate semantic Web features for implementing data-driven services. In order to properly demonstrate the feasability of our approach, evaluations were conducted according to appropriate statistical methods and used real-world datasets.
Keywords: Microservices, Linked Data, Semantic Web

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

REST	Representational State Transfer	29
XML	eXtensible Markup Language	30
CSV	Comma-Separated Values	30
DLaaS	Data Linking as a Service	33
RDF	Resource Description Framework	41
OWL	Web Ontology Language	42
URI	Universal Resource Identifier	42
SOA	Service-Oriented Architecture	43
API	pplication Programming Interface	43
URL	Universal Resource Locator	44
SOAP	Simple Object Access Protocol	45
MAS	Multi-Agent Systems	49
FIPA	Foundation for Intelligent Physical Agents	49
ACL	Agent Communication Language	49
JADE	Java Agent DEvelopment Framework	49
SLR	Systematic Literature Review	51
RQ	Research Question	51
RPC	Remote Procedure Call	54
WSMO	Web Service Modeling Ontology	56
WSDL	eWeb Services Description Language	56
WADL	Web Application Description Language	56
JSON-LD	JJavaScript Object Notation for Linked Data	56
SA-REST	Semantic Annotations of Web Resources	56
sdd-μs	Semantic Data-driven Microservice	72
L2R	Literal to Resource	73
AWS	Amazon Web Services	74
EC2	Amazon Elastic Compute Cloud	74
S3	Amazon Simple Storage Service	74
ENCM	Entity Number Control Mechanism	108
MTP	Message Transport Protocol	119

<center>**CONTENTS**</center>

1 INTRODUCTION

Information – as well as the knowledge that may be extracted from it – has become one of the main assets in business, scientific, industrial or governmental areas (JU, 2006; BAGOZI et al., 2017). Organizations are attempting to increase automation and interoperability by publishing data on the Web (ALONSO et al., 2008). Several datasets from a huge variety of sources are publicly available on the Web. Governments are committed to transparency when making available data regarding their activities. A multitude of datasets has been made available by public departments. For example, public security agencies publish data about crime rates and wanted people. Courts of law and Tribunals publish data about processes and sentences regarding a great variety of cases. Despite the fact that a significant number of datasets is available on the Web, they are not connected with one another. Far from the vision of Web of connected data (BERNERS-LEE, 2010), the current data publishing model results in the increase of isolated data silos.

According to Postman (1999), while there is an abundance of information, there is a shortage of knowledge. By combining data from multiples datasets a more complete and unified vision about the data can be achieved. For example, data regarding crime occurrences may be combined with geographic data to create a map of crime, useful for making decisions about public safety. This complementary vision represents a new insight that may be extracted from isolated datases, leveraging knowledge discovery processes. High-level information services can transform data and information into knowledge. These new emerging technologies refer to a variety of factors, architectures, standards, technologies, and models that make knowledge services possible. Knowledge services can be described as a means of exploiting and processing considerable masses of information resources, mapping them into useful knowledge that shows both the content and structure of that knowledge in which users can navigate and unravel them (JU, 2006). In order to create such data navigability, it is necessary to develop mechanisms capable of performing data analysis that result in the automatic generation of links. Therefore, augmenting the information retrieval in terms of maximizing the reuse of data (MARJIT et al., 2013). These publishing initiatives have to deal with many issues, such as the heterogeneity of information systems, the multitude of data presentation formats and access protocols (BANOUAR; RAGHAY, 2015).

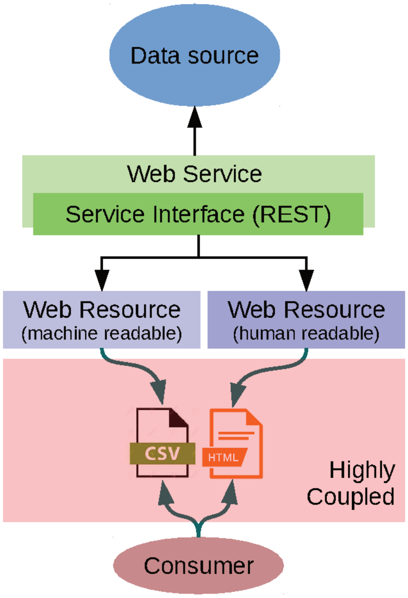


Figure 1 Web service as a data provider

According to Barhamgi & Benslimane (2009), modern enterprises are implementing Web services to expose/share data on the Web. This type of services is called data-providing or data-driven Web Services. In a broad sense, data-driven Web services can be seen as services specialized in providing and managing data. Data-driven Web services may be implemented by following RESTful principles (FIELDING, 2000), which provides a collection of best practices for modeling the information of a given system in terms of resources and their state management. Figure 1 shows an overview of a data-driven Web service used to expose a data source on the Web. In this example, a Web service exposes a data source through Web resources represented in a machine (CSV, JSON, XML, etc) and/or human readable formats (HTML documents). However, this model requires the consumer to know in advance implementation details to interact with the Web service. Therefore, consumers are highly coupled to Web service implementations. Eventual changes on a Web service implementation would make consumers out of date, which requires the development of a new version to re-establish the integration.

Web services may incorporate Semantic Web technologies, resulting in semantic Web services. According to McIlraith, Son & Zeng (2001), semantic Web services should provide information about their services, properties, execution interfaces, pre- and post-conditions, in a machine-readable format. Regarding RESTful Web services, the managed resources and their relationships must be semantically described in order to facilitate discovery, selection and invocation processes (MCILRAITH; SON; ZENG, 2001; MARTIN-FLATIN; LöWE, 2007; ISLAM; ABBASI; SHAIKH, 2010). This approach reduces the coupling between Web services and consumers, since it provides the means to consumers to interpret the data as well as communication details. As a result of the adoption of semantic Web techniques, the coupling between services and their consumers reduces from implementation contracts to conceptual definitions.

By exposing data on the Web through data-driven Web services, data integration from multiples data sources implies in composing RESTful Web services. RESTful service composition is a new challenging research area lacking standardized solutions (GARRIGA et al., 2016). Researchers and practitioners are asking for patterns that properly address RESTful Web service compositions. Based on these aforementioned statements, this thesis presents the following hypothesis:

Hyphotesis: Semantic Web and Linked Data offer all the necessary principles for data services integration.

In summary, Semantic Web is a collection of best practices for representing and publishing data on the Web. Linked Data is simply about using the Web to create links between data from different sources (BIZER; HEATH; BERNERS-LEE, 2009). It refers to a set of best practices for publishing and interlinking the structured data on the Web of data. In order to turn a set of unconnected data silos into the Web of data, distributed datasets must be linked to one another. Many techniques were developed to integrate data based on Linked Data principles. Ferrara, Nikolov & Scharffe (2011) proposed the term data linking to name the problem of finding equivalent resources on the Web of linked data. As previously mentioned, an important outcome of data linking is data reuse. Fragments of the same information may be present in multiple data sources. Thus, connecting these distributed fragments allows the information to be made available in a more useful fashion. In addition, certain information can be interpreted in different ways depending on a given domain of discourse. A given piece of information may be represented in different contexts by different data sources – a vehicle, for example, may be in the sales record of a car dealer, may be registered as a stolen vehicle by the police, and may also be in the records of an insurance company.

In order to make services available in a more cohesive and less coupled way, microservices are replacing traditional implementations known as monolithic applications. Microservices are designed to provide only a small subset of features regarding a well-defined business context (NEWMAN, 2015; ZIMMERMANN, 2016). The result is the fragmentation of data and functionalities into several components that communicate and operate together. In order to properly connect Web resources it is necessary to adopt a suitable service architecture, since the creation of links implies the existence of Web services responsible for responding to such links that now are part of these Web resources. The architecture of microservices contains the necessary characteristics to meet the requirements for publishing data on the Web through Web services implemented as data providers, which include: fine granularity interfaces, state isolation, self-management, low coupling, deployment in lightweight containers, polyglot programming and persistence.

1.1 PROBLEM STATEMENT

Connecting and publishing data on the Web poses several problems to be solved. This thesis mainly addresses the following problems.

Problem 1. Data-driven Web services and consumers are highly coupled. The current development approaches for developing data-driven Web services that publish data on the Web result in technical contracts full of implementation details. This level of coupling jeopardizes the evolution of Web services and frequently makes consumers incompatible with the latest service version.

Problem 2. Despite the fact that Semantic Web and Linked Data could be used as a guide for data integration, it is not clear how to adopt their standards, principles, and techniques for developing semantic data providers.

Problem 3. There is a lack of methodologies and concrete solutions for proactively connecting data from multiple data sources. Most of the proposals found in the literature do not connect data from multiple data sources. Instead, they are limited to manage data previously connected.

1.2 PROPOSED SOLUTION

This work presents DLaaS (Data Linking as a Service), a microservices infrastructure for publishing linked data. The proposed microservices infrastructure consists of a multitude of specific-purpose services that include data storage features, semantic enrichment, inference support, link creation approaches, data intersection analysis, data structure optimization and other functionalities to provide linked data as a service. DLaaS is capable of interconnecting data from multiple sources. Web resources from different datasets are connected through links, which are created based on data intersection. The data intersection is explored in different ways, allowing data to be linked throughout multiple perspectives. Multi-agent systems are also adopted to handle with the dynamic behavior and communication issues.

This work addresses the problem of connecting and publishing linked data from multiple data sources in the context of data service composition. Thus, new composition approaches are proposed along with the tooling support required for their respective implementations. Data services may be seen as services specialized in providing and managing data, acting as a data provider (VACULÍN et al., 2008; PAIK et al., 2017). This type of service only provides specific actions regarding the maintenance of its managed Web resources, without any outcome or effect on external resources or services. The DLaaS infrastructure makes use of semantic data-driven microservices, which are implementations that manage data according to a given domain ontology as well as support further semantic capabilities. An ontology is a formal, explicit specification of a shared conceptualisation (BORST, 1997). As a result, semantic data-driven microservices provide a conceptual layer that states how to interpret their managed Web resources.

Several research papers have been published describing strategies to address service composition problems. However, there are just a few that specifically address the composition of data-driven services. Most of them address the problem of service composition by assuming only implementations whose actions go beyond the scope of data lifecycle management. In addition, only few papers specifically consider microservice composition approaches. Several papers deal with the problem of connecting Web resources from multiple data sources and their availability through Web services. However, these works do not relate this problem to the context of service-oriented computing. Furthermore, research papers that address the adoption of the microservices architecture can also be found in the literature. The great majority of them, though, are focused on how to split out a monolithic application into several microservices or on non-functional features, such as deployment, load balance, testing, among others.

1.3 OBJECTIVES AND CONTRIBUTIONS

This thesis aims to develop a linked data publishing solution capable of automatically connecting Web resources from different data sources in order to maximize data reuse. Based on this main goal, the following contributions are expected:

- Evaluate how semantic Web technologies may be adopted in the context of service-oriented computing;
- Develop tools for automatic creation of links between Web resources managed by semantic data-driven microservices;
- Develop mechanisms to support the integration of data-driven microservices semantically described by heterogeneous ontologies;
- Allow the automatic generation of data-driven microservices and their respective allocation of computational resources;
- Maximize the reuse of data from multiple data sources through data linking approaches.
- Test the feasibility of the microservices architecture to implement data services.
- Reduce the coupling in the inter-service communication process of microservices.

1.4 METHODOLOGY

This work adopts the constructive approach proposed by Kasanen, Lukka & Siitonen (1993). Originally designed to be applied in Management Accounting Research, this approach aims at solving relevant problems through models, concrete artifacts, diagrams, plans, etc., rather than producing purely theoretical knowledge. The constructive approach can be divided into the following steps:

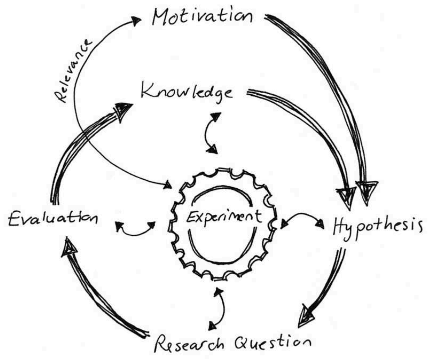


Figure 2 – Activity model for constructive design research proposed by Bang et al. (2012)

1. Find out a relevant research problem;
2. Obtain understanding about the topic;
3. Innovate by constructing a solution idea;
4. Demonstrate the feasibility of the proposed solution;
5. Show theoretical connections and contributions of the solution;
6. Examine the scope of applicability of the proposed solution.

According to Brandt & Binder (2007), executing experiments is pivotal for constructive design research. Therefore, Bang et al. (2012) propose a model for research design where the execution of experiments assumes the central role, as can be seen in Figure 2. In this model, the experimental process is carried out from a well-defined motivation in which results help to define hypotheses, research questions and further motivations.

Based on the activity model proposed by Bang et al. (2012), this work adopts an iterative and incremental approach, where the aforementioned constructive approach steps are repeatedly executed in form of iterations. Each iteration encompasses tasks that aim at finding partial results regarding constructive methodology steps. The adoption of such an iterative and incremental approach allows to adjust the research as soon as a given iteration is finished, which turns the research process more flexible and open to reuse the just obtained results. By combining the results of successive iterations, the requirements described by the constructive design research are expected to be fulfilled. The following items describe the iterations executed throughout this work.

- **Iteration 1**
  - Motivation: Evaluate the benefits and cost of combining data from multiple data sources;
  - Experiment: A platform for enriching police reports with additional information, such as data about criminal cases and geolocation;
  - Outcome: Partial results were obtained for steps 1, 3, 4 and 6 of the constructive design research. This iteration resulted in the following publication: *"The Platform to Enrich, Expand and Publish Linked Data of Police Reports"* presented at the 15th International Conference WWW/Internet (ICWI) (OLIVEIRA et al., 2016).
- **Iteration 2**
  - Motivation: Evaluate the feasibility of adopting the microservices architecture to perform complex business processes.
  - Experiment: An agent-based architecture for process-oriented microservice composition;
  - Outcome: Partial results were obtained for all constructive design research steps. This iteration has shown evidence that it is important to differentiate service-oriented composition approaches from those originally designed to deal exclusively with data management lifecycle. Further understanding of workflow approaches as well as valuable details on adopting multi-agent systems for supporting Web services composition were also obtained as a result of this iteration. This iteration resulted in the following publication: *"An Agent-based Composition Model for Semantic Microservices"*, presented at the 15th International Conference WWW/Internet (ICWI) (SALVADORI; HUF; SIQUEIRA, 2016).

- **Iteration 3**
  - Motivation: Connect Web resources managed by multiple data-driven microservices;
  - Experiment: A composition approach based on Data Linking for data-driven microservices.
  - Outcome: Partial results were obtained for all constructive design research steps. Further understanding about Data Linking, semantic Web and linked data were obtained as a result of this iteration. It was also identified that the adoption of the microservices architecture collaborates to an environment of heterogeneous ontologies. A composition approach and its implementation, realized by a specific-purpose service, were developed to perform data-driven microservices composition. This iteration resulted in the following publication: *"Publishing Linked Data Through Semantic Microservices Composition"*, presented at the 18th International Conference on Information Integration and Web-based Applications and Services (iiWAS) (SALVADORI et al., 2016).
- **Iteration 4**
  - Motivation: Connect Web resources described by heterogeneous ontologies;
  - Experiment: A framework for aligning heterogeneous ontologies used to describe Web resources managed by a microservices architecture;
  - Outcome: Partial results were obtained for all constructive design research steps. This iteration resulted in a more comprehensive understanding of the adoption of ontology alignment and semantic Web technologies to implement semantic microservices. Additionally, this iteration resulted in a collection of tools responsible for aligning ontologies used to describes multiples microservices in a heterogeneous scenario. This iteration resulted in the following publications: *"Improving Entity Linking with Ontology Alignment for Semantic Microservices Composition"*, published by the International Journal of Web Information Systems (IJWIS) (SALVADORI et al., 2017b); and *"An Ontology Alignment Framework for Data-driven Microservices"*, presented at the 19th

International Conference on Information Integration and Web-based Applications and Services (iiWAS) (SALVADORI et al., 2017a)

- **Iteration 5**
  - Motivation: Obtain the state of the art on the adoption of Semantic Web technologies in the context of Data-Driven Web Services;
  - Experiment: Literature Review;
  - Outcome: A Systematic Literature Review was conducted to identify how semantic Web technologies are being applied to implement Semantic Web services as well as to identify approaches that consider the differences between process-oriented and data-driven implementations. Additionally, a capacity model was developed to represent the different compliance levels regarding the adoption of semantic Web technologies in the context of data-driven Web Services. The results of this iteration will be part of a journal article that will be submitted in the coming months.
- **Iteration 6**
  - Motivation: Develop a mechanism for abstracting the process of developing microservices and connecting Web resources.
  - Experiment: sdd-µs: A generic semantic data-driven microservice
  - Outcome: The development of a semantic data-driven microservice capable of converting legacy data into semantic representations and publishing them according to Linked Open Data principles. Furthermore, sdd-µs is capable of optimizing its managed resources in order to maximize the potential data reuse. This iteration resulted in the following publication: *"Semantic Data-Driven Microservices"*, accepted for publication by the EEE Computer Society Signature Conference on Computers, Software and Applications. COMPSAC ´19.
- **Iteration 7**
  - Motivation: Develop a mechanism for managing multiple data-providers.

  - Experiment: DLaaS infrastructure;
  - Outcome: A platform for the automatic generation of microservices and their respective computing resource allocation. By using this infrastructure, users only submit their data to the infrastructure, which in turn connects Web resources from all managed data sources. The results of this iteration will be part of a journal article to be submitted in the coming months.

- **Iteration 8**
  - Motivation: Be able to deal with runtime changes.
  - Experiment: DLaaS Agents;
  - Outcome: The developing of specialized agents for executing tasks that manage the changes in the microservice infrastructure, such as the creation of new microservices at execution time, and changes in conceptual definitions (ontology change). Additionally, those agents are responsible for message exchange between the infrastructure components This iteration was important to allow the infrastructure to be resilient enough to deal with the microservice architectural constraints as well as the dynamic nature of data providers. The results of this iteration will be part of a journal article to be submitted.

1.5 THESIS OUTLINE

This work consists of the following chapters:

- **Chapter 1 - Introduction:** presents the problem addressed by this work, motivations, objectives and the research methodology.
- **Chapter 2 - Background:** presents the main concepts required for the understanding of this work.
- **Chapter 3 - State of the Art:** describes relevant research works and presents a comparative analysis. Additionally, this chapter presents a capacity model for Semantic Data Providers.
- **Chapter 4 - Data Linking as a Service:** presents an overview of the DLaaS infrastructure. This chapter describes the DLaaS components and proposed data linking approaches for data reuse.

- **Chapter 5 - Semantic Data-driven Microservice:** presents the proposed mechanism for abstracting the process of developing microservices as well as the strategy to maximize data reuse based on resource structure optimization.
- **Chapter 6 - Inter-Service Linking for Semantic Data-driven Microservices:** describes in detail the proposed data linking approaches and their respective tooling support.
- **Chapter 7 - Inter-Service Linking Evaluation:** presents the evaluation regarding the proposed composition approach and its respective tooling support.
- **Chapter 8 - Conclusions:** provides conclusive remarks and suggestions for future works in this research field.

2 BACKGROUND

This chapter presents the necessary concepts for the understanding of this work. It starts describing the concepts of Semantic Web and Linked Data, followed by the concepts of Data-driven Web services, microservices, data linking, ontology alignment and multi-agent systems. These concepts are pivotal to the understanding of the infrastructure proposed in this thesis, since it combines approaches, methods, algorithms and tools of all these areas to interconnect data from multiple and heterogeneous datasets, consequently promoting the data to a higher level of reusability.

2.1 SEMANTIC WEB AND LINKED DATA

According to Bizer, Heath & Berners-Lee (2009), the Web has a great potential to be a global linked data space. In this view, not only documents are made available and linked to other documents, but also their contents. In order to achieve this purpose, data is structured in RDF[1] triples (`subject-predicate-object`), in which the subject represents the feature being described, the predicate represents a characteristic of the subject, and the object is the value assigned to this characteristic. RDF (Resource Description Framework) is a data model for describing the semantics of resources and interconnecting them with other related information. By semantically describing resources, not only humans, but also machines are able to infer the meaning of data published on the Web.

By semantically describing data, every piece of Web content has a well-defined meaning. Semantic Web content should support understandability to the human reader and machine processability at the same time. This requires the ability to precisely and unambiguously specify the meaning. The key to machine processability of content on the Semantic Web is that it should be self-describing. This is achievable partly by producing a common language to specify data and metadata on the Web (ALONSO et al., 2008).

Metadata is generally defined as data or information about data. It may be used to store useful information as well as to describe relationships between individual objects fostering the abstraction of representational details such as the format and organization of data of a given

[1]https://www.w3.org/RDF/

domain knowledge. Domain-specific metadata can be constructed from terms in a domain-specific ontology (ALONSO et al., 2008; GARTNER, 2016). An ontology is a formal, explicit specification of a shared conceptualization (BORST, 1997). According to Gruber (1993), an ontology may be defined as the specification of a representational vocabulary for a shared domain of discourse which may include definitions of classes, relations, functions and other objects. In other words, an ontology is a set of terms of interest in a particular information domain and the relationships among them (ALONSO et al., 2008).

More complex constraints can be defined by adopting the Web Ontology Language (OWL)[2]. OWL is an ontology language that allows specifying classes and properties through description logic with boolean operators as well as the constraints on various properties(SINGH; HUHNS, 2004). Different types of constraints and axioms can be described at both the schema and the data levels. At the data level, OWL may by used to represent class membership as well as relationships between instances such same-as and different-from. At the schema level, OWL supports the description of relationships between classes such as subclass-of, disjointedness or equivalence.

Berners-Lee (2011) created the concept of linked data, which represents a set of best practices for publishing structured data on the Web. It consists of: (i) the use of HTTP URIs to identify and locate resources, (ii) providing useful information from URIs, properly represented in a standard model, and (iii) adding more links to related resources in order to obtain further information. These principles aim at creating a single global data repository, resulting in a network of connections that forms the foundations of a new Web. However, in order to properly publish data according to these aforementioned best practices, it is necessary to adopt suitable storage/publishing technologies. There are initiatives that address this matter, such as triple stores and SPARQL endpoints as well as semantic Web services techniques.

2.2 DATA-DRIVEN WEB SERVICES

Traditionally, Web services are implemented to provide a logic layer regarding a given domain, which implies executing business actions. Then, through the client's viewpoint, there is no open access to the data used to support the execution of the logic layer. However, this scenario has changed, and many applications are now built to di-

[2]https://www.w3.org/TR/owl-features

rectly interact with data provided by data-driven Web services, which primarily bring a uniform access to available datasets. According to Barhamgi & Benslimane (2009), modern enterprises are moving from a service-oriented architecture to data-sharing on the Web by exposing databases behind Web Services. The authors call this type of services as data-providing Web Services, where services represent parameterized queries on datasets that may be either relational or may adopt alternative data models, such as key-value stores and graphs.

In a broad sense, data-driven Web services can be seen as services specialized in providing and managing data. Several terms have been used to address these Web services such as data services, data providing services, data as a service, among others. Different definitions are given to these terms in the scientific literature. Carey, Onose & Petropoulos (2012) define data services as a specialization of Web Services capable of providing data by encapsulating a wide range of data-centric operations that can be deployed on top of data stores. Speiser & Harth (2011) have a more restrictive notion, which protects data services from any side effects, taking into account only read-only operations. Vaculín et al. (2008) present a different definition, in which a data providing service encapsulates one or more data sources into a set of Web Service operations. Paik et al. (2017) state that data as a service consists of the provision of uniform access through a standard interface for data consumers, bypassing the business logic layer.

Despite some differences, these concepts have in common the focus on exposing data through a Web interface. They also lead to the separation between the business logic layer and the data access layer as distinct services. As a result, a data-driven Web service can be seen as a particular implementation approach to SOA, where data, instead of business functionalities, plays the central role. There are many service interfaces that may be used for publishing linked datasets on the Web. Data-driven Web services may adopt one or more service interfaces to properly publish linked data according to different users expectations. According to Rietveld et al. (2015), the most adopted service interfaces are: SPARQL endpoint, HTTP Web API and File Dump.

SPARQL endpoint. A SPARQL endpoint[3] allows the execution of SPARQL queries on a dataset through HTTP. The SPARQL query language[4] allows expressing very precise selections of triples in RDF datasets.

[3]http://www.w3.org/TR/sparql11-protocol
[4]http://www.w3.org/TR/sparql11-query

HTTP Web API. Are services that use the semantics of the HTTP protocol. Usually implemented according to REST architectural principles (FIELDING, 2000), HTTP Web APIs are focused on managing Web resources life cycle, which includes managing their internal state and providing representation in multiple data formats.

Data Dump. File-based datasets are conceptually the most simple service interface: the data consists of all triples of the dataset combined into a compressed archive and published at a single URL.

2.3 MICROSERVICES

Microservices are defined as independent small services (NEWMAN, 2015). However, there is no accurate measurement to determine the size and level of independence of a given microservice. According to Newman (2015), each microservice must be developed and maintained by a single development team. This is important to ensure that a microservice is independent of the other services that comprise the application and other development teams. According to Zimmermann (2016), microservices can be seen as a particular implementation approach to build SOA applications, comprising both service development and deployment. Thus, microservices present an evolutionary and complementary strategy to develop SOA applications, adhering to tenets such as fine-grained interfaces, polyglot programming and persistence, and decentralized continuous delivery.

The main focus of microservices is to achieve the excellence of the software requirements regarding a well-defined business domain, which can be divided into several bounded contexts. A bounded context explicitly defines the boundaries of a given application domain through a logical boundary that delimits elements that are closely related. Microservices that implement a single bounded context aggregate elements that change for the same reasons, allowing them to be deployed independently. Due to this, microservices can be developed with the most appropriate technologies for their respective purposes.

In contrast to monolithic applications, which implement all functionalities of multiple domains in a single software artifact, microservices facilitate the maintenance and deployment as well as achieve the desired levels of resilience, since there is no central point of failure. In addition, it is possible to achieve greater scalability levels by replicating only the most demanded microservices. Richards (2015) classifies as functional microservices those ones that implement functional requirements of a given domain, and as infrastructural microservices those that implement non-functional requirements, such as authentication, monitoring, logging, among others.

Microservices may be developed as SOAP[5] Web services, as RESTful Web APIs (FIELDING, 2000) or as standalone systems connected by message brokers that support publisher/subscriber[6] messaging. In general, microservices are deployed using container virtualization technologies, which promote the environment isolation for running, developing and testing a single microservice. According to Matthias & Kane (2015), Docker is one of the most common platforms for deploying microservices. It facilitates the deployment by creating portable and immutable images, which contain a particular version of the microservice. These images can be initialized and executed by different containers on different servers.

Microservices developed as Web services may incorporate Semantic Web technologies, resulting in semantic microservices. According to McIlraith, Son & Zeng (2001), semantic Web services should provide information about their services, properties, execution interfaces, pre- and post-conditions, in a machine-readable format. Regarding RESTful Web APIs, the managed resources and their relationships must be semantically described in order to facilitate discovery, selection and invocation processes (MCILRAITH; SON; ZENG, 2001; MARTIN-FLATIN; LöWE, 2007; ISLAM; ABBASI; SHAIKH, 2010). Battle & Benson (2008) further restrict the concept of semantic Web services, requiring the adoption of standards such as RDF and SPARQL. Although the REST API Web developers community is still discussing the advantages and disadvantages of the service description, Alarcon & Wilde (2010) take a more pragmatic view and consider describing REST services as a positive mechanism for managing Web APIs.

Assuming that microservices implement only simple tasks, the execution of more complex tasks, which return results that add greater value to consumers, requires the composition of microservices. According to McIlraith, Son & Zeng (2001), the automatic composition of Web services requires the processes of selection, composition and interoperability of services to be performed automatically, based on a high-level description of a complex task that can not be completely performed by a single Web service. The same can be said about interlinking data published by multiple data-driven Web services in which such complex task is restricted to managing data life-cycle.

[5]`https://www.w3.org/standards/techs/soap`
[6]`https://www.w3.org/TR/websub/`

2.4 DATA LINKING

Data linking is the task of finding equivalent resources that represent the same real-world object (FERRARA; NIKOLOV; SCHARFFE, 2011). Data linking can be formalized as an operation that takes collections of data as input and produces a set of binary relations between their entities as output. The problem of data linking can be categorized into two main groups: connection of data from heterogeneous sources; and comparison of data for data cleaning, duplicate detection or merge/purge records.

A key requirement to properly produce link relations between entities is to determine the meaning of the matching. Usually, the matching is intended to link together entities that could be considered the same real world object, often expressed using the *owl:sameAs* property. However, the notion of identity can be interpreted among three different meanings: ontological identity, logical identity and formal identity (FERRARA; NIKOLOV; SCHARFFE, 2011). In the first notion, two different entities with different object descriptions are identified as the same real-world object. In the logical identity, two different entities represent the same object when they can replace each other in a logical expression without changing the meaning of the expression. Finally, the formal identity is used in cases where each entity of the data source can be uniquely identified by a standard property, such as ISBN for books, DOI for academic papers, email for user accounts, etc.

The problem of data linking is similar to database record linkage and also ontology schema matching, both widely explored in the literature (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007; EUZENAT; SHVAIKO, 2007; KöPCKE; RAHM, 2010). Data linking makes use of techniques from these areas, which can be divided into three main categories: value matching, individual matching and dataset matching. The value matching technique applies to linking entities that contain the same property value expressed in different ways. The individual matching technique is used for deciding whether two entities correspond to the same real-world object by analyzing their property values. Dataset matching takes into account all entities from two different data sources in order to create an optimal alignment between them.

2.5 ONTOLOGY ALIGNMENT

The adoption of microservices brings the idea of loose coupling and independent maintenance, which can lead to the design of heterogeneous ontologies describing the same domain. The OWL vocabulary, since its inception, includes predicates to denote equivalence of individuals, classes and properties. However, OWL equivalence vocabulary is not always used by ontology designers. This becomes a problem when software agents are confronted with data described by a different ontology, and are unable to use it because its semantics may not be understood by the agent even after fetching the ontology. The automatic detection of equivalence relations between heterogeneous ontologies, known as ontology matching (EUZENAT; SHVAIKO, 2007), aims to solve this problem and is an active research topic.

The matching operation yields an alignment A_1 between two ontologies O_1 and O_2 (PAVEL; EUZENAT, 2013). To construct the alignment, the following inputs may be used in addition to the ontologies: i) a known alignment A_0; ii) matching parameters (such as weights or thresholds); and iii) external resources. The alignment contains a set of correspondences between entities and properties of such ontologies. Each correspondence denotes a relation of equivalence, generalization or disjointness between two entities of O_1 and O_2 (PAVEL; EUZENAT, 2013). Generally, these correspondences are determined by means of a degree of similarity among the entities of ontologies. Euzenat & Shvaiko (2007) propose methods for assessing this similarity based on specific features of entities. The basic techniques are classified as follows:

- Name-based: considers their inputs as strings. This method seeks for similar elements (classes, individuals, relations) based on their names, labels or comments.

- Structure-based: instead of comparing only the names of entities, the structure of elements found in the ontologies – i.e., their taxonomic structure – is also compared.

- Extensional: analyzes instances of classes. If classes of two ontologies share individuals, the method most likely performs a correct match for these classes.

- Semantic-based technique: is a deductive method that uses the model-theoretic semantics. It often uses a reasoner in order to infer the correspondences.

2.6 MULTI-AGENT SYSTEMS

The growing complexity of distributed systems in current dynamic business environments requires more sophisticated methods and technologies to tackle the related emerging issues and requirements. Software agents, an AI-based technology, has demonstrated its potential in dealing with heterogeneous distributed systems (GRIFFITHS; CHAO, 2010). One of the most important application fields of agent-based systems is information management. In particular, the Internet has been shown as an ideal domain due to its intrinsically distributed nature and the sheer volume of information available. Agents can be used, for example, for searching and filtering this mass of information (KLUSCH, 2001).

The concept of an agent has become important in both Artificial Intelligence (AI) and mainstream computer science. Agent theory is concerned with the question of what an agent is, and the use of mathematical formalism for representing and reasoning about the properties of agents. Agent architectures can be thought of as software engineering models of agents(WOOLDRIDGE; JENNINGS, 1995). Perhaps the most general way in which the term agent is used is to denote hardware or software-based computer system that enjoys the following properties:

- **Autonomy:** agents are responsible for their actions and internal state. They independently operate without the intervention of humans or others. (CASTELFRANCHI, 1995).

- **Social ability:** agents interact with other agents (and possibly humans) via some kind of agent-communication language (GENESERETH; KETCHPEL, 1994). Moreover, the possibility of communication with other intelligent agents is the precondition of common action in pursuit of a goal (SIBBEL, 2001).

- **Reactivity:** agents perceive their environment and respond in a timely fashion to changes that occur in it (WOOLDRIDGE; JENNINGS, 1995). Every agent has its own model of the external world that surrounds it (SIBBEL, 2001).

- **Strategies and Decentral Control:** The purpose is to develop individual strategies that individual agents pursue and that ensure that a common goal can be achieved even without central regulation (SIBBEL, 2001).

Although there is no single definition of an agent all definitions agree that an agent is essentially a special software component that has autonomy that provides an interoperable interface to an arbitrary system and/or behaves like a human agent, working for some clients in pursuit of its own agenda. An agent system can be based on a solitary agent working within an environment. However, they consist of multiple agents. These multi-agent systems (MAS) can model complex systems and introduce the possibility of agents having common or conflicting goals.

One of the key components of multi-agent systems is communication. In fact, agents need to be able to communicate with users, with system resources, and with each other, if they need to cooperate, collaborate, negotiate and so on. In particular, agents interact with each other by using some special communication language(BELLIFEMINE; CAIRE; GREENWOOD, 2007). The FIPA-ACL[7] is grounded in speech act theory which states that messages represent actions, or communicative acts – also known as speech acts or performatives. The FIPA-ACL comprises a set of 22 communicative acts where every act is described using both a narrative form and a formal semantics based on modal logic that specifies the effects of sending the message on the mental attitudes of the sender and receiver agents. Examples of the most commonly used acts are: inform, request, agree, not understood, and refuse.

Initially motivated by the need to validate the early FIPA specifications, JADE is the most adopted agent platform that provides basic middleware-layer functionalities which are independent of the specific application and which simplify the development of distributed applications that exploit the software agent abstraction. JADE implements agent abstraction over a well-known object-oriented language, Java, providing a simple and friendly API (BELLIFEMINE; CAIRE; GREENWOOD, 2007).

2.7 FINAL CONSIDERATIONS

This chapter presented the background for the understanding of this work. First, we presented the concepts of semantic Web and linked data, which concerns to publish information on the Web in a structured and interconnected with other information, in order to create a global data space. In the following, we presented the concept of data-driven

[7]http://www.fipa.org/specs/fipa00061/SC00061G.html

Web services, which are services specialized in providing and managing data. Microservices are defined as dedicated services to perform with excellence only one functionality. However, to provide more significant value to consumers, microservices need to be combined. Considering the scope of data-driven microservices, such combination means linking Web resources published by multiple microservices. To address this matter, the concept of data linking was introduced. Moreover, ontology alignment was defined as a process that aims to identify relations of equivalence between classes and properties of heterogeneous ontologies. Finally, multi-agent systems are presented as a suitable solution to deal with the complexity of distributed systems in dynamic business environments.

3 STATE OF THE ART

This chapter presents a Systematic Literature Review according to the methodology proposed by Kitchenham (2004). Section 3.1 presents the review protocol, discussions and conclusions obtained by this review. Then, section 3.2 presents a non-systematic review on data-driven service composition approaches. Finally, section 3.3 proposes a capacity model for semantic data providers.

3.1 SYSTEMATIC LITERATURE REVIEW

A systematic literature review aims at providing means to identify, select, and analyze evidence related to a given research topic. This review aims to obtain the state of the art on description, selection, discovery and service composition. More specifically, it seeks to identify techniques and approaches, along with their characteristics and applications regarding their adoption for developing data-driven and action-based Web services. In addition, it seeks to identify how semantic Web techniques are being used in the context of Web services.

Generally, systematic literature reviews consider primary studies, which include papers with a original proposal. However, due to the large number of existing revisions focused on primary studies, only secondary works were considered, such as surveys and other reviews. Thus, through a meta-review, it is desired to group the existing evidence on the aforementioned topics of research in order to identify gaps and provide directions for new research.

According to the methodology proposed by Kitchenham (2004), a systematic literature review must be reproducible by other researchers, which implies defining a research protocol. Thus, research questions must be aligned as a search strategy, as well as its inclusion and exclusion criteria. Two research questions were defined (RQ1 and RQ2). The motivation of RQ1 is to identify in the literature the existence of specific description, selection, discovery and composition techniques for data-oriented services and for action-oriented services. For RQ2, the motivation is to identify how the semantic Web is used in the context of Web services, and what characteristics are considered to differentiate semantic Web services from traditional Web services.

RQ1: Do the reviews differentiate data-driven services from those ones based on actions?

RQ2: How semantic Web is adopted for developing semantic Web services?

Based on the aforementioned research questions, a search string was defined to find out relevant and potential reviews capable of answering the questions.

```
(("semantic service") OR ("semantic web service") OR
("service composition") OR ("service description") OR
("service discovery") OR ("service selection")) AND
(("survey") OR ("review") OR ("state of the art") OR
("systematic mapping"))
```

Three scientific databases were used to execute the search string: Scopus, IEEE Xplore and ACM DL. In order to select only the most relevant works, five steps were defined to analyze the articles returned by the search string, as shown in Figure 3.

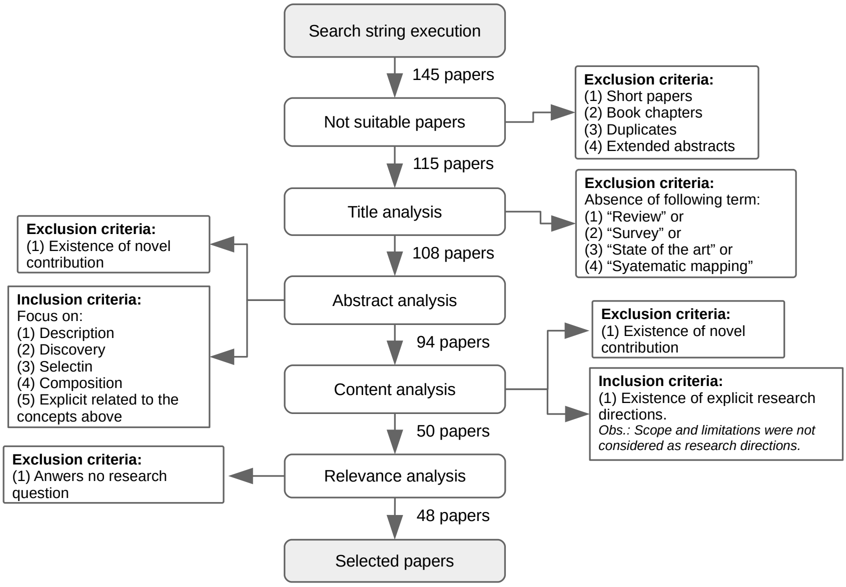


Figure 3 – Systematic Literature Review - steps

Among 48 selected papers, only 13 of them address characteristics that differentiate data-oriented from action-based Web services. Table 1 shows all selected works that answer RQ1. This table shows how the distinction between these two natures of services is made along with the context in which the review considers its application. Only 4 papers explicitly address specific techniques for each nature of service. The rest of them present only implicit evidences of such distinction.

Table 1 – Secondary works that answer RQ1

Research Review	**Distinction**	**Context**
Tosi & Morasca (2015)	Explicit	Semantic annotation
Garriga et al. (2016)	Explicit	Mashup
Lemos, Daniel & Benatallah (2015)	Explicit	Service composition
Wang & Shen (2015)	Explicit	Data-intensive services
Murguzur et al. (2014)	Implicit	Process modeling
Girolami, Chessa & Caruso (2015)	Implicit	Mobile social networks
Kapuruge, Han & Colman (2010)	Implicit	Process flexibility
Bartalos & Bielikova (2011)	Implicit	User goal
Duan, Yan & Vasilakos (2012)	Implicit	Telecom services
Gao, Urban & Ramachandran (2011)	Implicit	Data integrity
Issarny et al. (2011)	Implicit	Heterogeneous services
Nacer & Aissani (2014)	Implicit	XML Web services
Syu & Fanjiang (2013)	Implicit	Workflows

3.1.1 Data-driven and Action-based Web Services

Tosi & Morasca (2015) state that RESTful Web services are becoming popular and being used as an alternative to SOAP Web services. The explicit evidence of the distinction between data-oriented and action-based services can be noted when the authors refer to the adoption of RESTful Web services, where the concept of service is replaced by the concept of resource. This review presents papers that deal with semantic annotation techniques of Web services. However, it reviews workflow techniques in which operations and data are combined with one another and then, connected through links.

According to Garriga et al. (2016), REST Web services should describe details about resources rather than features. Authors characterize action-based services as RPC (Remote Procedure Call) services and data-driven services as resource services. In addition, REST Web service compositions are generally performed through mashups. This work describes mashups in accordance with the definition given by Rosenberg et al. (2008), as an approach to combine and reuse services and data from multiple sources to provide greater value to users. This paper divides mashup techniques into data-oriented approaches and process-oriented approaches. Data-oriented mashups comprise converting, transforming and combining similar data elements into a single high value-added data element. Process-oriented Mashups are more sophisticated and enable the action-based service composition for specific business processes.

According to Lemos, Daniel & Benatallah (2015), there are essentially two types of service compositions: (i) control flow constructs for process-oriented compositions and (ii) data flow constructs for data-driven compositions. Authors define Web services as software components capable of providing functionalities, data or user interface. Regarding data-driven services, this work adopts the Data services terminology originally defined by Carey, Onose & Petropoulos (2012), which defines them as a specialization of traditional Web services, which can be made available as a data access interface that encapsulates operations related to data maintenance. Unlike Garriga et al. (2016), which consider mashup a service composition approach, this work defines it as Web applications that aggregate data and functionality from multiple sources.

Wang & Shen (2015) state that Data-intensive services represent one of the most challenging applications for SOA. These services differ from traditional Web services due to the fact that they manage large amounts of data that require non-trivial composition processes, mainly due to special characteristics of Quality of Service. This type of service is discussed in the context of a service composition that adopts bio-inspired algorithms.

The following works do not make an explicit distinction between data-driven and action-based services. However, one can notice that this distinction is implicitly present. Murguzur et al. (2014), for example, state that a business process implemented by multiple services is defined through the data that is used as input and output parameters between distinct services. This work adopts the concept of data-driven business process as defined in Aalst, Weske & Grünbauer (2005), which associates the definition of activities with at least one data definition. In Girolami, Chessa & Caruso (2015), the concept of Content-based services is addressed regarding mobile social networks where this type of service is designed to share media contents with other devices. Kapuruge, Han & Colman (2010), Syu & Fanjiang (2013) and Issarny et al. (2011) propose data-oriented alternatives to workflows, compatibility models between services and interaction models, respectively. Bartalos & Bielikova (2011) and Duan, Yan & Vasilakos (2012) claim that Web services offer features, which can once and for all provide data.

For Nacer & Aissani (2014), Web services are effective mechanisms for integrating distributed data and applications on the Internet. Finally, Gao, Urban & Ramachandran (2011) state that the term data-oriented applies to services that behave as data sources.

As a primary result, it is clear that the distinction between data-driven and action-based Web services is not the most appropriate method to categorize the state of the art regarding their respective approaches. Although Garriga et al. (2016) uses this terminology to address service-oriented as RPC Web services and services that follow the REST architectural principles as data-driven services, Bartalos & Bielikova (2011) state that the main goal of Web services is to offer features that can make data available that are of interest to users. The concept of data-driven services is well defined and accepted by the scientific community. On the other hand, the concept of action-based services is not so widely accepted. Besides contemplating services with several different characteristics, it seems to be inadequate to characterize specific approaches.

Moreover, the concept of data-driven Web services concerns about how the compositions are performed. In general, data-driven Web services are associated with REST implementations. Such association, according to the selected works, results in mashups. Ordónez et al. (2015) cites that mashups are typically created at design time. According to Duan, Yan & Vasilakos (2012), the lack of standardization restricts the number of REST Web services that can be combined, in addition to making automatic creation of mashups impossible. Garriga et al. (2016) claim that compositions of RESTful Web services can be realized through WS-BPEL. However, the lack of RESTful service description patterns, along with the disuse of XML as the representation format for this type of service implementation, make this alternative unfeasible. The authors note that mashups represent a lighter alternative for data integration, even though they have not reached suitable maturity levels to address security, authorization, and quality of service issues.

3.1.2 Semantic Web Services

This discussion refers to how semantic Web techniques are applied for developing semantic Web services. Almost all works sustain the adoption of semantic techniques as being pivotal to automate service discovery, selection and composition. Thus, we seek to identify how these processes may be automated by adopting semantics, as well as to identify the characteristics that differentiate semantic Web services from traditional Web services.

Considering the 48 selected secondary works, 34 of them address the adoption of semantic Web techniques. However, 20 papers adopt semantics only to describe the service interfaces. Five papers describe the use of semantics to describe the information managed by Web services. Nine papers describe the adoption of semantic for distinct purposes, such as describing user goals and requirements, modeling processes, and so on. Papers that answer RQ2 are presented in Table 2.

Garriga et al. (2016) present a collection of semantic technologies for SOAP and REST Web services. They are divided into three layers: functional description layer, semantic annotation layer and ontology layer. The first layer syntactically describes service interfaces through description patterns such as WSDL, WADL and hREST. The second layer is related to semantic annotation techniques such as: SA-REST (Semantic Annotations of Web Resources) or MicroWSMO (a microformat based on WSMO - Web Service Modeling Ontology), which produce semantically-enriched service descriptions. The third layer is related to representation languages for Web services, such as WSMO-Lite, OWL-S, among others. Mechanisms for semantically describing data are also presented. Among them, stands out JSON-LD, used to annotate resources managed by RESTful Web services. According to the authors, such mechanisms are mandatory to enable the development of data-driven Web services.

According to Tosi & Morasca (2015), information managed by semantic Web services may be interpreted by software agents. The authors cite WSMO as one of the most significant initiatives in the area of semantic web services. The WSMO ontology allows describing features, interaction details, and information exchanged between users and Web services. This review discusses a collection of research papers that cover several aspects regarding the adoption of semantics in Web services, among them: automatic generation of ontologies, automatic creation of Web services and several service description approaches. One of the papers analyzed in this review describes the necessary steps for the development of semantic Web services, which includes: declaration of a domain vocabulary, identification and grouping information, identification of axioms and relationships, application of constraints and validations. The authors conclude that the analyzed work concentrates efforts on the development of new ontologies and tools rather than on concrete implementations, a fact that keeps the discussions at an abstract level.

Table 2 – Secondary works that answer RQ2

Work	**Adoption of semantics for**
Garriga et al. (2016)	Data and service description
Lemos, Daniel & Benatallah (2015)	Service description
Zilci, Slawik & Kupper (2015)	Service description
Tosi & Morasca (2015)	Data and service description
Elsayed & Salah (2015)	Service description
Hang & Zhao (2015)	Service description and user goals
Girolami, Chessa & Caruso (2015)	Service description
Mármol & Kuhnen (2015)	Service description and user goals
Dos Santos Rocha et al (2015).	Service description
Ordónez et al. (2015)	Service description, user goals and mashups
Nazmudeen & Buhari (2015)	Cluster modeling
Jula, Sundararajan & Othman (2014)	Service description
Nacer & Aissani (2014)	Heterogeneous environment
Sun et al. (2014)	Service description and user requisites
Grolinger et al. (2014)	Process modeling
Murguzur et al. (2014)	Service description
Campos, Rosa & Pires (2014)	Service description
Immonen & Pakkala (2014)	Service description
Platenius et al. (2013)	Service description
Syu & Fanjiang (2013)	Data and service description
Dong, Hussain & Chang (2013)	Service description
Leite et al. (2013)	Data and service description
Ngan & Kanagasabai (2013)	Service description
Wang & Wang (2013)	Service description and workflows
Duan, Yan & Vasilakos (2012)	Service description
Sun, Dong & Ashraf (2012)	Service description
Teka, Condori-Fernandez & Sapkota (2012)	Service description
Wu, Chen & Huang (2012)	Service description
Gao, Urban & Ramachandran (2011)	Service description
Issarny et al. (2011)	Service description
Bartalos & Bielikova (2011)	Service description and user goals
Ahmed & Boutaba (2011)	Service description and query processing
Al-Shargabi, Sheikh & Sabri (2010)	Service description
Strunk (2010)	Service description

According to Syu & Fanjiang (2013), the main goal of semantic technologies is to share and reuse information between different application domains and organizations. The authors cite that ontologies may be used for three different purposes. Modeling the knowledge of a particular domain is cited as the most common purpose of using ontologies. The second most common use is the adoption of service ontologies, which provide means for describing Web service interfaces in such a way that is understandable to software agents. Ontologies are rarely used to generate workflows, where specific ontologies are used to describe business processes. The authors conclude that data-driven workflows are the most important aspect for the automatic composition of services, since the data format is the main responsible for the compatibility between services. Therefore, the autors conclude that it is mandatory to semantically describe the data provided by Web services.

According to Nacer & Aissani (2014), knowledge representation is the more significant matter for implementing Semantic Web services, since it requires the adoption of mechanisms to use, share, discover and exchange knowledge between users and software agents in heterogeneous environments. According to the authors, semantic Web services must offer standardized mechanisms for representation, publication and localization of knowledge. The main motivation of this review is to identify problems and solutions related to system interoperability in heterogeneous environments. The authors pointed out that the ontology alignment is the main challenge for interoperability and data integration.

Ngan & Kanagasabai (2013) present a review on semantic Web service discovery approaches. The authors describe the main steps involved in the discovery process, which include semantic description, dissemination, mediation, storage, interpretation of user requests, negotiation and service selection. The authors cite that the service discovery mechanisms must be able to deal with the heterogeneity of platforms, data formats and ontologies for description of services and domains. The authors argue that WSMO is able to address these problems by using mediators.

As a conclusion, the state of the art does not properly define the concept of semantic Web service. However, it describes features that semantic Web services must have to leverage the automatic service composition, consequently facilitating data integration of applications deployed in large-scale and/or heterogeneous environments. It does not seem appropriate to define semantic Web services in a binary way. It does not seem possible to define semantic Web services in a way that evaluating whether a given service conforms with the definition will produce a binary (yes or no) answer. A more appropriate way would be to classify them through compliance levels regarding the adoption of specific semantic technologies. In order to further discuss this matter, session 3.3 proposes a capacity model for semantic data providers.

3.2 DATA-DRIVEN WEB SERVICE COMPOSITION

Several research works available in the literature address the problem of service composition. However, only a few specifically deal with the composition of data-driven services. Most of them address service composition by assuming only implementations whose actions go beyond the scope of entity lifecycle management. This section presents a literature review on original research works that consider approaches, methods, implementation and semantics applied specifically to data-driven Web services, as summarized in Table 3.

Table 3 – Works that address Data-driven Web service composition

Work	**Semantic usage**	**Service approach**
Taheriyan et al. (2012)	KARMA model for service description and RDF data	Web APIs (read only)
Lira et al. (2014)	SERIN for service description and RDF	RESTful Web APIs (read/write)
Mazurek et al. (2014)	Only schemas and metadata	RESTful Web APIs (read/write)
Rietveld et al. (2015)	Endpoints SPARQL and RDF	Endpoint SPARQL and third part Web APIs
Trinh et al. (2015)	KARMA model for service description and JSON-LD	Server Widgets
Xie et al. (2017)	endpoint SPARQL and JSON-LD	Data service APIs (read only)

Taheriyan et al. (2012) propose to adopt semantics for implementing Web APIs through a semi-automatic method by constructing semantic models. In this method, users provide URLs for invoking services along with a vocabulary to create a service model. This information is received by a Web service, which invokes the URLs in order to obtain sample data, and thus creates models to semantically represent Web API functionality. Semantic models are constructed using the KARMA model (GUPTA et al., 2015), which specifies a mapping between sample data from the Web API and a given vocabulary.

Lira et al. (2014) propose an approach to ensure data integrity managed by semantic data services (SDS). The approach makes use of Semantic Restful Interfaces (SERIN) to semantically describe the service interfaces. In addition, one can set data constraints by setting properties such as *NOT NULL*, *UNIQUE*, or even setting attributes used as instance identifiers, equivalent to primary keys in relational databases. As a result, the approach produces a specification for developing Web services designed exclusively for data management.

Mazurek et al. (2014) present an architecture for aggregating, processing and provisioning data managed by Web services. In addition, the CLEPSYDRA framework is presented as an implementation of the proposed architecture, allowing data publishing to be available through different protocols and formats that can be combined in a standard way. The framework is divided into three layers: aggregation, storage and processing. The first layer uses aggregating agents responsible for communicating with data providers and for managing their structures. The second layer is responsible for storing the information in NoSQL databases. Finally, the third layer processes data to perform previously programmed modifications.

Rietveld et al. (2015) present *Linked Data-as-a-Service*, an architecture that allows querying heterogeneous data sources. The architecture consists of multiple data sources, which are continuously updated by intelligent agents. The architecture solves four main issues regarding linked data publishing. The first problem concerns the lack of a single uniform standard for querying data. This problem is solved by defining a uniform query interface. The second problem is related to how the data is made available, and much of the data is available through *dump* files. This problem is addressed by providing an updated data query interfaces. The third problem is related to the lack of mechanisms for implementing Web services capable of making Web-scale data available in a simplified way. This problem is partially solved through a specialized service focused on publishing data. Finally, the fourth problem is associated with the difficulty of manipulating multiple data sources. The workaround for this problem uses metadata to define which data sources are to be used in a given federated query.

Trinh et al. (2015) propose an architecture for collaborative integration of heterogeneous data sources. Linked widgets are the main elements of the proposed architecture, being responsible for data standardization. There are three types of widgets: data, processing, and display. The first type is responsible for collecting data from one or multiple sources. The second type is responsible for combining data by performing enrichment, transformation and aggregation. The third type is responsible for presenting the information. This proposal allows final users to contribute to the development of client widgets, which communicate with linked widgets in order to execute a certain function. Therefore, users should input a semantic KARMA model(GUPTA et al., 2015), an executable function and a visualization interface. The architecture also defines a communication protocol used to facilitate the interaction between widgets.

Xie et al. (2017) present the Transparent Data as a Service (TDaaS) framework for integrating data from heterogeneous sources. The transparency is implemented in three layers: data integration, data fusion and service provisioning. The architecture defines three types of components: Data Service Provider (DSP), Data Service Consumer (DSC) and Data Operation Center (DOC). A DSP represents an organization that has information of interest. DSC is an entity that requests data services to the system. DOC is responsible for maintaining a shared vocabulary, collecting DSP data, and providing data services. As a result, this work provides a platform that allows DSPs to be registered and their data analyzed according to existing relationships. Thus, a DSC interested in a particular type of information may access the DOC and retrieve the desired data from multiple DSPs without knowing technical details about the structure and system organization.

Table 4 – Desirable features for semantic data providers

Work	**Inference Support**	**Legacy to Semantic**	**Ontology Alignment**	**Entity Linking**
Taheriyan et al. (2012)	✗	✓	✗	✗
Lira et al. (2014)	✗	✓	✗	✗
Mazurek et al. (2014)	✗	✓	✗	✗
Rietveld et al. (2015)	✓	✗	✗	✗
Trinh et al. (2015)	✓	✗	✓	✗
Xie et al. (2017)	✓	✗	✗	✗

Table 4 presents a comparison between the selected works that address data-driven service composition approaches regarding desirable features for semantic data providers. It is important to notice that only Trinh et al. (2015) include the ontology alignment process into their proposed architecture. None of these works adopt entity linking procedures for proactively connecting the data managed by their proposed solutions. Instead, they adopt data integration approaches that consist in merge/fusion procedures for providing a unified vision of multiples data sources. However, the main goal of modeling data providers for publishing data on the Web according to linked data principles is connecting data rather than merging information from multiple sources.

3.3 A CAPACITY MODEL FOR SEMANTIC DATA PROVIDERS

There have been several research works that address Semantic Web Services and their capabilities. It is broadly said that the adoption of semantic Web technologies facilitates service discovery, selection, composition and activation. However, it is not completely clear how Web services may adopt such semantic Web techniques. Moreover, it is not even clear the concept of semantic Web Services. Therefore, considering that semantic Web services are traditional implementations that adopts semantic Web technologies is not enough to characterize their attributes and consequently their costs and benefits.

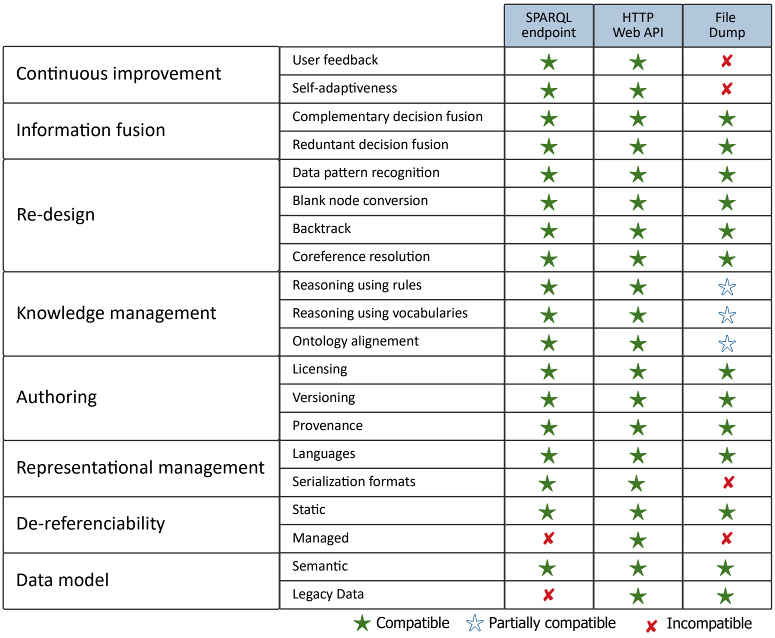
There have also been several research works focused on data quality, conformance and best practices for publishing linked data. Assaf & Senart (2012) present principles to describe the quality of linked data sources. Neumaier et al. (2017) discuss and introduce challenges of integrating openly available Web data as well as discuss data quality issues associated with Open Data on the Web and how Semantic Web techniques and vocabularies can be used in this context. Hogan et al. (2012) and Feitosa et al. (2018) respectively survey linked data conformance and best practices for publishing linked data.

Based on this literature review, this section proposes a capacity model for semantic data providers. The proposed capacity model aims at helping to better characterize the concept of semantic Web services in the context of data providers. More specifically, it discusses the role of several semantic capabilities for publishing linked data. As shown by Figure 4, the capacity model associates data quality and linked data characteristics with the most common service interfaces for publishing linked data. It defines eight dimensions, each one containing semantic capacities. Each of these dimensions, from bottom to top in Figure 4, are described in sections 3.3.1 to 3.3.8. According to Rietveld et al. (2015), SPARQL endpoints, HTTP Web APIs and File Dumps are the most common service interfaces for publishing linked data. Therefore, the proposed capacity model associates semantic features with these service interfaces in order to establish their compatibility.

3.3.1 Data Model

This dimension considers the way in which the data is made available to data providers. More specifically, it is considered how a given data source is described and organized. Depending on the data source's attributes, the data provider would have a distinct semantic capacity. Two alternatives are considered in this dimension: legacy data and semantically described data.

Legacy Data. Data sources without semantic description are considered legacy data. Data sources classified as legacy data maintain their original structure and organization. Examples of legacy data include relational databases, XML or JSON files, among others. However, in order to provide a semantic representation, it is required a mapping mechanism that associates data attributes with terms defined in a domain ontology. Generally, this association is performed when a query is being answered. However, inferences, as well as further seman-



		SPARQL endpoint	HTTP Web API	File Dump
Continuous improvement	User feedback	★	★	✘
	Self-adaptiveness	★	★	✘
Information fusion	Complementary decision fusion	★	★	★
	Reduntant decision fusion	★	★	★
Re-design	Data pattern recognition	★	★	★
	Blank node conversion	★	★	★
	Backtrack	★	★	★
	Coreference resolution	★	★	★
Knowledge management	Reasoning using rules	★	★	☆
	Reasoning using vocabularies	★	★	☆
	Ontology alignement	★	★	☆
Authoring	Licensing	★	★	★
	Versioning	★	★	★
	Provenance	★	★	★
Representational management	Languages	★	★	★
	Serialization formats	★	★	✘
De-referenciability	Static	★	★	★
	Managed	✘	★	✘
Data model	Semantic	★	★	★
	Legacy Data	✘	★	★

★ Compatible ☆ Partially compatible ✘ Incompatible

Figure 4 – Capacity Model for Semantic Data Providers

tic operations on the data, are drastically limited, since the data are not properly organized according to standards imposed by the semantic Web. Legacy data sources are incompatible with SPARQL endpoints since this type of service interface is only able to manage semantically described data.

Semantic Data. Data sources with this characteristic are available to data providers as semantic sources. Examples of semantic sources include triple stores or files in which data is represented according to a given standard such as RDF/XML, Turtle, JSON-LD, among others. A data provider may be able to convert legacy data into a semantic data source. In this scenario, the legacy data it is semantically enriched according to a predefined mapping and then materialized according to a given representation standard. Semantic data sources are compatible with all service interfaces considered for this model.

3.3.2 De-referenciability

Converting information into accessible Web resources is pivotal for creating the Web of linked data. This dimension is concerned with the accessibility levels of the data managed by data providers. Two alternatives are considered by this dimension: static and managed URLs. Both of them are only applied to semantic data sources.

Static. Static de-referenciability means that all the URLs that uniquely address and make Web resources accessible over the internet are created during the materialization process. However, there is no guarantee that these URLs are permanent and valid in case an infrastructural change occurs.

Managed. Data providers classified with this capacity are able to properly manage the accessibility of their Web resources. It means that these data providers ensure that a given URL associated with a provided Web resource is valid and able to return a semantic representation as long as the Web service is up and running. This capacity is incompatible with SPARQL endpoints and File Dump service interfaces, since they do not control the URLs that address their Web resources. Moreover, they are not even capable of responding to such links. Despite the fact that SPARQL endpoints offer an interface for answering complex queries, they do not resolve independent Web resource URLs. The service interface provided by a File Dump is even more restricted, since it provides only one URL to the entire data source.

3.3.3 Representational Management

The Representational Management dimension is concerned with data representation issues. Two capacities are considered in this dimension: languages and serialization formats. This dimension is an adaption of the versatility dimension proposed by Zaveri et al. (2015). The authors state that this dimension is also related to the interpretability of a data source, given that supporting more data representation formats results in data being more likely to be properly interpreted.

Serialization formats. Data providers classified with this capacity are able to offer multiple serialization formats to their consumers. Generally, the selection of a given serialization format is the result of content negotiation between the data provider and a given consumer. Such content negotiation is implemented by following standards that allow consumers to receive Web resources in a more appropriate format for their needs. This capacity is incompatible with the File Dump service interface since it just provides a link to a data source. A File Dump service provider may offer several links to the same data source serialized with distinct formats. However, this is not based on content negotiation, thus it is not considered compatible with this capacity.

Languages. This capacity means that a data provider is capable of providing Web resources in multiple human languages. All the considered service interfaces have a standardized mechanism that addresses this feature. For SPARQL endpoints, it is possible to inform the desirable human language directly in the query. For HTTP Web APIs and File Dump, it is done through the `Accept-Language` HTTP header, as specified by RFC 2616 [1].

3.3.4 Authoring

This dimension is committed with ownership of Web resources, and has as its main goal to establish a trust relationship by allowing publishers to take responsibility for their information as well as allowing consumers to verify data provenance, licensing and versioning.

Provenance. According to Gupta (2009), the term data provenance refers to a record trail that accounts for the origin of a piece of data together with an explanation of how and why it got to the present place. In order to reach this capacity, a data provider has to offer to

[1]https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.4

their consumers the means for verifying the authenticity of the data source based on a provenance vocabulary, such as author, contributors, publisher, etc.

Licensing. Licensing is defined as the granting of permission for a consumer to reuse a data source under defined conditions. Licensing is mandatory in the open data world. A license enables information consumers to use the data under clear legal terms (BATINI; SCANNAPIECO, 2016). To reach this capacity, data providers must include licensing information in their managed Web Resources.

Versioning. Versioning is directly related to provenance, since controlling the information trail is an important aspect for its implementation. Therefore, data providers classified with versioning capacity must be able to represent information updates, which implies that the content of a given Web resource cannot be changed without generating traceable information.

3.3.5 Knowledge Management

Knowledge management is a very broad concept, which encompasses a series of actions designed specifically to facilitate the sharing and integration of knowledge (RUBENSTEIN-MONTANO et al., 2001). This work adopts a pragmatic view and restricts its application to the use of semantic inference. Broadly speaking, inference on the Semantic Web can be characterized by discovering new relationships based on vocabularies or rule sets[2].

Reasoning using rules. This capacity means that a data provider should manage its data source taking into account predefined rules. An example of a rule is presented as follows:

```
:{ ?personA foaf:knows ?personB. }
=>
{ ?personB foaf:knows ?personA. }
```

This rule states that if a given personA knows a personB, then a personB also knows a personA. Considering that the data source contains originally the information that personA knows a personB, a data provider classified with this capacity should be able to automatically generating further information that meets with the predefined rules, in this case, generating personB knows a personA. There are two distinct

[2]https://www.w3.org/standards/semanticweb/inference

approaches to implement this feature, generating this new information at the moment a query is issued by a consumer, or an independence process executed at the moment the data provider is started. SPARQL endpoints generally offer built-in reasoners that can be enabled to generate such additional information at query-time. Similarly, HTTP Web APIs can implement such features to avoid the costs of a new materialization. However, File Data Dump service providers are partially compatible with this capacity since they have to make use of an independent procedure to generate the inferred information.

Reasoning using vocabularies. This capacity means that a data provider should manage its data source taking into account predefined ontology concepts. An example of such concepts is presented as follows:

```
:RedWine
  a owl:Class ;
    rdfs:subClassOf
      [ a owl:Restriction ;
        owl:onProperty :color ;
        owl:hasValue "red"
      ] .
```

RedWine is a new concept defined as a restriction. A data provider classified with this capacity level should be able to consider such conceptualization in order to create new information. As a result, the data provider should be able to answer queries for RedWine as well as validate inserts/updates regarding this concept. Similarly to the Reasoning using rules, the reasoning can take place at query time or during an independent materialization process. Additionally, it has the same service interface compatibility.

Ontology alignment. Data providers that reach this capacity should be able to manage Web resources described by heterogeneous ontologies. Based on a previously defined alignment, a data provider must be able to properly interpret consumers' requests involving queries that include terms defined in an ontology that differs from the one that was used to describe the materialized dataset, and retrieve the correct Web resource. File Dumps are classified as partially compatible with this capacity due to the fact that it is necessary to perform a new materialization that includes the previously defined alignments, while SPARQL endpoints and HTTP Web APIs permit implementing this feature at query time.

3.3.6 Re-design

This dimension encompasses capacities capable of modifying the structure of Web resources. Such modifications aim at improving data reusability by changing the content of Web resources in order to achieve higher levels of coherence and cohesion. Modifications considered in this dimension are: coreference resolution, backtrack, blank node conversion, and data pattern recognition. Due to the long processing time required for executing re-design procedures, it is considered that they are performed by independent processes rather than at query time. Thus, all service interfaces considered in this model are compatible with this dimension.

Coreference resolution. According to Cheatham & Pesquita (2017), coreference resolution algorithms attempt to determine when the same instance (i.e., individual) is referenced using different URIs. Coreference resolution can be thought of as data de-duplication, which has been an area of research for decades. Stated that, data providers that reach this capacity level should be able to identify duplicate instances and connect their respective Web resources.

Backtrack. Backtrack implies generating links into a given Web resource in order to state that its URL is associated with a given property in another Web resource. Data providers that achieve this capacity level provide richer and more navigable Web resources to their consumers.

Blank node conversion. Blank nodes are unnamed Web resources with identifiers that are only unique in a local scope. According to the quality assessment for linked data proposed by Zaveri et al. (2015), blank nodes should be avoided in order to make data much more interpretable. To address this matter, data providers should be able to convert blank nodes into addressable and accessible Web resources.

Data pattern recognition. Data sources may have a significant amount of duplicate data. However, it is possible to take advantage of data pattern recognition combined with entity linking techniques to identify and promote data duplication into new Web resources that may be connected with multiple Web resources. It improves data reusability at the same time that it increases interconnection among Web resources. Therefore, in order to reach this capacity, data providers should be able to re-design their Web resources based on data pattern recognition procedures and interlink the resulting Web resources.

3.3.7 Information fusion

According to Bergamaschi et al. (2018), information fusion is the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation. Differently from the Re-design dimension, in which modifications are restricted to link creation, this dimension goes a step further by incorporating data from multiple Web resources into a more representative one.

Redundant decision fusion. As aforementioned, data sources may have significant data duplication. Instead of addressing this duplication by using links that hold the semantics of the same real-world object, data providers classified with this capacity level should be able to fusion such resources into a single one. This capacity is especially important for data providers that support write operations, therefore, preventing data inconsistency.

Complementary decision fusion. Similarly to redundant decision fusion, complementary fusion aims at combining several Web resources that share enough information into a single one. This capacity is especially useful for performance issues, benefiting consumers that become able to retrieve a larger representation with a single request rather than fragmented ones linked with one another.

3.3.8 Continuous Improvement

Data providers may have to be able to handle data that changes over time. This dimension takes into consideration the capacities required to deal with this dynamic nature.

Self-adaptiveness. According to Immonen & Pakkala (2014), self-adaptiveness means that software is expected to fulfill its requirements at run-time by monitoring itself and its context, detecting changes, deciding how to react, and acting to execute decisions on how to adapt to these changes. This requires both context-awareness and reliability-awareness. Thus, data providers that reach this capacity level have to continuously monitor possible changes and decide how to react.

User feedback. In order to reach this capacity level, data providers should implement user feedback mechanisms. Example of aspects that can be considered include: allowing consumers to inform if a given data modification – e.g., a link or an information fusion – is correct or useful. Based on the resulting feedback, data providers may activate self-adaptiveness features to properly address possible issues.

3.4 FINAL CONSIDERATIONS

This chapter presents a systematic literature review on Web service description, selection, discovery, and composition. In this review, we have discussed specific approaches for data-driven and action-based Web services as well as identified the use of semantics in the context of Web services. This chapter also presents a review of original articles on data-driven services. Finally, a capacity model for semantic data providers is proposed based on the conducted reviews.

In order to properly define the concept of semantic Web service it is necessary to take into consideration several aspects, such as those described by the capacity model. In fact, what turns a Web service into a semantic one is not a single characteristic, but a collection of semantic features that a given implementation may or may not hold. Therefore, a Web service may be ranked according to a semantic scale that characterizes its semantic capabilities.

One can notice that the adopted service interface has little influence on semantic capacities. Despite SPARQL be the most compatible service interface, it does not ensure the second linked data principle that states the importance of using HTTP URIs for de-referencing entities. According to Stadtmüller, Speiser & Harth (2013), the combination of linked data with REST architectures allows to combine the advantages of both paradigms when offering functionality on the web: the easy data integration offered by linked data together with the flexibility of REST enables lightweight and adaptive services. However, a data provider that offers more than one service interface would be able to fulfill distinct consumer perspectives.

4 DATA LINKING AS A SERVICE

Data Linking as a Service (DLaaS) is a microservices infrastructure for publishing linked data. The proposed infrastructure aims at facilitating the execution of necessary processes to properly publish high quality linked data, which includes semantic enrichment, data linking, and publication. Its main goal is to link Web resources from distributed and heterogeneous datasets that share a certain level of data intersection, as depicted in Figure 5. For this purpose, it performs analysis on potential data intersection to create links between Web resources and therefore provides a navigable view of the whole collection of data.

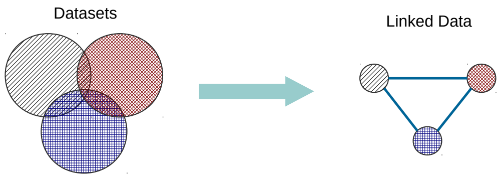


Figure 5 – From distributed and heterogeneous datasets to linked data

DLaaS is a microservice infrastructure to connect and publish entities of datasets as accessible Web resources. It makes use of semantic data-driven microservices, which work as data providers. As shown in Figure 6, each dataset is managed by a dedicated microservice, which is responsible for providing an access interface for the dataset's entities.

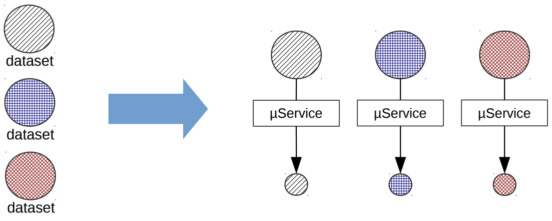


Figure 6 – Microservices as data providers

DLaaS combines microservices architecture and data linking principles for integrating data. As depicted by Figure 7, DLaaS is a virtual infrastructure for managing multiple DLaaS Platforms and DLaaS Containers. A DLaaS Platform is a configurable environment responsible for managing user submissions, which in turn are connected with multiple DLaaS Containers. A user submission is composed by a non-semantic dataset along with a configuration, which consists of a domain ontology and a semantic mapping that associates attributes of the dataset with terms defined in the domain ontology. A DLaaS container is a computing infrastructure for running instances of semantic data-driven microservices. Considering that platforms and containers are deployed in multiple distinct hosts, DLaaS must be able to take into consideration such a distributed environment in order to connect and publish data. Despite the fact that microservices are spread over several distinct containers managed by several platforms, their Web resources can be connected with one another based on methods for identifying data intersection that are proposed in this work. Furthermore, new platforms and containers can join the infrastructure at runtime, imposing significant challenges.

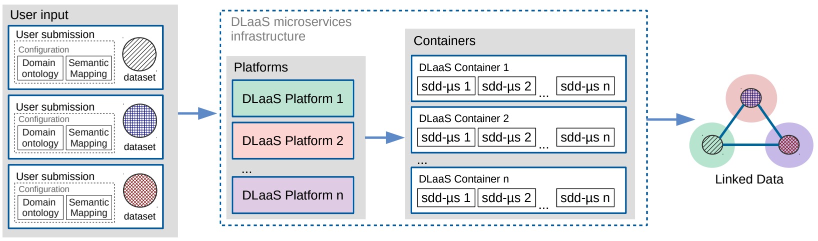


Figure 7 – DLaaS - architecture overview

Figure 8 describes in detail the DLaaS architectural components. A DLaaS Platform is composed by a submission endpoint, which is responsible for managing users input in form of HTTP REST requests, and by a JADE agent. DLaaS adopts multi-agent systems to properly support the dynamic nature of the infrastructure. Since new platforms and containers can be created at runtime, agents play an import role to propagate changes over the entire infrastructure. In addition, agents are used as an inter-container and inter-platform communication mechanism.

A DLaaS Container is composed of specific-purpose services, a computational environment for running sdd-µs instances and JADE agents. The specific-purpose services are responsible for performing specific tasks, which include data analysis for resource structure optimization, ontology alignment and data linking. These tasks are performed by Linkedator, Alignator and L2R respectively. Linkedator

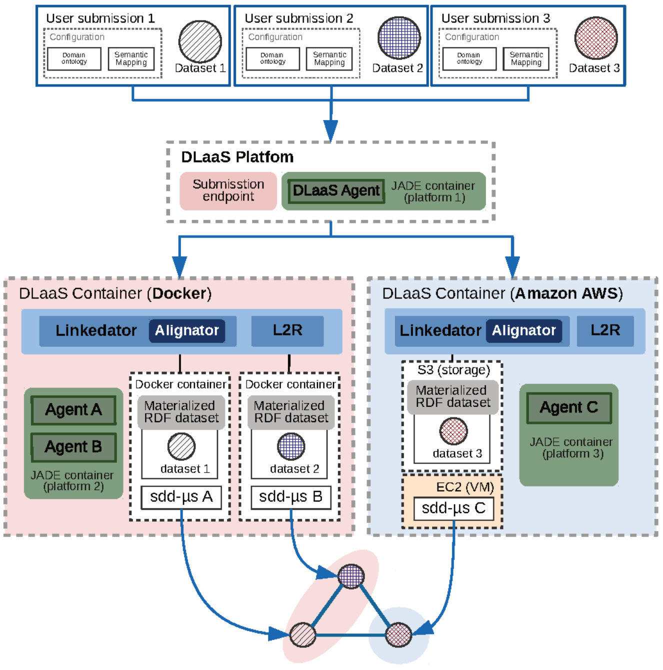


Figure 8 DLaaS - architecture in details

is responsible for linking Web Resources spread over several semantic microservices. Alignator is responsible for finding out alignment triples among several ontologies used to conceptualized data managed by several collections. L2R is a tool for converting literal values into independent and accessible Web resources. Further details on these specific-purpose services are presented in Chapter 7.

The sdd-µs, presented in Chapter 5, is a specialized service capable of converting non-semantic data into linked data. For each user submission, a sdd-µs instance is created for publishing its dataset as accessible Web resources. Two types of container are currently available in the existing version of DLaaS. One is based on Docker[1], which takes advantage of virtualization to isolate processes in a host operating system. The other one is based on the Amazon AWS cloud computing infrastructure. This implementation permits the use of this cloud provider infrastructure to instantiate microservices according to

[1]https://www.docker.com/

a variety of configurations. For this specific implementation type, sdd-µs instances are executed by EC2 virtual machines[2] and datasets are stored by the S3 storage solution[3]. For each running sdd-µs instance, a dedicated JADE agent is addressed to represent it in a multi-agent system. Agents exchange messages to allow the creation of links between Web resources managed by microservices spread over different containers. Agents are also associated with DLaaS platforms to permit the communication across several platforms.

4.1 SEMANTIC ENRICHMENT

Producing semantically enriched data is the first step towards the Web of data. However, most of the information produced is not described by an ontology nor structured as RDF triples, posing obstacles to data integration and reuse. In order to address this issue, DLaaS provides the means to semantically enrich datasets. The semantic enrichment is performed based on a semantic mapping that associates attributes of a dataset with terms defined in a domain ontology. The semantic enrichment is performed by the respective sdd-µs instance created at the moment a user submits a dataset to the infrastructure. Semantically describing the data results into a conceptual layer that could be useful to improve data reuse and interpretability.

Semantic applications are less vulnerable to changes in business requirements. When a conceptual layer is defined, a semantic-aware application may not need changes in its code base for addressing new business requirements. Let's consider the following example. Figure 9 exemplifies a Web resource semantically described and represented in JSON-LD. Let's imagine that the DLaaS infrastructure was initially capable of exposing resources about onto:Wine. Consider now that a requirement change demands listing wines based on their price. For example, create a new category for listing exclusive wines that cost from 500 to 1000 monetary units. In order to achieve such desired search, a non-semantic application would require the implementation of a new feature, resulting in a codebase change. However, DLaaS allows implementing that directly in the conceptual layer by defining a new semantic class as described by Figure 10. By improving the conceptual layer, DLaaS would expose now Web resources that represent *onto:ExclusiveWine* that match property values defined as OWL

[2]https://aws.amazon.com/ec2/
[3]https://aws.amazon.com/s3/

restrictions.

```
{
"@context": {
"onto": "http://example.com/ontology/wine.owl"
},
"@type": "onto:Wine",
"onto:name": "AnExpensiveOne",
"onto:price": "999.99",
"color": "onto:Red",
"grapeType": "onto:Albarossa",
"harvestYear": "1990"
}
```

Figure 9 – Example of a semantically enriched Web resource

```
:ExclusiveWine
        a owl:Class ;
        rdfs:subClassOf
        [ a owl:Restriction ;
          owl:onProperty :grapeType ;
          owl:minQualifiedCardinality "500"^^xs:nonNegativeInteger ;
        ] ;
        rdfs:subClassOf
        [ a owl:Restriction ;
            owl:onProperty :grapeType ;
            owl:maxQualifiedCardinality "1000"^^xs:nonNegativeInteger ;
        ] ;
```

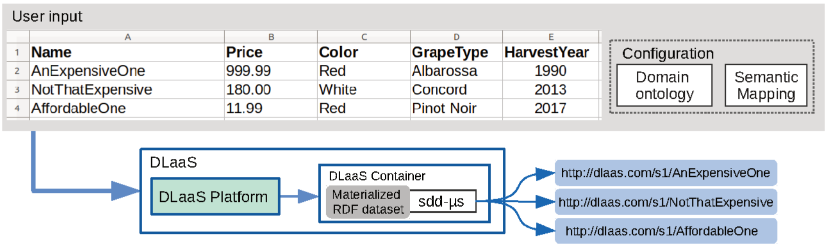
Figure 10 – Conceptual definition to address a requisite change

4.2 ACCESSIBLE WEB RESOURCES

As aforementioned, data is not usually published according to linked data principles. Due to this, data readability, reusability, and accessibility are significantly limited. Some examples are official government portals, such as the Transparency Portal of the Brazilian Government[4], which provides data about federal contracts, payments, federal workers, social programs and a variety of other public information, or the Public Security Secretariat of the state of São Paulo (SSP/SP)[5], which discloses police reports information, both publish non-semantic data. More specifically, these portals expose CSV files for download.

Publishing data through CSV (or JSON, XML, etc.) files drastically limits data accessibility. The reason is that when the file is

[4]http://www.portaltransparencia.gov.br
[5]http://www.ssp.sp.gov.br/transparenciassp



	A	B	C	D	E
1	**Name**	**Price**	**Color**	**GrapeType**	**HarvestYear**
2	AnExpensiveOne	999.99	Red	Albarossa	1990
3	NotThatExpensive	180.00	White	Concord	2013
4	AffordableOne	11.99	Red	Pinot Noir	2017

Figure 11 From records to accessible Web resources

associated with a single HTTP URL, inner registers that represent the actual data are not independently accessible. In contrast, DLaaS converts into addressable Web resources every record submitted by users. As depicted in Figure 11, a CSV file containing several records is converted and materialized as RDF resources. Each resource is addressed with a unique HTTP URL and available for consumers through a variety of data representation formats, such as JSON-LD, RDF/XML, Turtle, N3, among others. As a result, DLaaS provides semantically enriched data that can be easily accessible and reusable for many purposes.

4.3 DATA LINKING AND DATA REUSE

Corporations and governments produce, collect and store large volumes of data. Proper management of such data is not a trivial task. However, there are a variety of approaches that may be adopted to better handle it. The approach most widely adopted by Web applications consists in exposing data through web pages or data dumps. Nevertheless, few implementations follow principles such as universality and decentralization. Universality means that the information should be accessible through URIs, while decentralization means that there is no central authority to create and expose new data or to interlink existing data (BERNERS-LEE, 2010), avoiding the creation of data silos.

In fact, interlinking resources from distributed sources can be seen as a mechanism to improve data reusability. In this context, an existing Web resource is associated with other ones according to a given association strategy. As a result, interconnected resources provide more complete information, which allows to fulfill different user expectations and use data for different purposes rather than originally designed.

What differentiates the DLaaS from other proposals is the capability of interconnecting data, consequently improving its reusability. DLaaS considers three different methods for interlinking Web resources. The first method aims to change the Web resource structure to maximize data reuse. The second method aims at converting literal values into Web resources. Finally, the third method aims at creating links based on explicit semantic definitions.

4.3.1 Resource Structure Optimization

An important step toward the Web of Data is properly organizing and exposing data, which includes making decisions with regard to data structures and formats, as well as mechanisms to allow internal and external consumers to make use of the data. With this regard, this method applies techniques based on Mining Association Rules to identify patterns, allowing the data structure to be changed in order to maximize data reuse.

As can be seen in Figure 12, Data Mining algorithms are used to identify data patterns in Web resources from a dataset managed by a sdd-µs. The resulting Mining Association Rules are used to figure out a set of properties that can be promoted to an independent Web resource (newResource), and then linked to the original Web resources (resource1 and resource 2). Further details about this data linking strategy are described in Section 5.3.

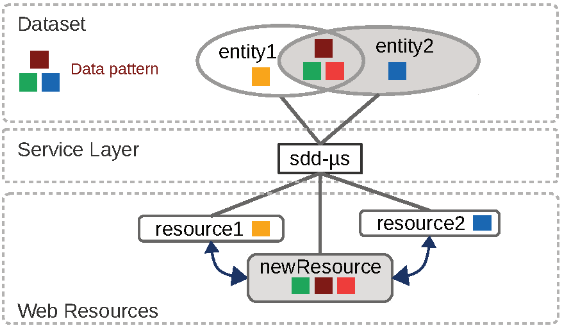


Figure 12 Data linking strategy based on Mining Association Rules

4.3.2 Explicit Semantic Definition

Information about a given Web resource may be spread over several datasets. Our data linking strategy aims at finding and connecting Web resources that represent the same real-world object but are managed by multiple sdd-µs instances. By creating links that correspond to object properties in a domain ontology. It takes as input a set of sdd-µs descriptions, a domain ontology and a Web resource representation that is meant to be enriched with links. Although the general data linking output results in a collection of mappings between two entities from two datasets, this data linking strategy is meant to append such links directly to a given Web resource.

The major design constraint is that the Web resources must be instances of classes previously defined in domain ontologies, and each sdd-µs can only use classes and properties from a single ontology. Furthermore, data attributes of resources must correspond to data properties defined by the ontology. Likewise, links among resources correspond to OWL object properties. These correspondences are not required for any purpose other than creating links and are therefore not enforced. Resources may contain data that is not present in the ontology and some object properties might never originate links.

An overview of this data linking strategy is shown in Figure 13. In this example the ontology contains two classes ($C1$, $C2$) and an object property P that has $C1$ as its domain and $C2$ as its range. $C1$ and $C2$ are managed, respectively, by sdd-µs1 and sdd-µs2. If a particular entity of $C1$, *entity*1 is related to an entity of $C2$, *entity*2, through property P, sdd-µs1 may not store a link to *resource*2, but only the necessary data to obtain the Web resource when querying from its respective service. From data that is actually managed by sdd-µs1, it is possible to represent the relation between *entity*1 and *entity*2 through a link in *resource*1 referring to *resource*2. Further details about this data linking strategy are presented in Section 6.1.2.

4.3.3 Literal to Resource Conversion

Literals often provide human-friendly information. For example, `rdfs:label` is used to provide a human-readable name for the resource, as opposed to the resource URI, which provides a machine-readable name for the resource. Another example is `rdfs:comment` that provides a textual description. These types of literal nodes are not generally

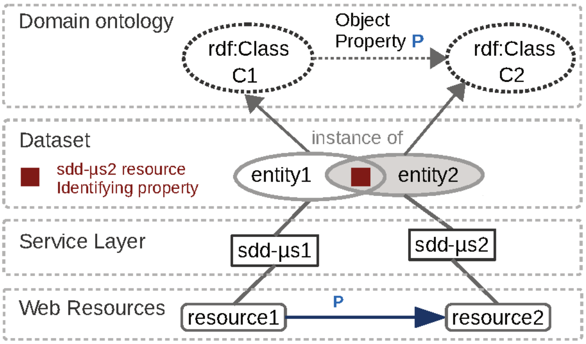


Figure 13 Data linking strategy based on explicit semantic definitions

useful for machine understanding and are not the primary aim of the Semantic Web (MEYMANDPOUR; DAVIS, 2016). Moreover, literals often hold textual content that cannot be dereferenced, consequently do not contribute to findability (BEEK et al., 2017), which means the ability to find Web resources under the current LOD publication paradigms.

With this regard, DLaaS performs analyses over the data from several sdd-µs in order to replace literal values with URLs that properly identify Web resources containing the same information as a value of `rdfs:label` property, as represented in Figure 14. Further details about this data linking strategy are described in Section 6.2.

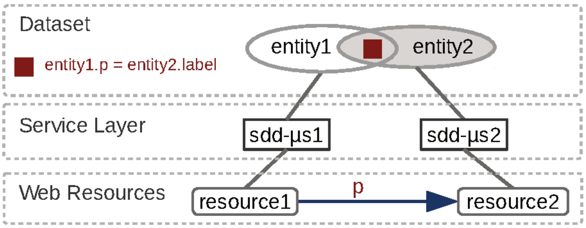
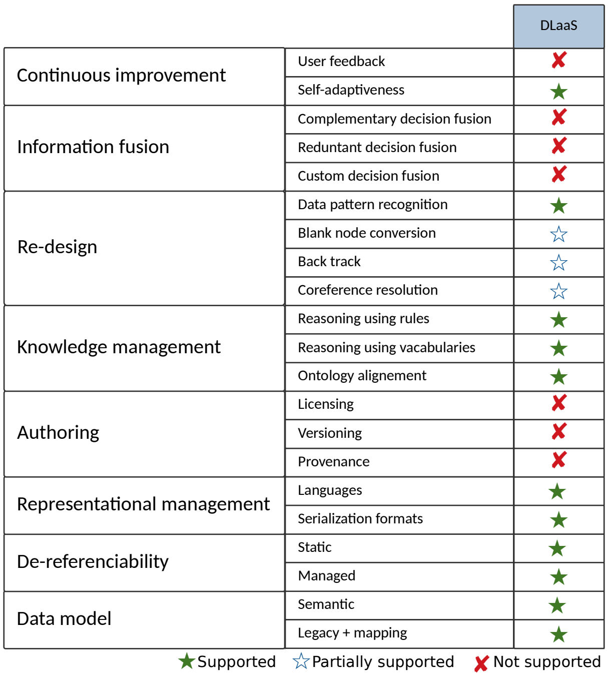


Figure 14 Data linking strategy based on literal to resource conversion

4.4 DLAAS SEMANTIC CAPACITY LEVEL

This section presents an analysis of the semantic capacity level achieved by the DLaaS infrastructure based on the Capacity Model for Semantic Data Providers previously presented in Section 3.3. As can be seen in Figure 15, DLaaS does not support any level of semantic capacities regarding authoring and information fusion dimensions. Instead, it is focused on Web resource re-design, knowledge and representational management, de-referenciability and data model. It is important to mention that all the capacities that are not supported by DLaaS are compatible with the infrastructure, i.e., they can be implemented to improve even more its compliance with the capacity model. However, these aspects require further research work to figure out the most appropriate mechanisms for implementing such features, which are out of the scope of this work.



		DLaaS
Continuous improvement	User feedback	✗
	Self-adaptiveness	★
Information fusion	Complementary decision fusion	✗
	Reduntant decision fusion	✗
	Custom decision fusion	✗
Re-design	Data pattern recognition	★
	Blank node conversion	☆
	Back track	☆
	Coreference resolution	☆
Knowledge management	Reasoning using rules	★
	Reasoning using vacabularies	★
	Ontology alignement	★
Authoring	Licensing	✗
	Versioning	✗
	Provenance	✗
Representational management	Languages	★
	Serialization formats	★
De-referenciability	Static	★
	Managed	★
Data model	Semantic	★
	Legacy + mapping	★

★ Supported ☆ Partially supported ✗ Not supported

Figure 15 – DLaaS - achieved semantic capacity level

Regarding the continuous improvement dimension, DLaaS supports only the self-adaptiveness capacity. This is due to its capacity of continuously monitoring possible changes through multi-agent-systems that are able to decide how to react when new microservices join the infrastructure as well as when any changes on semantic definitions and resulting ontology alignments take place.

With respect to re-design, DLaaS is capable of adapting the structure of Web resources originally modeled by data owners or originated by the semantic enrichment process by performing data pattern recognition techniques to maximize data linkage and consequently data reuse. Coreference resolution, back track and blank node conversion capacities were ranked as partially supported due to their respective outcomes being partially achieved by the execution of other processes rather than a dedicated process originally designed to handle them. Partially outcomes regarding coreference resolution and blank node conversion are obtained through the execution of a semantic enrichment process, which is able to identify redundant data, model it as a single Web resource and link it with other Web resources according to the aforementioned data linking strategy. Additionally, the semantic enrichment process takes into consideration the semantic mappings that hierarchically organize the data. As a result, a single non-semantic data will be converted into multiple Web resources uniquely addressed and accessible by a distinct HTTP URL instead of a local URI, which characterizes blank nodes.

Regarding the knowledge management dimension, DLaaS is capable of performing ontology alignments through the Alignator Framework, proposed in Section 6.1.3, to allow Web resources described by heterogeneous ontologies to be linked with one another. Furthermore, outcomes regarding reasoning using rules and vocabularies are achieved with the adoption of Apache Jena's OWL Rule Engine and of a SPARQL rewriter module, described in Section 5.4.

Semantic outcomes concerning representational management dimensions are achieved by providing to data consumers a variety of serialization formats to better represent Web resources. Additionally, the support for multiple human languages is provided by the built-in solution of SPARQL endpoint, which allows data consumers to include a language of choice when issuing queries.

With regard to the de-referenciability dimension, DLaaS supports both static and managed alternatives. Data owners may define a URL prefix to turn materialized Web resource identifies into managed HTTP URLs. An alternative is to keep them as originally materialized.

Therefore, DLaaS will not resolve its Web resources' URLs, maintaining this responsibility with the original data provider.

Finally, DLaaS supports both semantic materialization and legacy data along with a semantic mapping. Semantic datasets properly represented as RDF triples may be directly used as sources to DLaaS. Therefore, the semantic enrichment process can be skipped and the custom materialization will be used. On the other hand, DLaaS can be used to expose legacy data as semantic Web resources. It requires a semantic mapping to properly convert it into an RDF dataset that will be materialized. However, DLaaS is designed to work as a read-only data provider, which implies that manual modifications of the dataset may not result in changes to their respective Web resources and will not trigger any data linking process. On the other hand, modifications of terminological concepts will affect DLaaS behavior and will start internal processes to adapt the data according to such new definitions, which makes the infrastructure responsive and open to semantic refinements.

4.5 FINAL CONSIDERATIONS

This chapter presented an overview of DLaaS, a microservices infrastructure for publishing linked data. The main components of DLaaS were described, as well as the role played by them in the infrastructure. Additionally, the main features provided by DLaaS were presented, such as semantic enrichment of legacy data, the promotion of single data records to accessible Web resources and the creation of links between Web resources according to three data linking strategies. The first strategy is applied in the context of a single microservice, which reorganizes the structure of its Web resources to maximize data reusability. On the other hand, the second and third strategies are applied in the context of a collection of microservice instances, aiming to connect Web resources spread over the entire infrastructure.

This chapter also analyzed the DLaaS according to the Capacity Model for Semantic Data Providers previously presented in Section 3.3. This analysis emphasizes the focus of DLaaS on achieving semantic capabilities at the same time that it shows the strong and week points of this infrastructure. The following chapters describe in details the DLaaS internal components and evaluates their behavior and performance.

5 SEMANTIC DATA-DRIVEN MICROSERVICE

In recent years, the interest in data produced and published by companies and governments has increased. Such data has not to be necessarily attached to a specific logic layer. In this context, possible consumers are interested only in accessing the data, and not interacting with business procedures. An important step for publishing such data is to properly organize and expose data, which includes making decisions with regard to data structures and formats, as well as mechanisms to allow internal and external consumers to make use of it. Web technologies have been used to address these features. However, there are several approaches that may be adopted to better handle data exchange. As stated before, web pages and data dumps are the most widely adopted solutions for exposing data on the Web. Solutions better aligned with data exchange principles employ Web Services to expose data in a suitable format for being consumed by other software applications.

The vast majority of Web Services are implemented according to two main approaches: SOAP and REST. The former is defined in terms of the messages exchanged between a service provider and its clients. The latter provides a uniform interface to manage resources by following the semantics of the HTTP protocol. However, both approaches are used to expose functions defined in an application logic layer (PAIK et al., 2017).

Some proposals that facilitate the task of exposing data on the Web have been published in the literature. Research works such as Ontobroker (ANGELE, 2014) and DataOps (PINKEL et al., 2015) are focused on publishing linked data based on non-semantic data, whereas Linked REST APIs (SERRANO et al., 2017) and OntoGenesis (OLIVEIRA et al., 2017) are focused on augmenting legacy web services with semantic capabilities. However, none of them are focused on maximizing data reuse, which is one of the key features of the proposal presented in this thesis.

Another suitable alternative for exposing data on the Web is the use of microservices. Considering the set of characteristics that differentiate microservices from other implementation approaches – such as single-responsibility units, isolated state, distribution, elasticity and loose coupling – the microservices architecture turns out as a suitable alternative to develop data-oriented services. By explicitly separating data from business operations in distinct microservices, users can freely interact with data without the restrictions imposed by business operations. Due to this, data may be more effectively reused for different purposes.

This chapter presents sdd-µs, a specialized microservice capable of converting non-semantic data into linked data. The proposed service adopts the data-driven approach to implement microservices as data providers, which implies that the service does not implement any business operations, but only functionalities for managing the lifecycle of read-only data.

By using sdd-µs, there is no need to implement a Web Service to expose a data source on the Web. It converts simple data entries into semantic Web resources that can be linked to other resources provided by different data sources. In addition, it is able to identify data patterns and to adapt data structure in order to maximize data reuse. Furthermore, it provides means to infer new knowledge based on the available resources.

5.1 REFERENCE ARCHITECTURE

As shown in Figure 16, sdd-µs consists of several internal components, which includes an Ontology Manager, a Data Manager, an Inference Module, a Reusability Module and a Service Interface. When a data owner submits a dataset to DLaaS, a new instance of sdd-µs is created and becomes responsible for handling it. The Ontology Manager component is responsible for handling the configuration part of the submission, while the Data Manager is responsible for handling the data by adopting a suitable adapter. Adapters play an important role for interacting with non-semantic datasets. They work as data wrappers to access datasets in several formats, such as CSV, XML, JSON and others. Currently, only the CSV adapter has been implemented. However, other adapters can be easily implemented according to a Java interface and integrated to the sdd-µs.

When the sdd-µs starts for the very first time, the Data Manager component loads and converts the non-semantic dataset into RDF triples according to the semantic mapping provided by the user. These triples can be stored directly in the file system as an RDF file or in a triple store. Then, the ABox component (a set of assertional axioms) is responsible for managing these materialized triples, which represent the set of facts that refer to individuals exposed as accessible Web resources.

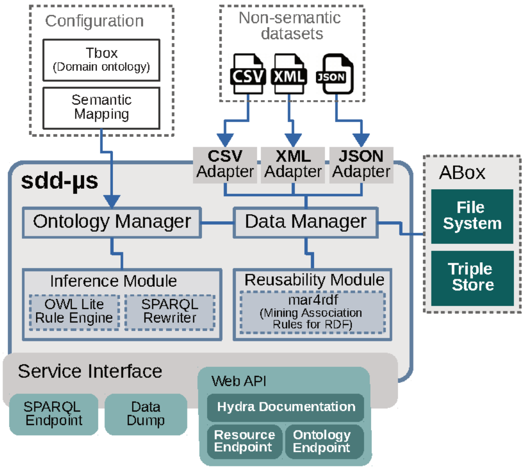


Figure 16 The sdd-µs architecture

Once materialized, the dataset can be retrieved through multiple service interfaces, which include a SPARQL endpoint, a data dump interface and a Web API. The SPARQL endpoint allows the execution of SPARQL queries on the dataset through HTTP GET requests. The data dump interface allows consumers to download a file that consist of all stored triples. Finally, the Web API is capable of providing linked data documents (Web resources with unique HTTP URLs) as well as a Hydra(LANTHALER, 2013) documentation that describes how to retrieve such entities. Web resources may be retrieved in a variety of formats, such as XML/RDF, Turtle, JSON-LD, among others, according to content negotiation. In addition, the Web API interface allows the domain ontology to be managed, which permits adding or changing concepts. Regardless of the chosen service interface, when a data consumer sends a request the Inference Module is activated to perform inferences based on defined terminological definitions.

The sdd-µs follows the set of best practices for publishing structured data on the Web defined by Berners-Lee (2011). However, what differentiates the sdd-µs from similar proposals found in the literature is the capability of optimizing data in order to improve its reusability. Two processes are performed to provide such a feature: semantic enrichment and resource structure optimization. These processes are described in the following two sections.

5.2 SEMANTIC ENRICHMENT

Producing semantically enriched data is the first step towards the Web of data. However, most of the information produced by governments, universities and enterprises is not available as such. Usually, data is not described by an ontology and is not structured as RDF triples, posing obstacles to data integration and reuse. In order to address this issue, sdd-µs provides the means to semantically enrich datasets. As depicted by Figure 16, sdd-µs accepts as input a configuration and a non-semantic dataset. The configuration contains a TBox (a set of terminological statements that conceptualize the dataset) and a semantic mapping that associates attributes of a non-semantic dataset with terms defined in the TBox.

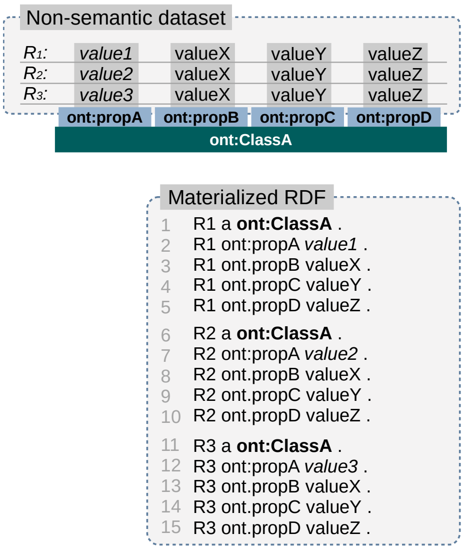
There are a variety of approaches for accessing the dataset managed by a data provider (DOAN; HALEVY; IVES, 2012). Data materialization and virtual integration are broadly adopted. In the former, the data is previously loaded, materialized according to a previously defined schema and stored. In the latter approach the data remains in the source and such materialization is performed to answer a query at runtime, without physically storing the data. The sdd-µs adopts the materialization approach, which means that the dataset is converted into RDF triples and stored as files or in a triple store when the service is initialized for the first time. Once the non-semantic dataset is converted into RDF triples, it is not used anymore. It is important to mention that sdd-µs also accepts as input a materialized RDF dataset. Then, the materialization step can be skipped.

Figure 17 shows an example of the semantic mapping for enriching a CSV file. The mapping is defined in JSON-LD syntax, in which keys are represented by CSV column headers and values are represented by the URIs of properties defined in the ontology. The reserved key `@type` is used to define a semantic class for each CSV record. In this example, `ont:propA` `ont:propB`, `ont:propC` and `ont:propD`, defined in the domain ontology, were associated with CSV column headers to be part of an independent resource instance of `ont:ClassA`. The resulting RDF materialization can be seen in Figure 18.

Advanced configurations can be applied to handle more complex mapping designs. Figure 19 shows an example of a semantic mapping where a CSV record is mapped into two distinct Web resources. This mapping results into a hierarchical organization in which Web resources are linked with one another through object properties. The resulting materialization can be seen in Figure 20.

```
{
  "@context": {
    "onto": "http://example.com/ontology/",
    "@type": "onto:ClassA",
    "CSV_colunmHeaderA": "onto:propA",
    "CSV_colunmHeaderB": "onto:propB",
    "CSV_colunmHeaderC": "onto:propC",
    "CSV_colunmHeaderD": "onto:propD"
  }
}
```

Figure 17 – Semantic mapping example



R1:	*value1*	valueX	valueY	valueZ
R2:	*value2*	valueX	valueY	valueZ
R3:	*value3*	valueX	valueY	valueZ

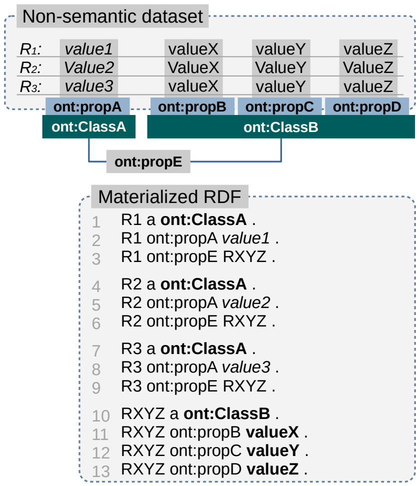
```
R1 a ont:ClassA .
R1 ont:propA value1 .
R1 ont.propB valueX .
R1 ont.propC valueY .
R1 ont.propD valueZ .

R2 a ont:ClassA .
R2 ont:propA value2 .
R2 ont.propB valueX .
R2 ont.propC valueY .
R2 ont.propD valueZ .

R3 a ont:ClassA .
R3 ont:propA value3 .
R3 ont.propB valueX .
R3 ont.propC valueY .
R3 ont.propD valueZ .
```

Figure 18 – RDF materialization example

One can notice that the hierarchical materialization strategy may result in a smaller semantic dataset in cases where there is significant data overlap. In this example, `ont:propB`, `ont:propC` and `ont:propD` were associated with a semantic class (`ont:ClassB`) in order to be part of an independent resource. In this particular case, several original records share the same values of these properties, resulting in linked resources and data reuse.

```
{
  "@context": {
    "onto": "http://example.com/ontology/",
    "@type": "onto:ClassA",
    "CSV_colunmHeaderA": "onto:propA",
    "onto:propE":{
      "@type": "onto:ClassB",
      "CSV_colunmHeaderB": "onto:propB",
      "CSV_colunmHeaderC": "onto:propC",
      "CSV_colunmHeaderD": "onto:propD"
    }
  }
}
```

Figure 19 – Semantic mapping example - hierarchical organization



```
R1 a ont:ClassA .
R1 ont:propA value1 .
R1 ont:propE RXYZ .

R2 a ont:ClassA .
R2 ont:propA value2 .
R2 ont:propE RXYZ .

R3 a ont:ClassA .
R3 ont:propA value3 .
R3 ont:propE RXYZ .

RXYZ a ont:ClassB .
RXYZ ont:propB valueX .
RXYZ ont:propC valueY .
RXYZ ont:propD valueZ .
```

Figure 20 – RDF materialization example - hierarchical organization

5.3 RESOURCE STRUCTURE OPTIMIZATION

Optimizing the structure of data is not a trivial task. It requires qualified specialists that have deep knowledge on the specific domain and software tools to assist them in this process. Though, even for a specialist, restructuring data in such a way that information would be better represented through linked resources may be something difficult to accomplish. By using data mining techniques, sdd-μs provides a means to optimize the initial RDF materialization in order to achieve higher levels of data reuse.

Table 5 – Example of records about employees

Property	R1	R2	R3
employeeName	Alice	John	Bob
employeeSecNumber	123	456	789
employeeBirthDate	01/01/1952	01/01/1954	01/01/1950
companyName	Void Corp	Void Corp	Acme Corp
companyLocation	São Paulo	São Paulo	New York
salary	2.000	1.000	1.000
admissionDate	01/01/2018	01/01/2018	01/01/2018

An important part of the optimization process is the discovery of association rules, which consists in determining relationships between sets of items in a very large database. Agrawal & Srikant (1994) state this problem as follows. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items. Let $D = \{t_1, t_2, ..., t_n\}$ be a set of n transactions, each one identified by a unique transaction id (TID). Each transaction t consists of a set of items from I and an itemset I is contained in a transaction $t \in D$ if $I \subseteq t$. The support of an itemset I is the percentage of transactions in D containing I. Association rules are of the form $r : I_1 \overset{c}{\rightarrow} I_2$, with $I_1, I_2 \subset I$ and $I_1 \cap I_2 = \phi$. Given the user defined minimum support (*minsup*) threshold, the problem of mining association rules can be divided in two sub-problems: (i) find all itemsets in D with support greater or equal to *minsup* and (ii) for each itemset found, generate all association rules $I_2 \overset{c}{\rightarrow} I_1 - I_2$ where $I_2 \subset I_1$. That been stated, sdd-µs adopts the algorithm A-Close proposed by Pasquier et al. (1999) as implemented in the Open-Source Data Mining Library (FOURNIER-VIGER et al., 2016).

To properly explain the structure optimization process, consider the following example. Table 5 presents data about three records, each one containing seven properties with values that share some level of association. The process starts with converting each record into a transaction, which implies converting its values into a set of items as an ordered numerical vector. This step consists in creating an index that associates each literal value with a unique integer number, as shown in Figure 21. These vectors are organized into a single matrix used as input to the A-Close algorithm.

It is necessary to perform several iterations in order to recognize frequent closed itemsets and then, generate association rules in different support thresholds. Iterations start from the maximum support

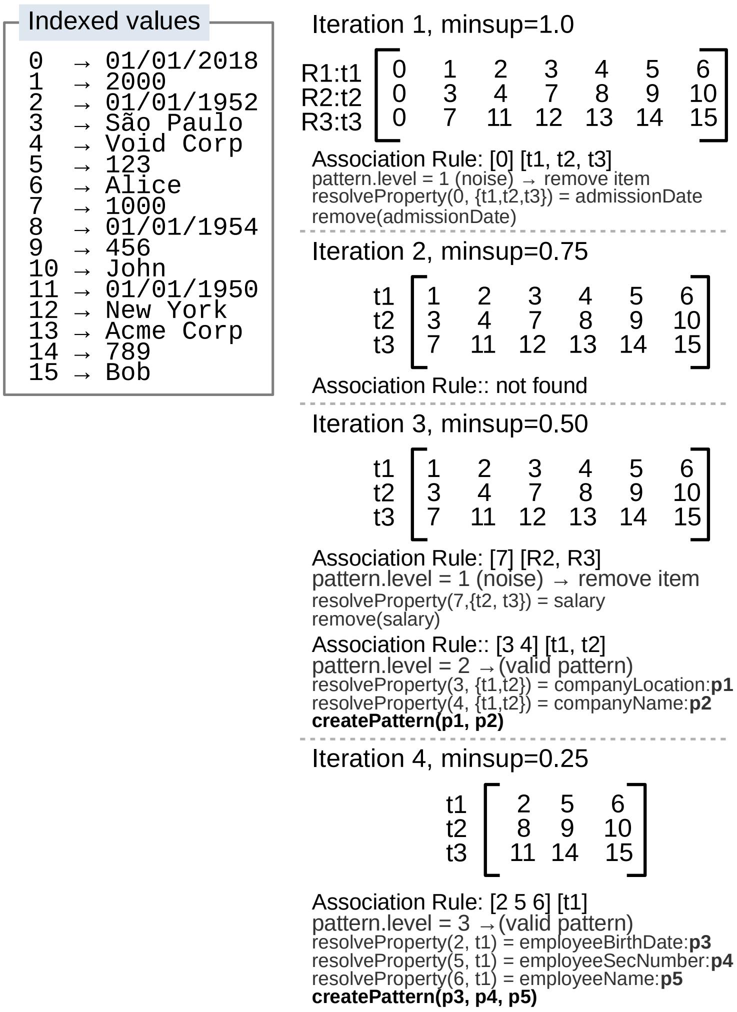
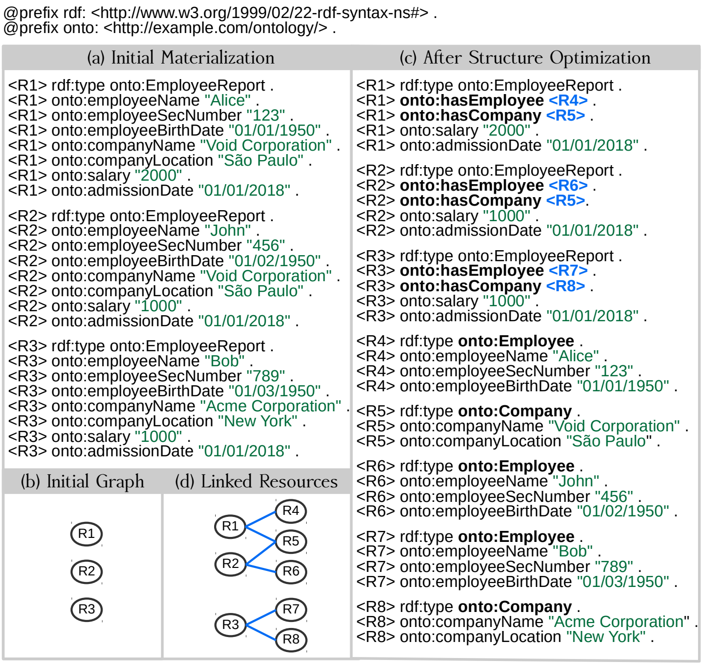

Indexed values
0 → 01/01/2018
1 → 2000
2 → 01/01/1952
3 → São Paulo
4 → Void Corp
5 → 123
6 → Alice
7 → 1000
8 → 01/01/1954
9 → 456
10 → John
11 → 01/01/1950
12 → New York
13 → Acme Corp
14 → 789
15 → Bob
Iteration 1, minsup=1.0
R1:t1 0 1 2 3 4 5 6
R2:t2 0 3 4 7 8 9 10
R3:t3 0 7 11 12 13 14 15
Association Rule: [0] [t1, t2, t3]
pattern.level = 1 (noise) → remove item
resolveProperty(0, {t1,t2,t3}) = admissionDate
remove(admissionDate)
Iteration 2, minsup=0.75
t1 1 2 3 4 5 6
t2 3 4 7 8 9 10
t3 7 11 12 13 14 15
Association Rule:: not found
Iteration 3, minsup=0.50
t1 1 2 3 4 5 6
t2 3 4 7 8 9 10
t3 7 11 12 13 14 15
Association Rule: [7] [R2, R3]
pattern.level = 1 (noise) → remove item
resolveProperty(7,{t2, t3}) = salary
remove(salary)
Association Rule:: [3 4] [t1, t2]
pattern.level = 2 →(valid pattern)
resolveProperty(3, {t1,t2}) = companyLocation:p1
resolveProperty(4, {t1,t2}) = companyName:p2
createPattern(p1, p2)
Iteration 4, minsup=0.25
t1 2 5 6
t2 8 9 10
t3 11 14 15
Association Rule: [2 5 6] [t1]
pattern.level = 3 →(valid pattern)
resolveProperty(2, t1) = employeeBirthDate:p3
resolveProperty(5, t1) = employeeSecNumber:p4
resolveProperty(6, t1) = employeeName:p5
createPattern(p3, p4, p5)

Figure 21 – Mining association rules example

and will be decreased according to a configurable parameter. In this example, the decrease rate was set to `0.25`. The first iteration is setup with minimal support to 1.00, which results in finding the association rule `[0]` in `t1, t2, t3`, with level 1. The level represents the number of items in a given itemset. Only association rules with level equal or greater than 2 are eligible to be part of a pattern, otherwise they are considered noise. The next step is to find the properties associated with the resulting itemset. In this case, the itemset `[0]` corresponds to the literal value `01/01/2018`, which in its turn is associated with the property *admissionDate* for all transactions. Finally, this item is removed from all itemsets of the initial matrix. It is important to mention that removing previously found closed itemsets is essential to properly recognize further association rules using lower support thresholds. Iteration 2 is setup with `minsup=0.75`, however, there is no association rule that is identified using this threshold. Iteration 3 results in two association rules. The first one is considered noise, which results in removing all items associated with the property `salary`. The second one is considered valid and its items are resolved to properties to be part of a pattern. In this case, a new semantic class is created and properties `companyLocation` and `companyName` will be restructured in an independent resource. In iteration 4, properties `employeeBirthDate`, `employeeSecNumber` and `employeeName` are also combined to be part of an independent resource.

The aforementioned steps are performed to recognize association rules and then generate patterns that essentially create new semantic classes and `objectProperties`. Based on these new concepts, a new RDF materialization is performed in order to update the dataset with the resulting structure optimization. Figure 22 (a) describes the initial RDF materialization, that basically translates the records of Table 5 into resources. This literal translation results in a disconnected graph, as shows Figure 22 (b). However, as Figure 22 (c) shows, the new RDF materialization contains new resources that aggregate properties according to the resulting patterns. These new resources are then connected with existing ones, resulting in a connected graph of linked resources, as describes Figure 22 (d). Despite the fact that, for this example, the resulting optimized materialization required more triples to represent the same information, a very large dataset with a significant level of data overlap would result in a smaller optimized dataset. Moreover, datasets without a clear separation of concepts or whose records have no apparent relation with each other will also benefit from the resource structure optimization.



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix onto: <http://example.com/ontology/> .
```

(a) Initial Materialization

```
<R1> rdf:type onto:EmployeeReport .
<R1> onto:employeeName "Alice" .
<R1> onto:employeeSecNumber "123" .
<R1> onto:employeeBirthDate "01/01/1950" .
<R1> onto:companyName "Void Corporation" .
<R1> onto:companyLocation "São Paulo" .
<R1> onto:salary "2000" .
<R1> onto:admissionDate "01/01/2018" .

<R2> rdf:type onto:EmployeeReport .
<R2> onto:employeeName "John" .
<R2> onto:employeeSecNumber "456" .
<R2> onto:employeeBirthDate "01/02/1950" .
<R2> onto:companyName "Void Corporation" .
<R2> onto:companyLocation "São Paulo" .
<R2> onto:salary "1000" .
<R2> onto:admissionDate "01/01/2018" .

<R3> rdf:type onto:EmployeeReport .
<R3> onto:employeeName "Bob" .
<R3> onto:employeeSecNumber "789" .
<R3> onto:employeeBirthDate "01/03/1950" .
<R3> onto:companyName "Acme Corporation" .
<R3> onto:companyLocation "New York" .
<R3> onto:salary "1000" .
<R3> onto:admissionDate "01/01/2018" .
```

(c) After Structure Optimization

```
<R1> rdf:type onto:EmployeeReport .
<R1> onto:hasEmployee <R4> .
<R1> onto:hasCompany <R5> .
<R1> onto:salary "2000" .
<R1> onto:admissionDate "01/01/2018" .

<R2> rdf:type onto:EmployeeReport .
<R2> onto:hasEmployee <R6> .
<R2> onto:hasCompany <R5>.
<R2> onto:salary "1000" .
<R2> onto:admissionDate "01/01/2018" .

<R3> rdf:type onto:EmployeeReport .
<R3> onto:hasEmployee <R7> .
<R3> onto:hasCompany <R8> .
<R3> onto:salary "1000" .
<R3> onto:admissionDate "01/01/2018" .

<R4> rdf:type onto:Employee .
<R4> onto:employeeName "Alice" .
<R4> onto:employeeSecNumber "123" .
<R4> onto:employeeBirthDate "01/01/1950" .

<R5> rdf:type onto:Company .
<R5> onto:companyName "Void Corporation" .
<R5> onto:companyLocation "São Paulo" .

<R6> rdf:type onto:Employee .
<R6> onto:employeeName "John" .
<R6> onto:employeeSecNumber "456" .
<R6> onto:employeeBirthDate "01/02/1950" .

<R7> rdf:type onto:Employee .
<R7> onto:employeeName "Bob" .
<R7> onto:employeeSecNumber "789" .
<R7> onto:employeeBirthDate "01/03/1950" .

<R8> rdf:type onto:Company .
<R8> onto:companyName "Acme Corporation" .
<R8> onto:companyLocation "New York" .
```

Figure 22 – Example of resource structure optimization result

5.4 SUPPORT FOR INFERENCE

Reasoning is an important feature, specially for data-driven implementations, which are focused on data and on the potential knowledge that may be inferred. Moreover, it may be seen as the most important purpose for adopting semantic Web techniques. This feature allows the derivation of new facts from those explicitly present in the data and in the concepts defined in the TBox. Such definitions can be used to provide distinct perspectives over the data and to empower data integration. The former may be implemented by defining class hierarchy such as subclasses, or by modeling restrictions to create new concepts based on specific conditions. The latter may be implemented by defining equivalences between classes and properties as well as between resources that represent the same object in the real world, usually expressed with `owl:sameAs`. It can also be seen as a built-in mechanism for dealing with data heterogeneity.

```
R1: <Resource1> a <http://example.com/onto/ClassA> .
    <Resource1> <http://example.com/onto/propA> "value1" .

T1: @prefix owl: <http://www.w3.org/2002/07/owl#> .
    @prefix onto: <http://example.com/onto/> .
    @prefix anotherOnt: <http://example.com/anotherOnt/> .
    onto:propA a owl:DatatypeProperty .
    onto:propA owl:equivalentProperty anotherOnt:propX

Q1: SELECT ?resource  WHERE {
    ?resource anotherOnt:propX "value1" .
    }

Q2: SELECT ?resource  WHERE {
      { ?resource onto:propA "value1" } UNION
      { ?resource anotherOnt:propX "value1" }
    }
```

Figure 23 – Query rewriting example

For this purpose, the Inference module provides reasoning support based on two different implementations: Apache Jena OWL Rule Engine and a SPARQL rewriter. The former uses the Apache Jena implementation[1] to perform inferences. The latter, a contribution of this work, rewrites a SPARQL query that requires reasoning into a new query that incorporates such elements based on a domain ontology analysis. Rewriting queries represents an alternative to the standard inference support offered by Jena. This alternative is necessary given that Jena may present a significant performance degradation, depending on the size of the dataset and on the complexity of the query.

Figure 23 shows a simple example of query rewriting. R1 presents a materialized resource, T1 shows a TBox that defines an equivalence between two properties. Q1 presents a SPARQL query that requires inference support to retrieve R1, since the resource has been previously materialized with an equivalent property. Finally, Q2 presents the resulting SPARQL query after the rewriting process. Currently, the query rewriter supports class and property equivalence, subclasses, as well as `owl:sameAs` statements.

[1]https://jena.apache.org/

5.5 EVALUATION

In this section we describe our experimental methodology and analyze the obtained results. The objective of this evaluation is to measure the efficiency of the resource structure optimization process regarding data reuse and its cost in terms of processing time. In addition, this evaluation aims at identifying the impact of performing reasoning, by comparing two different inference approaches: Jena OWL Rule Engine and SPARQL rewriting. In order to allow the replication of experimental results, source code and instructions for setting up the environment are available in a public repository[2].

The evaluation used real data from two distinct data providers. The first dataset is provided by the Public Security Secretariat of the state of São Paulo (SSP-SP)[3] - Brazil. The SSP/SP system publishes police reports that describe suspicious death, intentional homicide, robbery followed by murder, car theft, among others. Information about the report such as location, police station and date of the incident are available. For this evaluation, only police reports that describe car theft were considered. A total of 175 CSV files were downloaded, containing reports from 2003 to 2017. The second dataset is provided by The NYC Open Data portal[4], which publishes a variety of datasets, including data about business, health, education, government, environment, among others. For this evaluation, was considered the dataset Parking Violations (NYC- PVI)[5] - Fiscal Year 2019, which describes information such as vehicle details, data and location in which the violation took place, type of violation, among others. This dataset is provided as a single CSV file, which contains violations from 1 January, 2018 to September 10, 2018. This dataset is monthly updated.

Figure 24 shows the results regarding the resource structure optimization process. The SSP-SP dataset optimization is represented by Figure 24 (a). The initial materialization required 24.057 million triples to represent the information. In the initial materialization, each CSV record was converted into a single RDF resource, which holds all mapped properties. As a result of the structure optimization, a new materialization was created with all recognized data patterns as well as the necessary links to connect the original data with the new resources.

[2]https://salvadori.bitbucket.io/projects/sddms/

[3]http://www.ssp.sp.gov.br/transparenciassp/

[4]http://opendata.cityofnewyork.us/

[5]https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2019/pvqr-7yc4

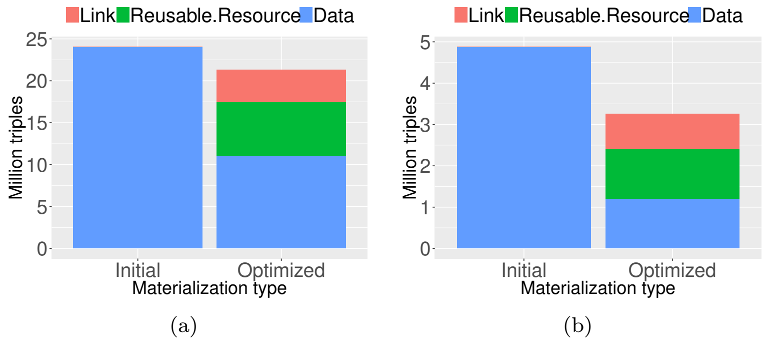
Those patterns are represented by resources, which are instances of new concepts. The optimized materialization required 21.292 million triples to represent the same information, resulting in a reduction of 11.7%. However, the most important result is that it was able to properly reorganize the information among more reusable resources. Three patterns were recognized for this dataset. The first one aggregates properties that describe the stolen vehicle. The second aggregates properties that describe the police station responsible for the report. The last one represents the common information about the location where the incident took place, which includes region, street and city name.

For the NYC-PVI dataset, the optimization resulted in a significant reduction of the materialized dataset, as shows Figure 24 (b). Two patterns were recognized for this dataset. The first one aggregates properties that describe the vehicle. The second one aggregates properties that describe the violation type along with the street name. While this optimization resulted in a significant reduction of the resulting dataset, it came up with a pattern that does not necessary represent a common sense reorganization. The reason for that is the lack of data to properly recognize the pattern; however, it represents the real nature of the data. In other words, the resulting patterns are result of the data, and do not necessarily follow a given domain logic.

Figure 24 (c) shows the processing time to perform the optimization for each dataset. The size of the dataset has a important effect on the processing time. The spikes represent the points where vectors of transactions were recreated due to a found pattern or a noise. it is worth to mention that the main factor is the size of the matrix rather than the minimal support threshold. For this reason, we can notice the decreasing of the required processing time.

This evaluation also takes into consideration the Inference Module, comparing Apache Jena OWL Rule Engine with the sdd-µs SPARQL rewriter. In order to compare these two inference approaches, three queries, shown by Figure 25, were evaluated. These three queries were issued every time a new Web resource was inserted into the RDF dataset throughout the materialization process for the SSP-SP dataset. Query Q1 only retrieves the latest inserted Web resource, Q2 filters a collection of resources based on the property `"timeOfDay"`, and Q3 produces the same result of Q2, however it requires reasoning for inferring the equivalence between `"TheftAutoReport"` and `"CriminalReport"`.

Figure 26 shows the results regarding the execution of these aforementioned queries. In Figure 26 (a), Q1, Q2 and Q3 were executed without inference support. One can see that Q3 produces no



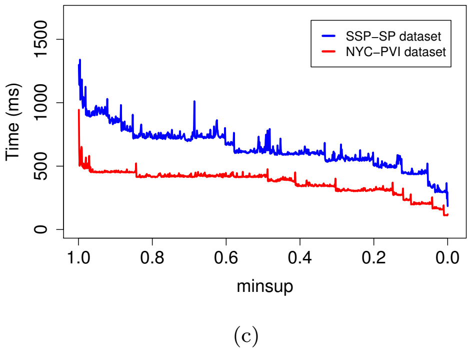
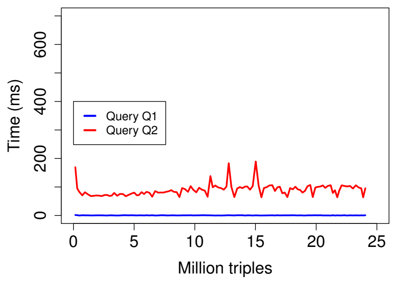


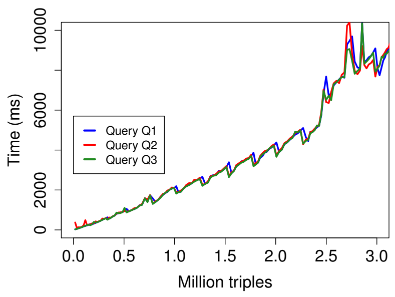
Figure 24 – Structure resource optimization results: (a) SSP-SP dataset, (b) NYC- PVI dataset and (c) processing time

```
@prefix onto: <http://www.public-security-ontology/> .
@prefix anotherOnto: <http://www.anotherOntology.com/> .

Q1: SELECT  ?p ?o { <{resource_URI}> ?p ?o}

Q2: SELECT ?resource  WHERE {
       ?resource a onto:TheftAutoReport .
       ?resource onto:timeOfDay "EVENING" .
    } limit 100 offset 0;

Q3: SELECT ?resource  WHERE {
       ?resource a onto:CriminalReport .
       ?resource anotherOnto:periodOfDay "EVENING" .
    } limit 100 offset 0;
```

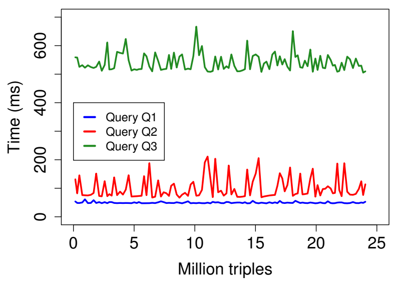
Figure 25 – SPARQL queries



(a)



(b)



(c)

Figure 26 – Query response time: (a) no inference support, (b) Jena inference enabled and (c) sdd-ms query rewriter

results as it requires reasoning features. However, considering all the 24.057 million triples according to the initial materialization[6], the execution time for Q1 is less than one millisecond, while for Q2 it is more than 100 milliseconds. However, when the Jena OWL Rule Engine is enabled, the time required for Q3 to produce the expected results increases sharply, as can be seen in Figure 26 (b). It is worth noticing that, regardless of the query, the execution time was similar. Approximately 10 seconds were required to execute each query over 3 million triples. Finally, Figure 26 (c) shows the result for the sdd-µs Query Rewriter. In this approach, the execution times for Q1 and Q2 were similar when compared to the execution without inference support. Moreover, the required time to execute Q3 was in the interval between 500 and 600 milliseconds.

5.6 FINAL CONSIDERATIONS

This chapter presented sdd-µs, a service capable of providing linked data based on non-semantic data sources. By using sdd-µs there is no need to implement new Web Services to expose data on the Web. The sdd-µs provides an efficient inference support for issuing queries on large semantic datasets. Moreover, it maximizes the reuse of data by performing a resource structure optimization process to identify data patterns. Based on these patterns, properties are combined in order to create new semantic concepts. Then, all resources with those properties will be restructured, resulting in a connected graph of linked resources.

Consumers can interact with the data through multiple service interfaces in order to fulfill different expectations and uses. However, what differentiates the sdd-µs from other proposals is the capability of optimizing data in order to improve its reusability. In addition, sdd-µs provides an efficient support for inference, which allows refining the data retrieval behavior based on conceptual terminologies.

Evaluation experiments showed the efficiency of the proposed optimization process, through which data patterns were recognized in real datasets, resulting in a significant reuse of data. In addition, the evaluation showed that the Jena OWL Rule Engine is not suitable for performing even simple inferences. By rewriting the SPARQL queries, sdd-µs query rewriter dramatically reduced the execution time, allowing the use of inference, even for a small set of axioms.

[6]There were no significantly different results in the initial and the optimized materialization for executing the queries used in this evaluation.

6 INTER-SERVICE LINKING FOR SEMANTIC DATA-DRIVEN MICROSERVICES

This chapter presents implementation details on data linking strategies previously presented in sections 4.3.2 and 4.3.3, but with a data-driven microservice composition perspective. With regard to explicit semantic definition data linking strategy, Linkedator is proposed as a development tool for facilitating the implementation of services meant to provide linked Web resources in a microservices architecture. An important feature is to deal with Web resources semantically described by heterogeneous ontologies. In order to address this feature, Alignator is proposed as a support tool for identifying alignments of heterogeneous ontologies allowing compositions in cross-domain environments. With regard to literal to resource conversion, L2R is proposed as a support tool for converting literal values into Web Resources. Finally, JADE agents are presented as a communication mechanism for linking Web resources of the entire DLaaS infrastrucure.

6.1 EXPLICIT SEMANTIC DEFINITION

This composition method aims at linking Web Resources spread over several semantic microservices. The proposed composition method exploits the potential data intersection on data-driven microservice descriptions to create semantic links between resources and therefore provide a navigable view of the whole microservice architecture. A plain microservice architecture requires that either microservices generate links to other microservices, which is a form of coupling, or that the microservices are designed in such way that clients are not required to follow links from one microservice to another.

The proposed composition method aims to create links that correspond to object properties in a domain ontology. It uses individual matching techniques considering a formal notion of identity as defined by (FERRARA; NIKOLOV; SCHARFFE, 2011). It takes as input a set of microservice descriptions, a domain ontology and a representation of the resource that is meant to be enriched with links. However, it requires that relations between classes be semantically defined in a domain ontology.

6.1.1 Architectural Constraints

This composition method assumes a Web-based microservice architecture, in which each microservice handles a set of resource classes. These data-driven microservices must follow three architectural constraints for the composition method to be applied.

The first constraint requires that each microservice provide ways to access their managed resources given some identifying information regarding the resource. Identifying information is widely present in real world resources and it may be necessary even in APIs that fully adopt REST architectural style constraints (FIELDING, 2000) due to the need to interface with legacy systems. Examples of person identifying attributes are passport number, social security number, user login, e-mail address, etc.

With regard to the second constraint, microservices must semantically describe managed resources. It is also required to describe how to access entities from identifying data. Finally, these descriptions must be accessible to other components of the microservice architecture.

The third and final constraint is that the representations provided by microservices must allow the inclusion of hyperlinks. However, the microservices themselves are not required to include links in their resource representations. As a result of this constraint, consumers are able to distinguish between links and literal information.

It is important to notice that the proposed method does not require that microservices follow the REST principles. However, the composition method only creates and appends links to representations, and if a microservice violates REST constraints, these violations are not shadowed by the proposed method, but are exposed to consumers. Furthermore, only microservices capable of dealing with semantically enriched resources and able to provide means to access them are eligible to adopt this composition method.

6.1.2 Linkedator

his section presents Linkedator, a development tool for composing semantic microservices in agreement with the composition method described in Section 6.1. Linkedator is divided into 3 components:

Core[1], API[2] and Jersey[3]. The first component is responsible for creating links. The second one encapsulates the core functionalities into a Web API. Finally, the third component is a development tool for implementing microservices by using the reference Java technological stack for RESTful Web Services.

Linkedator-Core is the main component of the framework, responsible for creating links in JSON-LD (LANTHALER; GüTL, 2012) representations. JSON-LD is a representation format based on JSON, which provides support for linked data. The core component is composed by three modules: a service repository, an ontology model and a link engine, as shown by Figure 27. The service repository holds the semantic microservice descriptions, which should describe all the necessary details to interact with their resources. Hence, participating microservices must have their descriptions registered in this module.

The ontology model holds information about semantic classes and properties of resources managed by participating microservices. The link engine module is responsible for analyzing the ontology model, which comprises identifying object properties and creating links between entities provided by registered microservices.

There are two different methods for creating links: direct and inverse. In the direct method, a resource that is about to be linked has blank nodes containing shared information with other resources managed by different microservices. In the inverse method, this representation does not contain such information. Nevertheless, the link engine is capable of creating links to other representations based on the object properties defined in the ontology model.

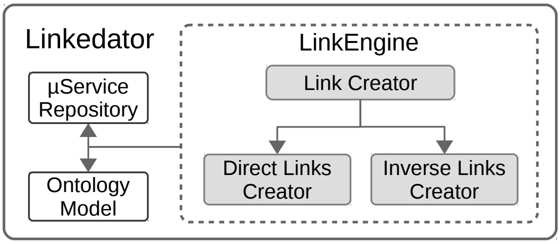


Figure 27 – Architecture of Linkedator-Core

[1]`https://github.com/ivansalvadori/linkedator`
[2]`https://github.com/ivansalvadori/linkedator-api`
[3]`https://github.com/ivansalvadori/linkedator-jersey`

Algorithm 1 Direct Links Creation Algorithm

```
 1: procedure CreateDirectLinks
 2:     $rep$ = $informed\ representation$
 3:     $objProp$ = OntologyModel.repObjProp($rep$)
 4:     for each $p \in objProp$ do
 5:         $classes$ = µServiceRepo.classes($p$.range)
 6:         for each $c \in classes$ do
 7:             $t$ = findUriTemplate($c$, $rep.objProp$)
 8:             $link$ = resolveTemplate($c$, $t$, $rep$)
 9:             $rep.p$.append(“owl:sameAs”, $link$)
10:             $rep$.append(“@type”, $e$.URI)
11:         end for
12:     end for
13: end procedure
```

Algorithm 1 shows the necessary steps to create direct links and to append them to a given representation. First, the object properties of the informed representation are identified, resulting in an array used to select suitable classes managed by registered microservices (lines 3-6). The selected classes must match the range of the identified object properties. The next step (line 7) is responsible for finding a suitable URI template that can be filled with data on the representation to identify a resource compatible with the property range. The selected URI template represents a link with variables that could be used to access representations. In line 8, the variables of the selected URI template are replaced with the informed representation data. Finally, the resulting link is associated with the property *owl:sameAs* and appended to the informed representation (line 9). The *@type* property, which defines the semantic class of the representation (line 10), is also appended to it.

Algorithm 2 describes the necessary steps to create inverse links and to append them to a representation. In this process, object properties that have domain in the informed representation class are selected, which means selecting all object properties that could be part of the informed representation. As the informed representation does not contain the intersection data, it is necessary to create new elements to represent the referenced object. Within these new elements, which represent blank nodes, both the resolved link associated with *owl:sameAs* and the *@type* property are appended.

Algorithm 2 Inverse Links Creation Algorithm

```
 1: procedure CreateInverseLinks
 2:     $rep$ = $informed\ representation$
 3:     $objProp$ = OntologyModel.objPropByDomain($rep$)
 4:     for each $p \in objProp$ do
 5:         $classes$ = µServiceRepo.classes($p$.range)
 6:         for each $c \in classes$ do
 7:             $t$ = findUriTemplate($c$, $rep$)
 8:             $link$ = resolveTemplate($c$, $t$, $rep$)
 9:             $newElement$.append(“owl:sameAs”, $link$)
10:             $newElement$.append(“@type”, $e$.URI)
11:             $rep$.append($p$.uri, $newElement$)
12:         end for
13:     end for
14: end procedure
```

Linkedator-API is a component that encapsulates Linkedator-Core into a Web API meant to be accessible for all the participant microservices. Linkedator-API exposes mainly two functionalities: register a semantic microservice description; and invoke the core component to create and append links to a given representation. It is important to notice that the Linkedator-API works only as a mediator for the Linkedator-Core; the functionalities are actually performed by the core component.

Figure 28 shows a sequence diagram that represents the processes of microservice registration and link creation. Firstly, the Linkedator-API loads the ontology file. Then, a microservice should perform an HTTP POST request for registering its description. When a given consumer interacts with a registered microservice, the microservice sends the requested representation to the Linkedator-API for creating all possible links. The Linkedator-API appends the resulting links to the representation and returns it to the microservice, which forwards it to the consumer. The link creation process is transparent to the consumer. Furthermore, new microservices are able to join and register their description at run time, resulting in more data sources and consequently more possibilities for interlinking representations.

The Linkedator-API supports additional configurations, such as link verification and caching. When the link verification is enabled, all links generated by the engine are verified. The verification is performed by executing an HTTP HEAD request to the link, which must result in

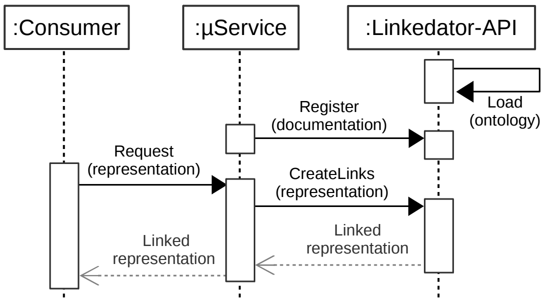


Figure 28 – Microservice description registry and link creation processes

an HTTP OK response (status code 200), otherwise the link is ignored. This verification is an assurance that the representation will be enriched only with links that are valid at creation time. When caching is enabled, the results of the link validation process are stored in a cache, avoiding to validate a link repeatedly and improving the performance of the composition as a result.

Linkedator-Jersey is a tool for developing microservices using JAX-RS, the standard technology for developing RESTful Web Services on the Java platform. Its main goal is to automatically create the microservice description by analyzing the annotations used to create endpoints implemented with Jersey – the JAX-RS reference implementation. Figure 29 shows an example of a microservice description described in Hydra format (LANTHALER, 2013). The microservice described in this example is able to manage instances of class *"http://ontology#ClassX"* defined in the domain ontology. Firstly, the path to the OWL ontology file must be informed. Then, the types of entities that the microservice is able to manage are defined as semantic resources. Semantic resources have to define at least one URI template, otherwise links cannot be created to that entity. In this example, the defined URI template is used to obtain resource representations of *ClassX* by informing an entity's property value. A microservice description must semantically define its URI templates, which implies the semantic description of the template variables. In this example, the meaning of variable *y* is defined by property *"http://ontology#propertyY"* described in the domain ontology.

Linkedator-Jersey also facilitates the interaction with Linkedator-API. By using this component, the microservice description registry is

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "http://api.example.com/doc/",
  "@type": "ApiDocumentation",
  "supportedClass": [{
    "@id": "http://ontology#ClassX",
    "@type": "IriTemplate",
    "template": "resource{?y}",
    "mapping": [{
      "@type": "IriTemplateMapping",
      "variable": "y",
      "property": "http://ontology#propertyY"
    }]
  }]
}
```

Figure 29 – Example of a microservice description

performed automatically when the service starts. The developer only has to configure the address of the Linkedator-API in the configuration file. The second role of the component is to automatically intercept all consumer requests and transparently invoke the Linkedator-API to create links in representations served to consumers.

6.1.3 Ontology Alignment

A monolithic application usually provides several features through a single software artifact. When a monolithic application is used to manage the lifecycle of entities, it usually manages several types of entities in a given domain. In this scenario, there is no data integration problem, since all information is managed by a single provider. On the other hand, the adoption of a microservices architecture results in a separation of such features into several independent small services. In this scenario, each microservice manages a subset of entities that may require combining entities provided by other microservices in order to perform the same features that the monolithic application is capable of. One can argue that each microservice should be able to perform a complete business goal; thus, there is no need for composing them. This allegation takes into account only the originally designed perspective. However, when reusing a microservice in a different context, there is no guarantee that a given goal can be fulfilled by only one microservice.

In addition to the single responsibility principle, which leads to fine-grained interfaces, microservices take advantage of independent development, leading to distinct implementations and entity models.

These characteristics pose challenges on how to compose them to fulfill business goals. One possible solution is adopting Semantic Web technologies to provide machine-readable descriptions of the data managed by a microservice, as well as interaction details. Although the adoption of semantic technologies leverages data integration, the independence of modeling and development could minimize their expected benefits. For instance, microservices may be semantically described by heterogeneous ontologies, which result in data interpretation problems. Ontology alignment techniques can be adopted to tackle this issue, since they aim at figuring out equivalent concepts among different ontologies.

In the context of microservices, ontology alignment has distinct characteristics that differ from the traditional problem, such as: entities are provided through a Web interface and the necessary interaction information may also require alignment before usage. However, the most important difference from the traditional ontology alignment is that equivalence statements are obtained based on entities resulted of the interaction between microservices and their consumers, since it is not possible to directly access a dataset. This work addresses the problem of putting such diverse concepts together to create an integrated model that allows the access and understanding of the information provided by several microservices described by heterogeneous ontologies. The integrated model is created by applying ontology matching techniques specifically adapted to the context of data-driven microservices.

In general, it is not realistic to assume that data provided by services will always be defined by a single ontology (FELLAH; MALKI; ELçI, 2016), specially when they are developed by independent teams to fit distinct application (sub-)domains. The problem of distinct ontologies can be solved with an alignment between the classes and properties of the heterogeneous ontologies Pavel & Euzenat (2013).

In order to address this issue, it is proposed Alignator[4], a tool for aligning heterogeneous ontologies used to describe the information managed by data-driven microservices. Alignator aims at finding out alignment triples among several ontologies considering data entities described by them, allowing the creation of semantic links between resources managed by microservices in cross-domain scenarios.

Alignator relies on the existence of intersection between data exposed by different services. Specifically, Alignator exploits the property values shared between entities. As an example, consider the description of a person, exposed by μS_A where each person has a `foaf:name`, and the description in another service (μS_B) where the name attribute

[4]`https://github.com/ivansalvadori/alignator-core`

is present as `taxpayer:fullName`. If µS_B exposes a query interface for taxpayer name, then a correferent can be found with a `foaf:name` from µS_A. Such type of data sharing is justifiable in microservices, as maintaining links between the data in µS_A and µS_B would incur a certain level of coupling between the interfaces and deployment of the microservices.

In this scenario, the semantic equivalence between `foaf:name` and `taxpayer:fullName` is not known. In fact, it is not possible to differentiate `taxpayer:fullName` from `foaf:name`. Yet, the intersection can still be exploited by blindly obtaining potential related entities from µS_B based on the values of attributes of a Web Resource obtained from µS_A. The set of related resources can be used to feed extensional ontology matchers (PAVEL; EUZENAT, 2013); thus obtaining, among others, the equivalence between `foaf:name` and `taxpayer:fullName`.

Alignator Architecture. Alignator aims at finding out alignment triples among several ontologies considering microservice Web resources described by them. It is divided into four main components, as depicted in Figure 30. The first component is the µ*Service Description Repository*, which stores documents that describe interaction details of registered microservices. The second component is a µ*Service Entity Loader*, which is capable of obtaining related Web resources from registered microservices based on a sample. The third component, the *Ontology Manager*, is responsible for managing ontologies used by registered microservices and their corresponding loaded Web resources. Finally, the fourth component is the *Ontology Matcher*, designed to find equivalent semantic properties and classes. The resulting alignments produced by Alignator are then considered by a data consumer interested in accessing Web resources semantically described by different ontologies and managed by different microservices. Alignator may also be adopted directly by microservices or middlewares, as previously shown by Salvadori et al. (SALVADORI et al., 2017b).

The µService Repository stores semantic descriptions, which describe Web resources that a microservice is able to manage and how to obtain them. Alignator adopts Hydra documentation as the default format for representing microservice descriptions, as previously presented in Figure 29. However, this module is open to accept other service documentation formats.

The µService Entity Loader is responsible for loading related Web Resources from registered microservices. It is able to access the µService Description repository and execute HTTP requests based on URI templates. When a Web resource is obtained from a given mi-

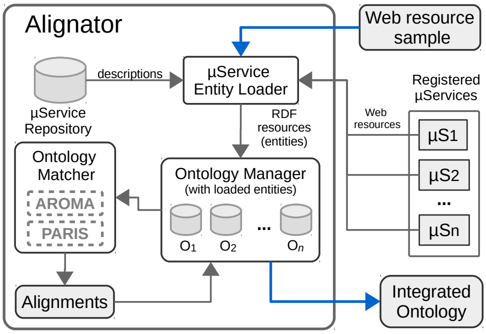


Figure 30 – Alignator architecture

croservice, it is converted to an RDF resource and added as an entity to an integrated ontology. This component plays an important role, since the ontology matching process is better performed when not only ontological concepts are defined, but also when entities are considered. Then, when a Web resource is loaded, the Ontology Manager is invoked to add the loaded Web resources into the corresponding ontology.

The Ontology Manager is responsible for managing all ontologies used by registered microservices. It is also responsible for creating an integrated ontology based on alignments produced by the Ontology Matcher. It holds not only ontological concepts, but also manipulates Web resources provided by microservices and loaded by the µService Entity Loader, keeping independent repositories for each domain ontology. There is also an entity number control mechanism (ENCM) that allows setting the maximum number of entities in each ontology. It means that when an ontology achieves the threshold, the Ontology Manager flushes the entities of the ontology so as to reduce memory consumption.

Finally, the Ontology Matcher is at the heart of Alignator. It takes a set of ontology files as input and produces alignment statements. The ontologies are described in OWL and the resulting alignments are represented through the use of *owl:equivalentProperty* and *owl:equivalentClass* predicates. Typically, ontology matchers are not expected to align simultaneously more than two ontologies. Due to this limitation, ontologies are combined in pairs, resulting in k-combinations defined by $C(O_n, 2)$, where n is the number of ontologies.

Currently, the ontology matcher component can use one of two ontology matchers. One of them is AROMA (Association Rule Ontology Matching Approach) (DAVID, 2007). AROMA employs extensional techniques to analyze the set of instances of entities in order to compute the correspondences and obtain the alignment between different ontologies. The approach used by AROMA allows to match both equivalence relations as well as relations between classes and properties of ontologies.

Another adopted ontology matcher is PARIS (Probabilistic Alignment of Relations, Instances and Schema) (SUCHANEK; ABITEBOUL; SENELLART, 2011). PARIS iteratively computes alignments for individuals, properties and classes, which are counted in subsequent iterations. Instance alignment locates a property shared by the ontology that acts as a highly inverse functional property and for which the two correferent individuals (subjects) share the same object. For subclass relations, the probability that $c_1 \subseteq c_2$ is defined by the number of instances of $c_1 \cap c_2$ in proportion to the number of instances of c_1. Similarly, for sub-properties, it assumes that the probability of $p_1 \subseteq p_2$ is related to the number of subject-object pairs occurring with both p_1 and p_2 in proportion to those with p_1.

Dynamic Perspective. In order to provide a suitable explanation of how Alignator works along with its components and external actors, a sequence diagram is depicted in Figure 31. The starting point is the semantic description registry, where descriptions and ontologies used by microservices are sent to the Alignator framework to indicate that those microservices are participating members. During this process, Alignator registers both the semantic description and the ontology of a microservice separately into the µService Repository and the Ontology Manager, respectively.

The ontology matching process starts when a client sends a Web resource sample to Alignator, which in its turn asks the Entity Loader to obtain all possible related Web resources from registered microservices. Firstly, the Entity Loader obtains the semantic descriptions managed by the µService Repository. Then, based on those descriptions, URI templates are processed with all their input parameters replaced by the values extracted from the sample. This process may result in several invalid URIs, which will return an HTTP 404 status code. However, any Web resource that result from this process are forwarded to the Ontology Matcher to perform the matching process and figure out new alignment triples. The resulting alignment triples are sent to the Ontology Manager to be incorporated in the integrated ontology, which contains the concepts regarding all registered ontologies, as well as their equivalent relations.

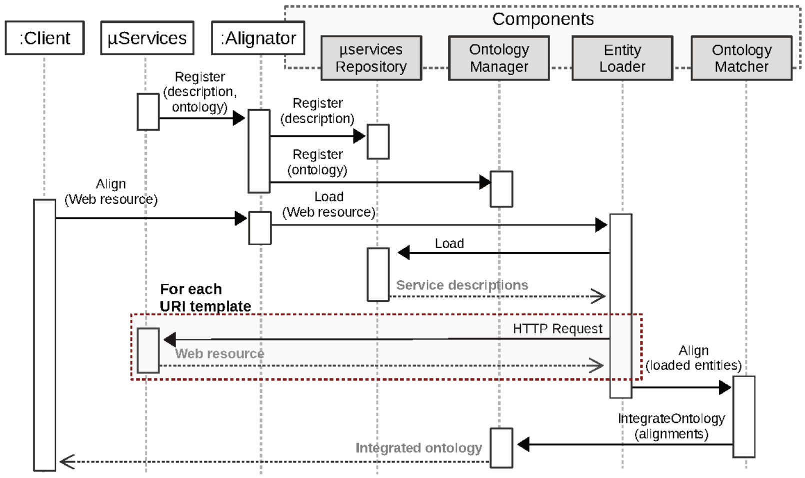


Figure 31 Alignator - sequence diagram

6.1.4 Alignator as an Internal Module

The use of Alignator is not necessary attached to sdd-µs or to the DLaaS infrastructure. Therefore, it can be used by any data provider that follows its design constraints. In the context of DLaaS, Alignator is used as Linkedator internal module, in which the Link Engine sends the same Web resource informed by the data consumer to Alignator that outputs the resulting integrated ontology, as shown in Figure 32.

In order to clarify Alignator's functionality, Figure 33 shows a sequence diagram representing the process of creating alignments along with Linkedator. Firstly, microservices send an HTTP request to register their descriptions and domain ontologies in Linkedator, which also registers them in Alignator's repository. The link creation process starts when a given consumer performs an HTTP request to Linkedator containing an entity meant to be enriched with links. After adding all possible links, Linkedator forwards the requested Web resource as input to Alignator so as to load related Web resources. Then, Alignator invokes all registered microservices according to URI templates, replacing their parameters with values contained in the requested Web resource. As a result, Alignator loads a set of Web resources from microservices, which are used as input to find out alignments. All the resulting alignments are intended to be informed to Linkedator through an inte-

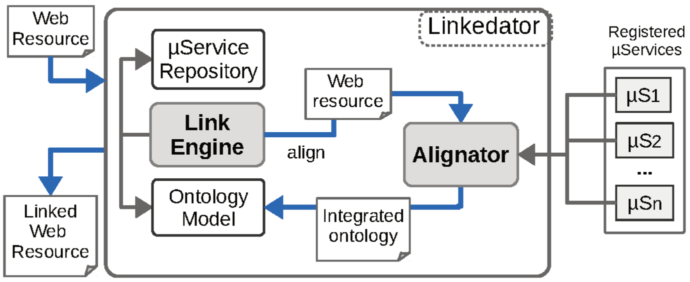


Figure 32 Alignator as a internal module

grated ontology, which should consider new alignment statements for forthcoming consumer requests. All links created by considering equivalent properties are associated with the property *rdfs:seeAlso*, which indicates that additional information might be provided by following that link.

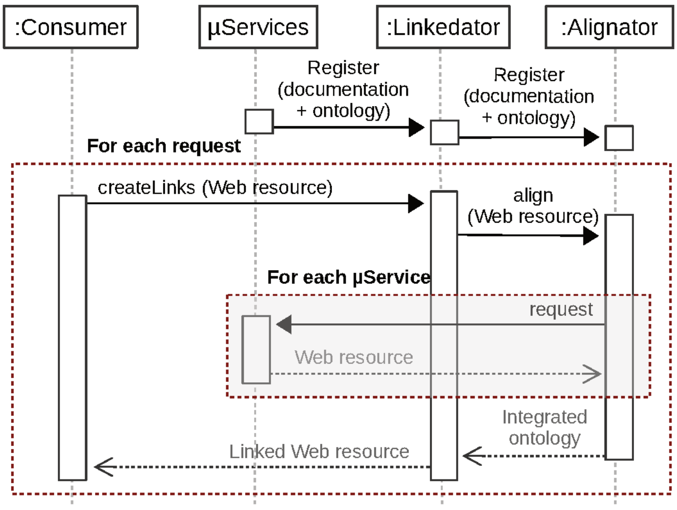


Figure 33 Alignator as a module - sequence diagram

6.2 LITERAL TO RESOURCE CONVERSION

Literal values often provide human-friendly information. For example, *rdfs:label* is used to provide a human-readable name for a given Web resource, whereas a URI is used to provide a machine-readable name. Literal nodes are not generally useful for machine understanding and are not the primary aim of the Semantic Web, which is incorporating structured information into the Web. However, some literal nodes may contain useful information (MEYMANDPOUR; DAVIS, 2016). For example, a *rdfs:label* property may represent the name or the birth date of a person that is being described through a Web resource

A disadvantage of using literals to represent Web resource information is the lack of findability, which represents the ability to find resources under the current LOD publication paradigms (BEEK et al., 2017). In fact, it is possible to represent literal information by associating a given property with a suitable Web resource URI that describes the same real-world object. This association results in a higher level of Web resource connectivity as well as improves the potential data reuse. DLaaS provides means to automatically convert literals into Web resources. This conversion is performed by a specific-purpose module described in the following subsection.

6.2.1 L2R

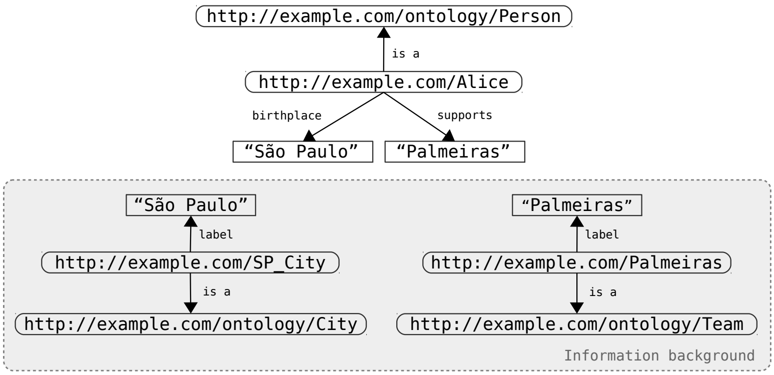
Literal to Resource (L2R) is a tool for converting literal values into independent and accessible Web resources. In order to properly explain this conversion, let's consider the example described in Figure 34. Considering that a given Web resource uniquely identified by the URL *http://example.com/Alice* holds the properties *birthplace* and *supports* associated with literal values *"São Paulo"* and *"Palmeiras"*, respectively. Let's assume that it is available an information background that holds a collection of Web resources meant to be used as targets for replacing literal values. As depicted in Figure 34 (a), Web resources from the information background contain literal values that are used as the reference to perform such conversion. Then, L2R is capable of performing analyses over literal values to identify matches to replace literal values with the URLs of Web resources. As shown in Figure 34 (b), the literal values *São Paulo* and *Palmeiras* are replaced by the URLs *http://example.com/SP_City* and *http://example.com/Palmeiras*, respectively. As a result, properties originally associated with literals are now referencing Web resources that hold much more meaningful information and may be linked to another Web resources, resulting in an interconnected and navigable graph, as shown in Figure 34 (c).

Disambiguation. Converting a literal into a Web resource may result in ambiguities. Figure 35 (a) shows an example where a given literal value could be replaced by two distinct types of Web resources. In this example, the literal *"São Paulo"* could be replaced by a Web resource that represents a city or a soccer team. In order to address this issue, L2R executes indexing procedures to figure out contextual information, which associates predicates and their respective ranges. Considering this example, it is possible to identify that predicates *birthplace* and *supports* are associated with resources that are instances of *http://example.com/ontology/City* and *http://example.com/ontology/Team*, respectively. Based on this contextual information, it is possible to disambiguate this literal value and properly replace it with the correct Web resource, as shown by Figure 35 (b). It is important to mention that L2R is capable of disambiguating literal values taking into account hierarchical classes, such as sub-classes definitions. However, the disambiguation process results are restricted to the background information quality as well as its resulting contextual information. In other words, it is only possible to disambiguate a given literal value if the background information is rich enough to provide the necessary context.

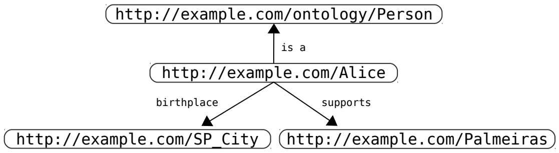
L2R Architecture. L2R is composed by three internal components: Converter, Indexer and Contextualizer. Initially, the background information is indexed by the Indexer in a map data structure, as shows step (1) in Figure 36. The current implementation employs MapDB[5], which permits the creation of data collections backed by off-heap or on-disk storage. Then, the Indexer creates a map in which the key is the literal value associated with the property *rdfs:label*, and the value is the Web resource URI. Once indexed, the information background is passed as input to the Contextualizer, which creates two maps. The first one associates predicates with resource URIs. The second one associates resource URIs and their respective semantic classes.

After the indexing process, an L2R client would be able to request a conversion of a given Web resource. As shown in Figure 36, during step (2) L2R accepts as input a collection of triples in which at least one object value should be represented as a literal value and outputs triples in which these literals are replaced by Web resource URIs, as shown in step (3).

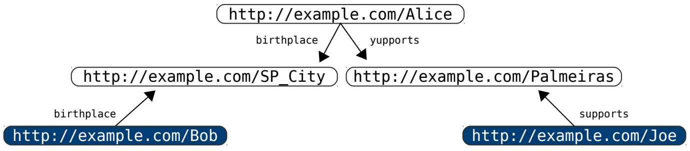
[5]http://www.mapdb.org/


http://example.com/ontology/Person
is a
http://example.com/Alice
birthplace
supports
“São Paulo”
“Palmeiras”
“São Paulo”
label
http://example.com/SP_City
is a
http://example.com/ontology/City
“Palmeiras”
label
http://example.com/Palmeiras
is a
http://example.com/ontology/Team
Information background

(a)

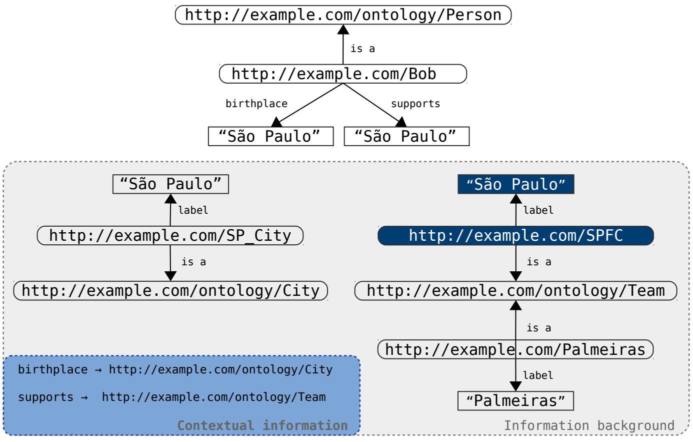

http://example.com/ontology/Person
is a
http://example.com/Alice
birthplace
supports
http://example.com/SP_City
http://example.com/Palmeiras

(b)



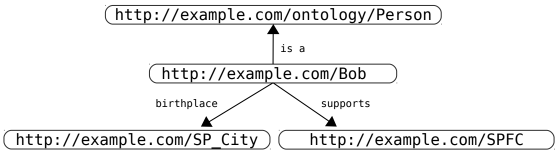
(c)

Figure 34 – L2R convertion: (a) Web resources with literal values, (b) converted Web resources (c) interconnected graph



http://example.com/ontology/Person
is a
http://example.com/Bob
birthplace
supports
“São Paulo”
“São Paulo”
“São Paulo”
label
http://example.com/SP_City
is a
http://example.com/ontology/City
“São Paulo”
label
http://example.com/SPFC
is a
http://example.com/ontology/Team
is a
http://example.com/Palmeiras
label
“Palmeiras”
birthplace → http://example.com/ontology/City
supports → http://example.com/ontology/Team
Contextual information
Information background

(a)



(b)

Figure 35 – L2R disambiguation: (a) Web resources with ambiguity, (b) converted Web resources

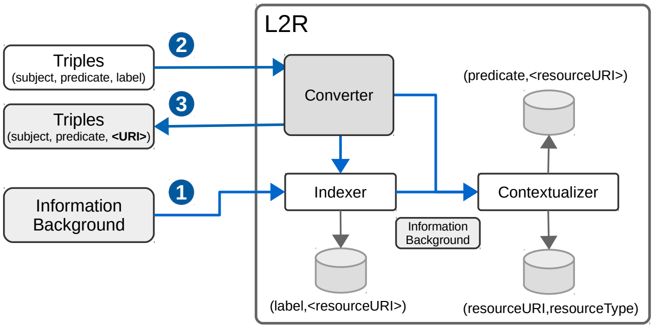


Figure 36 – L2R architecture

Algorithm 3 shows the steps to convert literal values into Web resource URIs. First, for all informed triples, URI candidates are loaded based on the current object, which represents a literal value (line 3). In the next step (line 4), all valid types of classes are loaded for a given predicate. Then, all URI candidates are tested against the loaded context to disambiguate literals. Finally, the resulting URI candidates are selected as objects of new triples composed by the current subject and predicate (line 7), replacing the literal value previously represented by $t.object$. It is worth mentioning that L2R is a free and open source software that may be used independently of the DLaaS infrastructure. The reference implementation is available in a public repository[6].

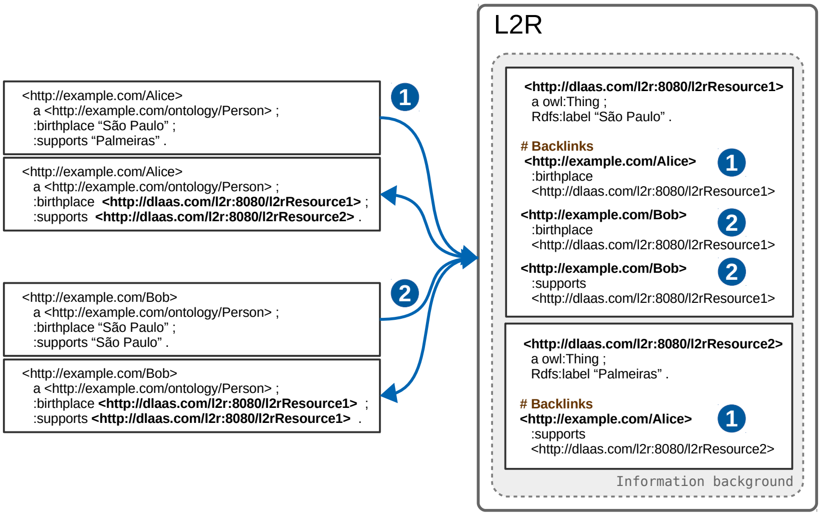
Automatic creation of background information. L2R is able to create the background information automatically. This feature assumes that there is no background information available as reference for converting a given literal value into a Web resource. Therefore, L2R will create new Web resources based on consumer inputs. In this scenario there is no disambiguation process, and the background information is created during request time. Each request for conversion provides entries for the background information.

Figure 37 shows an example in which the literal to resource conversion is performed without predefined background information. In 37 (1), a Web resource is sent to L2R to get their literal values replaced by resource URIs. This Web resource has two literals associated with properties, which results in the creation of two Web resources. Additionally, it creates backlinks or backtrack links to properly connect

[6]https://github.com/ivansalvadori/l2r

Algorithm 3 L2R Conversion Algorithm

```
1:  procedure Convert(triples)
2:      for each $t \in triples$ do
3:          $uriCandidates$ = indexer.load($t.object$)
4:          $validTypes$ = contextualizer.load($t.predicate$)
5:          for each $c \in uriCandidates$ do
6:              if $validTypes$.contains($c$) then
7:                  $convertedTriples$.add($t.subject$, $t.predicate$, $c$)
8:              end if
9:          end for
10:     end for
11:     return $convertedTriples$
12: end procedure
```



```
<http://example.com/Alice>
   a <http://example.com/ontology/Person> ;
   :birthplace "São Paulo" ;
   :supports "Palmeiras" .
```

```
<http://example.com/Alice>
   a <http://example.com/ontology/Person> ;
   :birthplace  <http://dlaas.com/l2r:8080/l2rResource1> ;
   :supports  <http://dlaas.com/l2r:8080/l2rResource2> .
```

```
<http://example.com/Bob>
   a <http://example.com/ontology/Person> ;
   :birthplace "São Paulo" ;
   :supports "São Paulo" .
```

```
<http://example.com/Bob>
   a <http://example.com/ontology/Person> ;
   :birthplace <http://dlaas.com/l2r:8080/l2rResource1>  ;
   :supports <http://dlaas.com/l2r:8080/l2rResource1>  .
```

```
<http://dlaas.com/l2r:8080/l2rResource1>
  a owl:Thing ;
  Rdfs:label "São Paulo" .

# Backlinks
<http://example.com/Alice>
  :birthplace
  <http://dlaas.com/l2r:8080/l2rResource1>

<http://example.com/Bob>
  :birthplace
  <http://dlaas.com/l2r:8080/l2rResource1>

<http://example.com/Bob>
  :supports
  <http://dlaas.com/l2r:8080/l2rResource1>
```

```
<http://dlaas.com/l2r:8080/l2rResource2>
  a owl:Thing ;
  Rdfs:label "Palmeiras" .

# Backlinks
<http://example.com/Alice>
  :supports
  <http://dlaas.com/l2r:8080/l2rResource2>
```

Figure 37 – L2R - automatic creation of background information

the just created Web resources with other resources in which its URI is an object. In Figure 37 (2), another Web resource is sent to L2R. However, this time it reuses a previously created Web resource to replace the literal. Despite this second request be resulted in ambiguity, it provides a more interconnected graph, consequently more reusable.

6.3 MULTI-AGENT SYSTEMS

Despite DLaaS being designed to support read-only data access, it provides support for dynamic events. New datasets may be submitted at runtime, consequently influencing interlinking outcomes between Web resources. Additionally, data owners may change semantic definitions by adding new concepts and changing or removing existing ones. Each agent is responsible for handling a set of specific dynamic events that may require establishing communication with other agents. Then, DLaaS adopts multi-agents systems to tackle dynamic features regarding changes in the infrastructure.

Agents were implemented to manage changes regarding three distinct aspects: (i) new data set submissions; (ii) domain ontology and; (iii) background information. Any modification of these aspects is automatically managed by their respective agents. Then, data owners have no direct control over how modifications are handled. However, customized procedures may be developed as a new JENA agent on these three aforementioned aspects, which turns DLaaS flexible to fulfill dynamic features not previously implemented.

Figure 38 shows how agents are spread over the DLaaS infrastructure. There are four different types of agents: DLaaS Agent, Linkedator Agent, L2R Agent, and sdd-µs Agent. The DLaaS agent is

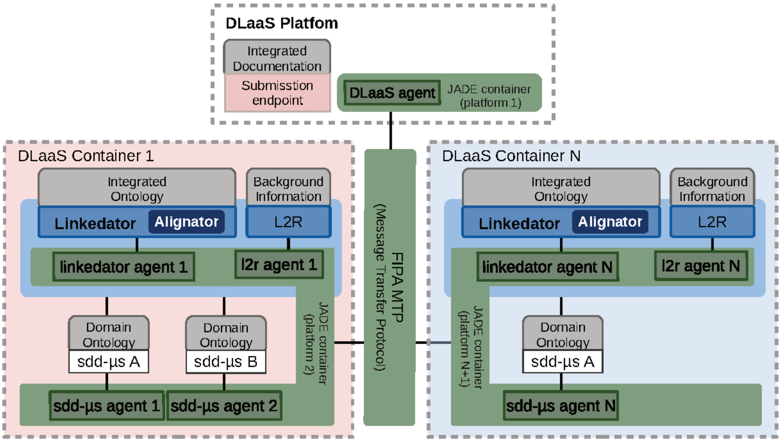


Figure 38 DLaaS agents

responsible for maintaining the integrated Hydra documentation about all deployed sdd-µs instances. Linkedator agents are responsible for transpassing Linkedator requests over DLaaS containers. L2R agents are responsible for managing background information databases. For each sdd-µs instance it is allocated an sdd-µs agent, which becomes responsible for managing conceptual changes. DLaaS platforms and containers have the JADE agent platform, presented in Section 2.6, previously configured to host these aforementioned agents, which result in a distributed agent architecture. DLaaS adopts MTP protocols to allow agents deployed in several JADE platforms to interoperate. Therefore, MTP promotes interoperability between different JADE and non-JADE platforms according to messaging standards defined by FIPA-ACL.

DLaaS Agent. Its main goal is to maintain an integrated Hydra Documentation to describe all Web resource types managed by the entire infrastructure. This agent implements a single behavior described as follows:

`Behavior:` CallForNewService
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.INFORM
`Message Content:` sdd-µs Documentation
`Origin:` sdd-µs Agent
`Outcome:` Updates the integrated Hydra Documentation

Linkedator Agent. Its main goal is to perform the explicit semantic definition service composition across multiple DLaaS containers. This agent implements five behaviors. The first one aims at receiving messages from other Linkedator agents. The second is intended to request other Linkedator agents to enrich a given Web resource with links. When a given sdd-µs requests the Linkedator that is deployed in the same container, the Linkedator agent intercepts the request and propagates it to other Linkedator agents deployed in other containers. Then, the agent interacts directly with Linkedator to request its services. Once the request is concluded, the agent returns the linked Web resource to the requesting Linkedator agent.

The third behavior aims at receiving messages from sdd-µs agents about semantic relationship updates. The fourth one is intended to receive messages from other Linkedator agents about ontology alignment updates. finally, the last one aims at informing other Linkedator agents about new alignment originated from its internal module Alignator. These behaviors are described in the following.

`Behavior:` CallForExplictSemanticLinking
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.REQUEST
`Message Content:` Web resource representation
`Origin:` Linkedator Agent
`Outcome`: Returns a linked version of the informed Web resource

`Behavior:` RequestForExplictSemanticLinking
`Type:` Generic Behaviour
`Performative Act:` ACLMessage.REQUEST
`Message Content:` Web resource representation
`Origin:` Linkedator
`Outcome:` Returns a linked version of the informed Web resource

`Behavior:` CallForOntologyUpdate
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.INFORM
`Message Content:` RDF triples
`Origin:` sdd-µs Agent
`Outcome:` Updates its respective domain ontology

`Behavior:` CallForOntologyAlignment
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.INFORM
`Message Content:` RDF triples
`Origin:` Linkedator Agent
`Outcome`: Updates its respective domain ontology

`Behavior:` InformOntologyAlignment
`Type:` Generic Behaviour
`Performative Act:` ACLMessage.INFORM
`Message Content:` RDF triples
`Origin:` Linkedator Agent
`Outcome:` None

L2R Agent. Its main goal is to perform the literal to resource conversion across multiple DLaaS containers. It implements two behaviors. The first one aims at receiving messages from other agents. The second behavior is intended to request other L2R agents to convert literal values of a given Web resource into Web resource URLs. Similarly to the Linkedator Agent, when a given sdd-μs sends a request to the L2R that is deployed in the same container, the L2R agent intercepts the request and forwards it to other L2R agents deployed in other containers. Then, each agent interacts directly with its instance of L2R to request its services. Once the request is concluded, the agent returns the converted Web resource to the requesting L2R agent.

`Behavior:` CallForLiteralToResourceConversion
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.REQUEST
`Message Content:` Web resource representation
`Origin:` L2R Agent
`Outcome`: Returns a converted version of the informed Web resource

`Behavior:` RequestForLiteralToResourceConversion
`Type:` Generic Behaviour
`Performative Act:` ACLMessage.REQUEST
`Message Content:` Web resource representation
`Origin:` L2R
`Outcome:` Returns a converted version of the informed Web resource

sdd-μs Agent. Its main goal is to propagate the changes about its sdd-μs domain ontology over multiple DLaaS containers. It implements two behaviors. The first one aims at receiving messages from other agents. The second behavior is intended to inform other sdd-μs agents that a given domain ontology has been updated. Additionally, this behavior interacts with Linkedator Agents to inform changes on semantic relationship definitions. As a result, multiple services that provide Web resources of the same domain can benefit from updated concepts.

`Behavior:` CallForOntologyUpdate
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.INFORM
`Message Content:` RDF triples
`Origin:` sdd-μs Agent
`Outcome`: Updates its respective domain ontology

`Behavior:` InformOntologyUpdate
`Type:` Cyclic Behaviour
`Performative Act:` ACLMessage.INFORM
`Message Content:` RDF triples
`Origin:` Domain ontology
`Outcome:` None

6.4 FINAL CONSIDERATIONS

This chapter presented implementation details on development and support tools for data linking strategies with a data-driven microservice composition perspective. Linkedator was presented as an implementation solution for performing the explicit semantic definition data linking strategy. Alignator was presented as a support tool for identifying alignments of heterogeneous ontologies, allowing compositions in cross-domain environments. L2R was proposed as a support tool for converting literal values into Web Resources. Finally, a multi-agent system based on the JADE platform was presented as a communication mechanism for linking Web resources of the entire DLaaS infrastructure, providing support for dynamic events.

Linkedator connects Web resources managed by multiple microservices without introducing coupling between services. Alignator allows Web resources described by different ontologies to be connected with one another. L2R is a tool for converting literal values into independent and accessible Web resources. It is capable of dealing with ambiguities as well as creating its own background information dataset. DLaaS agents were introduced for dealing with dynamic changes in the infrastructure. They are able to manage a unified Hydra documentation, to spread semantic definition modifications, to inform new microservices that have joined the infrastructure and request linking and literal to resource conversion across multiple containers.

7 INTER-SERVICE LINKING EVALUATION

According to Tosi & Morasca (2015), researchers mainly keep their efforts in defining new ontologies and tools, instead of the real implementation of Semantic Web Services. As a result, proposals are kept at an abstract level that does not help the wide adoption of these technologies. On the contrary, this chapter employs concrete implementations and tests them using real-world datasets to evaluate the proposed inter-service linking strategies.

This chapter is organized as follows. Section 7.1 evaluates the Explicit Semantic Definition strategy implemented by Linkedator. Section 7.2 presents an evaluation of ontology alignment algorithms adopted by Alignator. Section 7.3 evaluates Alignator as an internal module of Linkedator. Section 7.4 evaluates the Literal to Resource conversion strategy implemented by L2R. Finally, Section 7.5 evaluates the use of multi-agent systems for implementing dynamic features as well as a communication mechanism for DLaaS components.

7.1 LINKEDATOR EVALUATION

7.1.1 Case Study

In order to show how the Explicit Semantic Definition strategy can be applied using Linkedator, this section presents a case study based on criminal, financial and immigration records. Some technical details on the use of Linkedator are also presented in this section.

The domain ontology, summarized in Figure 39, contains four classes: *Person, Financial Transaction, Criminal Record* and *Immigration Record.* To save space, data properties of the classes are shown as light gray rectangles attached to the classes, represented by ellipses; and object properties are represented by directed edges. $Person$ acts as a central class whose instances may be related to the other three classes, which, in turn, have properties in the reverse direction. To avoid coupling, object properties are not stored as links to resources in other microservices. Therefore, such links are created by Linkedator.

The architecture of the case study is shown in Figure 40. Instances of $Person$ are fully handled by $\mu Service1$. Instances of the other three classes are handled by different providers, each one represented by a different microservice. Four instances – from $\mu Service2$

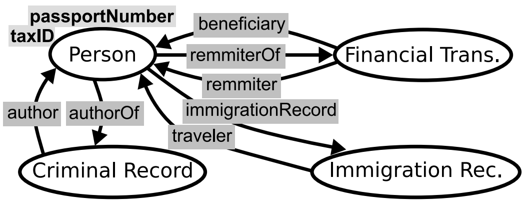


Figure 39 – Simplified case study ontology

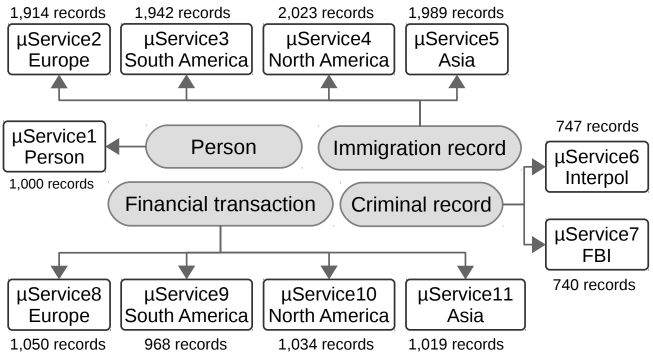


Figure 40 – Microservice and data source details - case study

to µ*Service*5 – manage immigration data from different continents. µ*Service*6 and µ*Service*7 manage criminal records from two providers, respectively FBI and Interpol. Finally, instances from µ*Service*8 to µ*Service*11 manage financial transactions from different continents. All microservices are implemented without knowledge from one another and, therefore, the representations produced by them do not include links to related resources on other microservices. Figures 41, 42, 43 and 44 show examples of Web resources used in the case study.

```
{
  "@context": { "ontology": "http://ontology#" },
  "@type": "ontology:Person",
  "ontology:taxID": "733-11716-531-23",
  "ontology:passportNumber": "253-5022-82-43967-856",
  "ontology:firstName": "Nicole",
}
```

Figure 41 – A *Person*, from µService1

```
{
  "@context": { "ontology": "http://ontology#" },
  "@type": "ontology:ImmigrationRecord",
  "ontology:reportFrom": "ontology:Europe",
  "ontology:traveler": {
    "ontology:passportNumber": "253-5022-82-43967-856"
  },
  "ontology:declaredMoney": "1787.82",
}
```

Figure 42 – An *Immigration Record*, from µService2

```
{
  "@context": { "ontology": "http://ontology#" },
  "@type": "ontology:CriminalRecord",
  "ontology:criminalAgency": "ontology:Interpol",
  "ontology:author": {
    "ontology:taxID": "733-11716-531-23"
  },
  "ontology:registerNumber": "062-11441-05780-76",
  "ontology:crime": "Fraud"
}
```

Figure 43 – A *Criminal Record*, from µService6

```
{
  "@context": { "ontology": "http://ontology#" },
  "@type": "ontology:FinancialTransaction",
  "ontology:transactionFrom": "ontology:NorthAmerica",
  "ontology:transactionID": "581-52207-12414-84",
  "ontology:amount": "17528.46",
  "ontology:remmiter": {
    "ontology:taxID": "733-11716-531-23"
  },
  "ontology:beneficiary": {
    "ontology:taxID": "720-72890-123-53"
  }
}
```

Figure 44 – A *Financial Transaction*, from µService10

The fact that *taxID* and *passportNumber* are identifying properties of *Person* is not stated on the ontology, but is derived from the URI template descriptions given by the microservices. µ*Service*1 provides templates that given either a *taxID* or a *passportNumber* produce an URI that, if valid, identifies an instance of *Person*. The µ*Service*1's templates allow Linkedator to enrich the Web resource at Figure 44 with *owl:sameAs* links for the transaction's remitter and beneficiary, as shown in Figure 45.

In the case of *Person* Web resources, such as the example in

Figure 41, no reference remains to related resources. However, since µ*Service*2 to µ*Service*11 provide URI templates that use *taxID* and *passportNumber* as parameters, links for the object properties *immigrationRecord*, *remitterOf* and *authorOf* can be constructed by Linkedator from Figure 41. The result of this enrichment is shown in Figure 46. To keep this evaluation as brief as possible, some links are omitted.

In this case study, data was randomly generated from a predefined schema using the *Mockaroo* tool[1]. A total of 1,000 *Person* records were generated, and related instances of the other three classes were randomly generated and associated to persons using one of the two identifying data properties of *Person*: *taxID* and *passportNumber*. After the generation of a single dataset containing all persons, the data was distributed among the microservices, each microservice managing the instances which corresponded to their definition. For example, µ*Service*5 manages only data of arrivals on Asian immigration departments.

```
{
  "@context": { "ontology": "http://ontology#" },
  "@type": "ontology:FinancialTransaction",
  "ontology:transactionFrom": "ontology:NorthAmerica",
  "ontology:transactionID": "581-52207-12414-84",
  "ontology:amount": "17528.46",
  "ontology:remmiter": {
    "ontology:taxID": "733-11716-531-23",
    "@type": "ontology:Person",
    "owl:sameAs": "http://.../person?taxId=733-11716-531-23"
  },
  "ontology:beneficiary": {
    "ontology:taxID": "720-72890-123-53",
    "@type": "ontology:Person",
    "owl:sameAs": "http://.../person?taxId=720-72890-123-53"
  }
}
```

Figure 45 – A financial transaction Web resource after Linkedator enrichment

For each generated *Person* record there was a probability of 25% that the person would have no immigration record. For each person that was selected to have immigration records, a uniformly distributed random number between 1 and 20 of records were generated by randomly selecting the data attributes from lists and predetermined ranges. The same approach was used to generate criminal records and financial transactions. For criminal records, the probability of a per-

[1]http://www.mockaroo.com/

```
{
  "@context": { "ontology": "http://ontology#" },
  "@type": "ontology:Person",
  "ontology:taxID": "733-11716-531-23",
  "ontology:passportNumber": "253-5022-82-43967-856",
  "ontology:firstName": "Nicole",
  "ontology:immigrationRecord": [{
    "@type": "ontology:ImmigrationRecord",
    "owl:sameAs": "http://.../immigration/253-5022-82-43967-856"
    }, { }, { }, { }
  ],
  "ontology:authorOf": [{
    "@type": "ontology:CriminalRecord",
    "owl:sameAs": "http://.../criminal/733-11716-531-23"
    }, { }, { }, { }
  ],
  "ontology:remmiterOf": [{
    "@type": "ontology:FinancialTransaction",
    "owl:sameAs": "http://.../financial/733-11716-531-23"
    }, { }, { }, { }
  ]
}
```

Figure 46 – A person Web resource after Linkedator enrichment

son having no record was 50% and the number of records for a person was uniformly distributed between 1 and 5. The probability of a person having no financial transaction was of 25%, while the number of transactions per person was uniformly distributed between 1 and 10.

The result of this process was a single JSON document comprising all persons with nested criminal, financial and immigration records. The instances on this document were distributed across the 11 microservices of Figure 40 according to the criteria associated with each microservice, as previously discussed. After this process, $\mu Service1$ outputs Web resources describing $Person$, such as described in Figure 41, without links to instances of the other classes related to the person. On the other hand, the microservices that provide immigration, financial and criminal records produce Web resources that contain blank nodes that include one of the person's identifying data properties. An example can be seen in Figure 42, where the property *traveler* instead of having a link to a $Person$, has a blank node with a *passportNumber*. The blank node represents an anonymous individual, which Linkedator later identifies as being the same individual as a person managed by $\mu Service1$.

7.1.2 Methodology

In this section we describe our experimental methodology and results. The objective of this evaluation is to find out which factors influence the response time regarding the adoption of the proposed composition method and of Linkedator. In order to allow the replication of experimental results, source code and instructions for setting up this evaluation are available in a public repository[2].

The evaluation is performed through an evaluation question, which aims at finding potential money launderers based on the data managed by microservices that compose the case study. It is important to say that data used in this evaluation is hypothetical and is not related to real personal records held by an information repository. In order to classify a given person as a potential money launderer, its record must fulfill three characteristics: total amount of financial transactions must be equal or greater than a million dollars, total of declared money on immigration records must be equal or greater than a hundred thousand dollars, and must have a previous criminal record of money laundering.

The components of the case study presented in the previous section have been deployed into the Amazon EC2 (Elastic Compute Cloud) environment. Twelve machine instances have been created, a dedicated instance for each microservice, where eleven instances have been used for each microservice implemented in the case study and a dedicated instance for Linkedator-API. The selected instance configuration was m3.medium, with Intel Xeon E5-2670 v2 (Ivy Bridge) processors running at 2.6 GHz, 3.75 GiB of memory, running Ubuntu Server 14.04 LTS (HVM) and Oracle Java Development Kit (JDK) 8. The Java Virtual Machine (JVM) was configured to allocate an initial memory pool of 64 MB, with an upper limit of 128 MB.

In order to answer the evaluation question, a client application has been developed to interact with microservices. Four different scenarios have been created. In the first scenario there is no microservice composition. Therefore, the client application must know implementation and deployment details, such as URI templates and server addresses. This scenario results in Web resources without links, hence the client application must directly issue HTTP requests to each microservice. In the second scenario, microservices are registered in the Linkedator-API and their Web resources are linked, but with occa-

[2]`https://salvadori.bitbucket.io/projects/phd/experiments/linkedator`

Table 6 – Experiment scenarios' results

Scen.	**Core**	**Jersey**	**Client**	**Rec. size**	**Reqs**
(A)	-	-	287.5 ms	891.5 chars	11,000
(B)	0.2615 ms	17.02 ms	290.1 ms	1,618.5 chars	11,000
(C)	61.505 ms	79.61 ms	304.1 ms	1,501.0 chars	6,484
(D)	38.364 ms	56.834 ms	276.4 ms	1,501.0 chars	6,484

sional invalid links. In this scenario, the client application only knows the URI to µ*Service*1 and is able to retrieve a list of *Person* records enriched with links to other Web resources. In the third scenario, the Linkedator-API is configured to validate links, which ensures that Web resources contain only valid links. Finally, in the fourth scenario, the Linkedator-API caches link validation results.

7.1.3 Experimental Results

The client application was executed for all scenarios described above and the results are shown in Table 6. This table presents the mean response-time/request for creating links spent in the Linkedator-Core and Linkedator-Jersey as well as the total time perceived by the client. It also presents the mean Web resource size/request and the number of requests (Reqs) the client application must execute in order to go through all necessary records for each scenario (Scen.), which are identified as:

(A) Not linked;

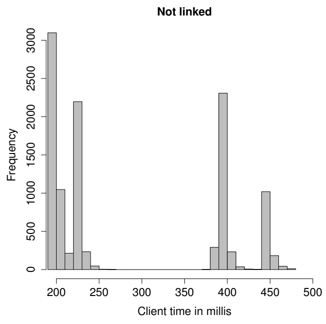
(B) With invalid links;

(C) Only valid links;

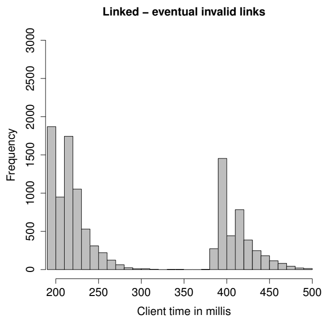
(D) Only valid links (cached).

One can notice that the link creation process takes a significant amount of time, especially when link validation is enabled. However, the link creation process does not affect the overall client time. On the other hand, the Web resource size increases 81.547% for the second scenario, which encompasses occasional invalid links, and 68.37% for the third and fourth scenarios, in which there are only valid links.

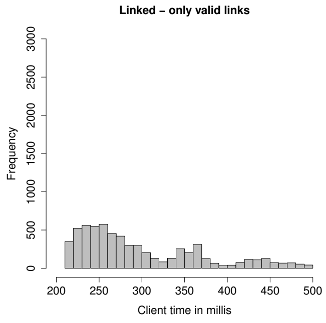
Figure 47 shows how the link creation process is perceived by the client with respect to response time distribution. Figure 47 (a)



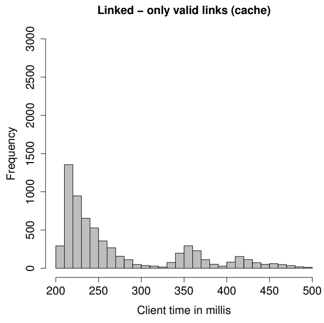
(a)



(b)



(c)

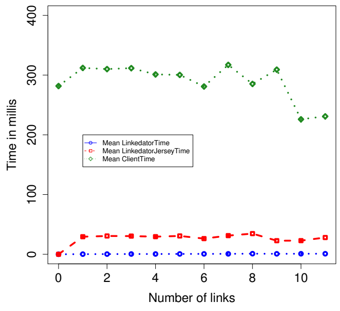


(d)

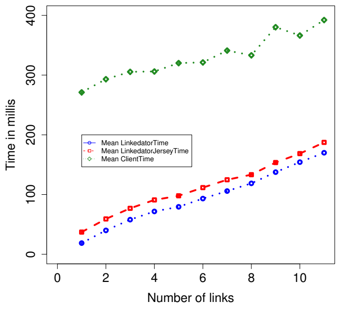
Figure 47 – Client response time distribution (a) Not linked; (b) Linked with occasional invalid links; (c) Linked with only valid links; (d) Linked with only valid links - link validation results cached.

presents the time distribution considering the first scenario (not linked resources). Most requests in this scenario are executed between 200 ms and 250 ms, followed by two spikes in 400 ms and 450 ms. Figure 47 (b) presents the time distribution considering the second scenario (linked resources with eventual invalid links). In this scenario, most requests are executed within two ranges, the first one between 200 ms and 300 ms, and the second one between 400 ms and 450 ms. Figure 47 (c) presents the distribution of requests executed in the third scenario (resources contain only valid links). One can notice that the link validation process distributes requests more evenly between 200 ms and 300 ms, reducing the spikes observed in previous scenarios. However, the use of cache, presented in (d), concentrates the execution of most requests between 200 ms and 250 ms, similar levels achieved in (b). These results show that the capability of validating links is fundamental to significantly reduce the response time perceived by consumers. This is especially important to datasets that contain a high level of missing resource correspondence. Then, Linkedator eliminates de costs that clients would have to pay to access invalid resources.

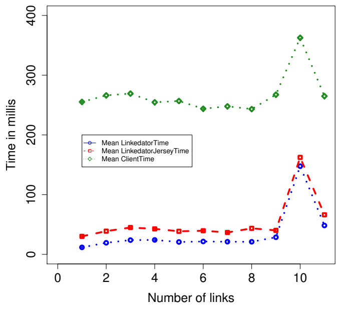
Figure 48 shows response times considering the number of links evaluated by Linkedator. One can notice that the response time of Linkedator-Jersey is similar to the response time of Linkedator-Core,



(a)



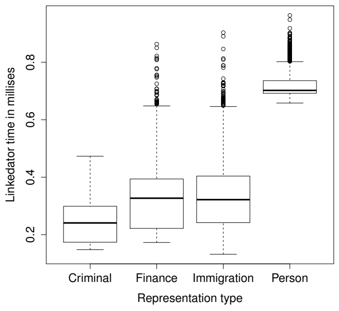
(b)



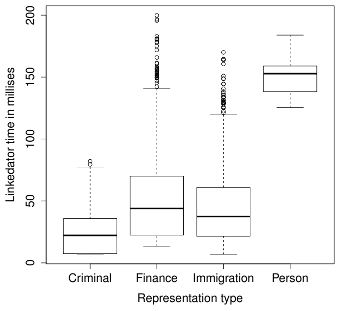
(c)

Figure 48 – Impact of number of links on response time: (a) Linked with occasional invalid links; (b) Linked with only valid links; (c) Linked with only valid links - link validation results cached.

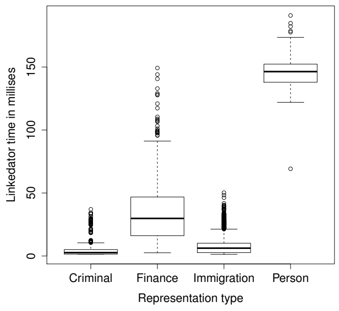
since all microservices are deployed in the same datacenter. In (b), the higher the number of links inserted into a Web resource, the higher the response time, due to the need for executing a validation request. On the other hand, in (c), with caching enabled, the increase of client response time is not perceived for most resources. However, there is a spike when Web resources contain 10 links. The reason for this is directly related to the content of those Web resources, which fits $Person$ records. In the case study, each $Person$ record is potentially linked to other 10 Web resources: four immigration records, four financial transactions and two criminal records. Each of those links is unique,



(a)



(b)



(c)

Figure 49 – Impact of resource type on response time: (a) Linked with occasional invalid links; (b) Linked with only valid links; (c) Linked with only valid links - link validation results cached.

resulting in 10 cache misses given the client behaviour. As a result, *Person* records do not benefit from caching, whereas their linked resources take advantage of reusing links previously validated, as can be seen in Figure 49. Despite the fact that in some cases the cache is not effective, validation of links inside the microservice architecture tends to be faster than validation by clients, which are often positioned at the internet edge, where round-trip time to the microservices is significantly larger.

7.2 EVALUATION OF ALIGNATOR INDIVIDUALLY

This section describes the experimental design to individually evaluate Alignator. The objective of this evaluation is to find out which factors influence the alignment strength regarding the adoption of Alignator with both AROMA and PARIS ontology matching algorithms. In order to allow the replication of experimental results, source code and instructions for setting up this experiment are available in a public repository[3].

7.2.1 Methodology

In order to evaluate Alignator, a testbench application and a microservice have been developed, as depicted in Figure 50. A synthetic dataset of 1,000 entities with characteristics to be exploited by the evaluation is managed by both the testbench application and the microservice. Each entity of the dataset contains four properties associated with alphanumeric values and a payload information. Two distinct ontologies were created to describe the entities managed by the testbench application and by the microservice. The testbench application was designed to interact with Alignator by sending each entity of the dataset as an input to the ontology matching process, considering different characteristics applied to the entities provided by the microservice. The main goal is to identify important characteristics that affect the resulting alignments in order to establish guidelines for modeling microservices.

Four factors were considered for this evaluation: A) the number of shared property values, B) the entity's payload size, C) the correspondence level between entities of the two ontologies meant to be

[3]`https://salvadori.bitbucket.io/projects/phd/experiments/alignator`

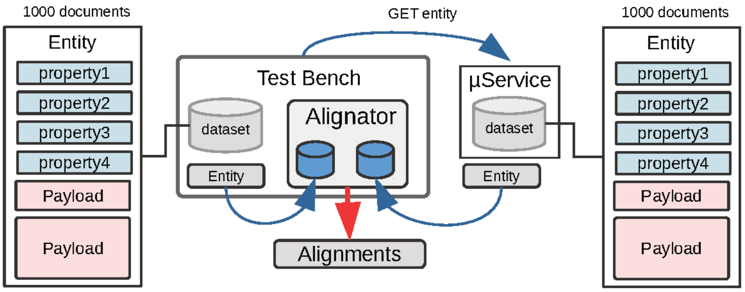


Figure 50 Alignator Evaluation scenario

Table 7 Factors and levels

	Factors	**-1**	**+1**
A	Shared property values	2	4
B	Payload size	1 word	100 words
C	Entity correspondence level	50%	100%
D	Access order	fixed	random

aligned and D) the order in which entities are sent to Alignator. These factors were considered because they can be controlled or considered in the microservices' development process in order to maximize the alignment results. As shown in Table 7, two distinct levels were associated with each factor: a low and a high level. For factor A, previous experiments showed that AROMA requires individuals to share at least two common properties in order to produce property or class alignments that could not be otherwise trivially inferred. While PARIS has no such requirement, the levels of this factor were chosen to accommodate AROMA and avoid distortions in analysis. For factor B, the lower level represents a payload with only one word, which is equivalent to one more property; and the high level is equivalent to a small text property value with 100 words. Factor C is not totally controlled by the developer, however, it is likely to be a known information and may be considered at design time. Finally, factor D exploits the order in which entities are sent to Alignator. Fixed access means that entities are sent to Alignator in the same order throughout the experiment execution, whereas random access results in a different order for each observation. Fixed access could be seen as a software agent interaction mode, and random access is closely related to a human interaction mode, in which entities are not accessed following a predefined procedure.

Factors in Table 7 describe data and access pattern. The Ontology Matchers are not considered factors. Instead, each matcher is analyzed independently, aiming to identify how it responds to the scenarios modeled by these factors, coupled with Alignator.

In order to obtain a statistical analysis of effects, a 2^k experimental design (JAIN, 1991) was adopted, in which the aforementioned factors and their levels were considered for each of the four evaluated factors. For each ontology matcher algorithm, the experiment was replicated three times to properly account for variability, resulting in 48 observations. The execution order was completely random to make sure that response variables are independently and individually distributed. The experiment was executed using the computational infrastructure provided by the Cloud Computing for Cooperation[4], with the following configuration: a dedicated server with 24 Intel Xeon X5690 processors at 3.47GHz and 148 GiB of primary memory, running the CentOS-7 operating system and openjdk version 1.8.0_141 with maximum heap size set to 2 GiB for both the microservice and the testbench application.

7.2.2 Analysis of Effects

The effects of the factors in Table 7 were analyzed with respect to the number of requests necessary until the alignment strength reported by the matcher under analysis reached a satisfactory value. Two response variables were used as possible definitions of satisfactory: $N_{0\%}$ and $N_{1\%}$, where the $N_{\delta\%}$ notation should be read as "the number of requests made by the testbench application (Figure 50) before the reported alignment strength for `ont0:o0p1 owl:equivalentProperty ont1:o1p1` had a value s such that $s - s_{max} \leq \frac{\delta}{100}$, where s_{max} is the maximum strength observed for any of the 1000 requests in the experiment." Informally, $N_{0\%}$ and $N_{1\%}$ represent the number of requests until, respectively, reaching maximum alignment strength and reaching maximum alignment strength with 1% tolerance (as for both matchers $s \in [0, 1]$).

Analysis for $N_{0\%}$. Two linear models yielding $N_{0\%}$ from the evaluated factors were adjusted to the experimental data. One model for the AROMA matcher and another for PARIS. Both homoscedasticity and normality assumptions, required by the adopted analysis method, were met by the models. Further, an adjusted R^2 of 0.9712

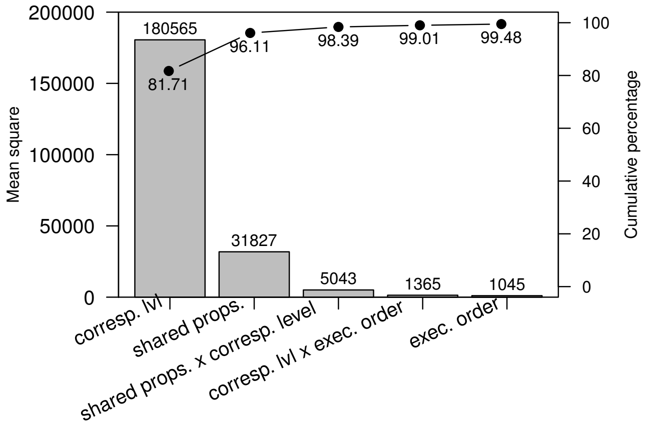
[4]http://www.c3lab.tk.jku.at

was obtained for the model corresponding to AROMA, and an adjusted R^2 of 0.6826 for PARIS. Both R^2, which denotes a good fit and, thereby, supports the conclusions on significance of effects.

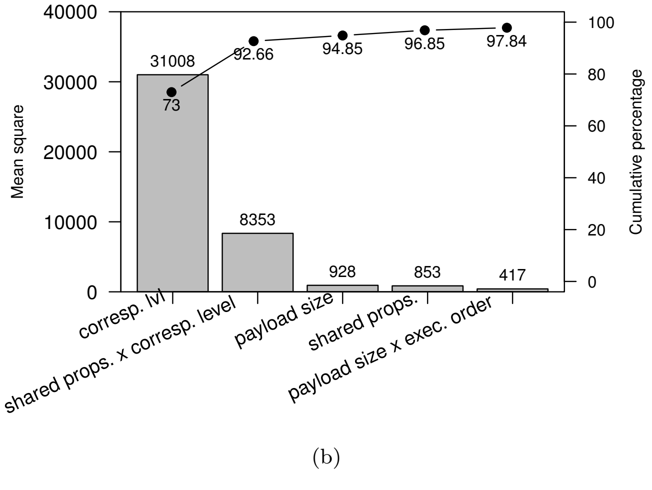
Among all considered factors and their pairwise interactions for AROMA, the following presents statistically significant effect at a level of significance of 5% ($\alpha = 0.05$): (C) entity correspondence level, (A) shared property values, (D) access order, (A+C) the interaction between shared property values and entity correspondence level, (C+D) and the interaction between entity correspondence level and access order. For PARIS, the significant factors and pairwise interactions are: C, A, D, A+C, C+D. The payload size presented no statistically significant effect on the response variable as well as other not mentioned interactions for both ontology matchers. Figure 51(a) and Figure 51(b) present, respectively for AROMA and PARIS, Pareto charts with the ANOVA mean square and the cumulative percentage of variability explained by the statistically significant factors. These charts reveal that, for both matchers, factor C (correspondence level) is responsible for almost all variability in $N_{0\%}$. Important differences, revealed by the Pareto charts, are that PARIS is more sensitive to access order (D) and less sensitive to the number of shared properties (since it requires only one shared property between individuals to infer correferences).

The different requirements of AROMA and PARIS, for the number of shared properties, also appear in Figure 52. This chart displays the effects on $N_{0\%}$ when each factor is changed from its low level (-1) to its high level (1). For AROMA, increasing the number of shared properties reduces $N_{0\%}$ from 170 to 118. This effect is not observed for PARIS, which is designed to identify correferent individuals from a single shared property.

Figure 52 reveals that the most significant factor for both matchers is the correspondence level (C). For AROMA, changing C from 50% (-1) to 100% (+1), a decrease in $N_{0\%}$ from 205 to 83 is expected. This result is intuitive, since AROMA will always be able to identify correferent individuals, and will have more evidence sooner to infer that properties `o0:o0p1` and `o1:o1p1` are equivalent. In the case of PARIS, the opposite occurs: $N_{0\%}$ is expected to increase from 993.29 to 995.75 when the correspondence level increases. PARIS identifies only `rdfs:subPropertyOf` ($\subseteq$) relations between properties, which Alignator combines using simple average to infer `owl:equivalentProperty` (=). For these experiments, in general, $Pr(\texttt{o0p1} \subseteq \texttt{o1p1}) \geq Pr(\texttt{o1p1} \subseteq \texttt{o0p1})$ when not all individuals have correspondents (through these properties). Moreover, the range on which $N_{0\%}$ is affected by any fac-



(a)



(b)

Figure 51 – Pareto charts for the $N_{0\%}$ models: (a) AROMA and (b) PARIS.

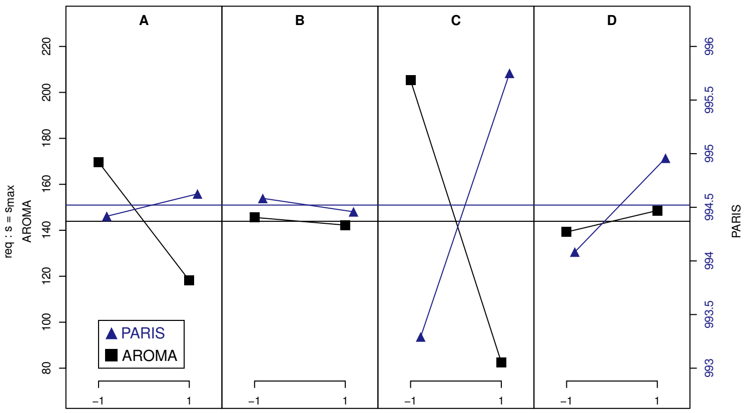


Figure 52 – Main Effects plot for response variable $N_{0\%}$ (first request with maximum strength)

tor is considerably small: from 993 to 996. This range is negligible in practical scenarios and occurs due to the significant slower rate which PARIS increases the strength of its alignments. The reader can refer to Figure 53 (a) and (c), which show alignment strength versus request number, respectively for AROMA and PARIS.

Analysis for $N_{1\%}$. As previously discussed, while it highlights behavior differences among AROMA and PARIS, $N_{0\%}$ is somewhat disconnected from a practical scenario. Using $N_{1\%}$ offers a counterpoint. Again two linear models were generated, now with an adjusted R^2 of 0.8485 for AROMA and 0.9847 for PARIS.

Among all considered factors, and their pairwise interactions, for AROMA, the following factors presented statistically significant effect at a level of significance of 5% ($\alpha = 0.05$): (C) entity correspondence level, (A) shared property values, (D) access order, (A+C) the interaction between shared property values and entity correspondence level, and (C+D) the interaction between entity correspondence level and access order. For PARIS, the significant factors and pairwise interactions are only factors C and D.

Figure 54 reveals a reverse situation from that observed for $N_{0\%}$: the range in which $N_{1\%}$ is affected is small for AROMA (from 11 to 27) and larger for PARIS. Factor C (correspondence level) remains the factor with largest impact for both aligners, and for PARIS, $N_{1\%}$ is expected to decrease from 563 to 754 when C changes from 50% to 100%.

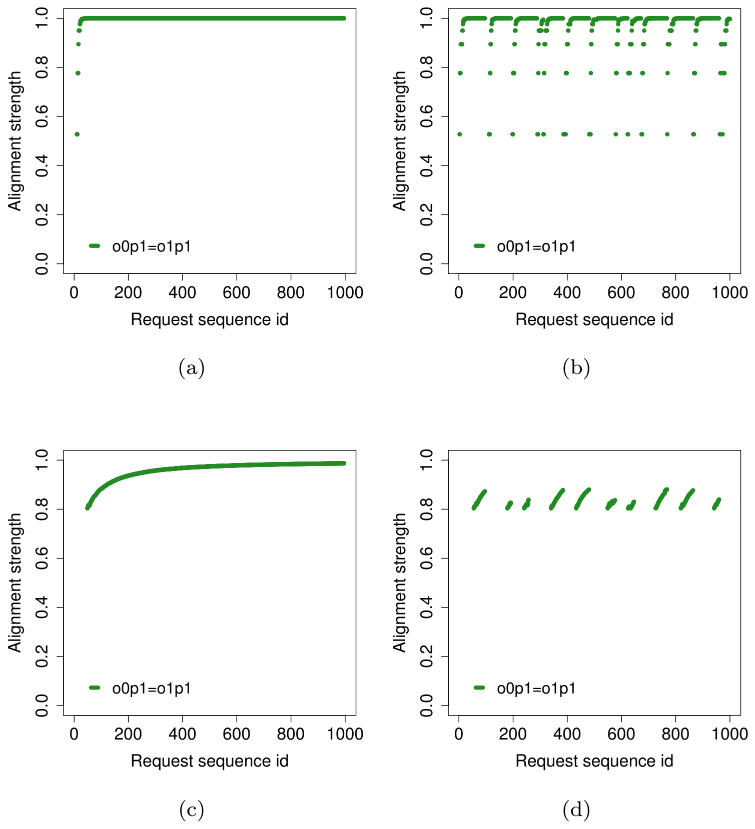


Figure 53 – Alignment strength and time of alignment per request; (a) alignment strength without ENCM - AROMA; (b) alignment strength with ENCM=1000 - AROMA; (c) alignment strength without ENCM - PARIS; (d) alignment strength with ENCM=100 - PARIS

Final remarks. Most preceding discussion centered on factors A and C. Factor B (payload size) was only significant in the model for $N_{1\%}$ using AROMA. However, the effect (decrease of $N_{1\%}$ from 20 to 18) is negligible for all practical purposes. Factor D (access order) reveals that the particular ordering of the data in the microservice is, in the four models, above average with respect to reaching high alignment strengths. However, we observe that the effects, even when

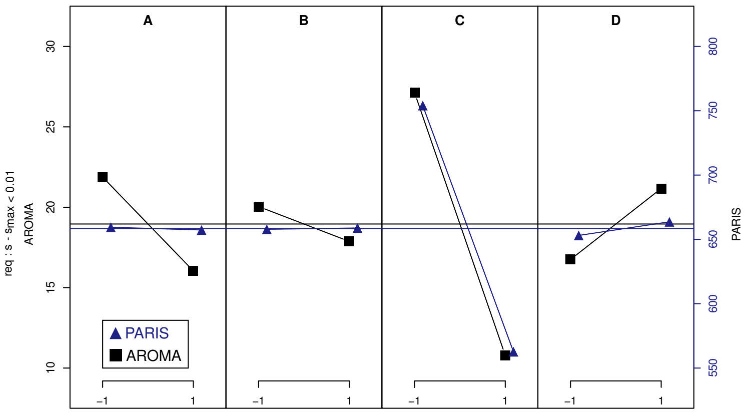


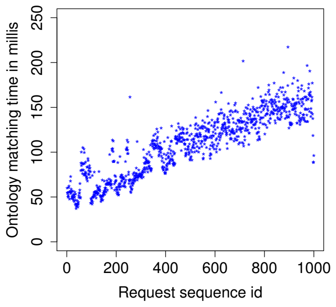
Figure 54 – Main Effects plot for response variable $N_{1\%}$ (first request with strength within 1% of maximum)

statistically significant, are small in the light of practical applications. The largest observed effect of D, in absolute values, is in Figure 54 when $N_{1\%}$ increases from 635 to 664 for PARIS when access is switched from fixed to random. The Pareto graphs also show that the contribution of D to variability pales in face of factor C.

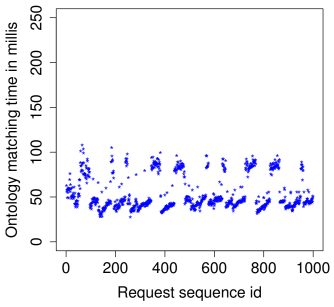
In a practical scenario, reaching the maximum alignment strength may not be necessary, as usually the threshold is fixed for the adopted matcher. The response variable $N_{1\%}$ is closer to this situation. AROMA requires fewer requests to reach high alignment strengths, but it is affected by factor A (shared properties), requiring at least 2 shared properties to generate alignments. PARIS, on the other hand, is not affected by A, but presents a slower evolution of the alignment strength. In summary, the choice between AROMA and PARIS depends on (1) whether the individuals have properties acting as composite keys or as a single key; (2) how many requests are acceptable, for a given application, before alignment strength reaches a threshold.

7.2.3 Ontology Matching Time

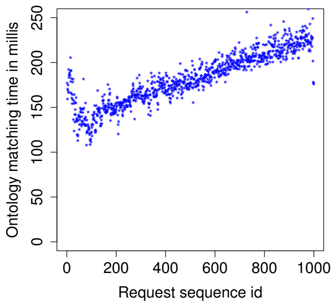
In addition to the experimental design presented in the last section, an analysis regarding the ontology matching was conducted to study Alignator's behavior. As aforementioned, Alignator relies on the existence of intersection between data exposed by different data-driven



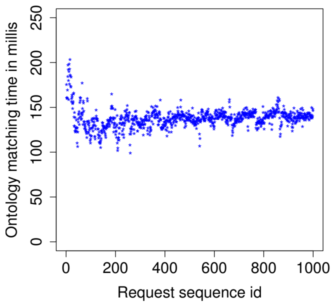
(a)



(b)



(c)



(d)

Figure 55 – (a) alignment time without ENCM - AROMA. (b) alignment time with ENCM=100 - AROMA; (b) alignment time without ENCM - PARIS. (d) alignment time with ENCM=100 - PARIS

microservices. Based on a sample entity, it loads related entities from registered microservices. Hence, the resulting entities are added to each ontology that is aligned by the Ontology Matcher component. The more entities are in an ontology, the higher is the alignment strength. However, accumulating entities into ontologies affects the time to obtain alignments. In order to address this issue, a mechanism to control the number of registered ontologies' entities was developed in the Ontology Manager component. By defining a maximum number of entities, the ontology matching time is reduced while the maximum alignment strength is achieved.

Figure 55 shows the ontology matching time per request for each entity of the dataset. In Figure 53 (a) and Figure 55 (a), one can notice that the maximum alignment strength that states `o0p1` described by the ontology `http://ontology0#` is equivalent to `o1p1` described by the ontology `http://ontology1#` is achieved before the 100th request. However, the entity number control mechanism (ENCM) was not enabled in this case. As a consequence of that, the ontology matching time required to align these properties is significantly increased, as can be seen in (a) and (c) of Figure 55. By setting the ENCM to 100, when a given ontology reaches 100 entities, all entities are discarded. As can be seen in (b) and (d) of Figure 53, the alignment strength drops after every 100 requests. It is important to mention that the highest achieved alignment strength is considered despite this drop caused by ENCM. As at any time, at most 100 entities are considered, the ontology matching time per request is significantly reduced, as shown in Figure 53 (d) and Figure 55 (d).

With respect to the alignment strength attained by both ontology matchers, we can see that AROMA reaches a maximum before PARIS, as shown in Figure 53 (a) and (c). The ontology matchers also required different times for performing alignments. For both configurations, with and without ENCM, PARIS spent significantly more time when compared to AROMA. Furthermore, the ontology matching time analysis shows the importance of finding out a suitable number of entities that are required to attain the maximum alignment strength in order to ensure an adequate performance and alignment strength. One can notice that ENCM set to 100 entities was appropriate to AROMA, as shown in Figure 53 (a), which was able to achieve its maximum alignment strength. On the other hand, as can be seen in Figure 55 (a), this configuration was inappropriate to PARIS, resulting in lower alignment scores.

7.3 EVALUATION OF ALIGNATOR AS A MODULE

This section describes the evaluation of Alignator as an internal module of Linkedator. The goal is to find out how the ontology alignment process affects the link creation process. In order to allow the replication of experimental results, source code and instructions for setting up this experiment are available in a public repository[5].

[5] `https://salvadori.bitbucket.io/projects/phd/experiments/alignatormodule`

7.3.1 Case Study

The case study is based on open data published by the Transparency Portal of the Brazilian Government[6], which provides CSV files regarding to federal contracts, payments, federal workers, social programs and a variety of other public information. Three different datasets have been chosen for this case study, namely, records of a) federal civil servants, b) payments for government purchases and contracts, and c) suppliers of goods or services to the government. A total of 690,930; 8,670 and 50,080 records about civil servants, payments and suppliers, respectively, have been used in this case study, all published in January 2017. All datasets share with one another some intersecting information, such as the full name and document number of a person, as well as the identification number and name of suppliers.

As shown in Figure 56, three microservices have been designed for managing the records of each selected dataset. µService Payments is able to retrieve a payment resolving the URI template `?paymentID`. It is also possible to retrieve a list of all payments through the URI suffix `/list` without informing parameters. Suppliers can be retrieved by resolving the URI template `/?{registerNumber}` and the µService Servants is responsible for providing an interface, through the URI template `?{docID, fullname}`, to retrieve civil servants by their document identification and full name. Figure 56 also illustrates the intersecting information between entities of datasets. On one hand, Civil Servants share their document identification and full name with Payments. On the other hand, Suppliers share their register number and name with Payments, although only register number is used by the microservice as the unique identifier.

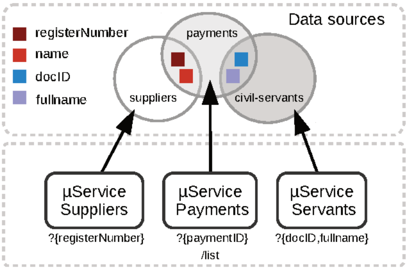


Figure 56 Case study's components

[6] `http://transparencia.gov.br`

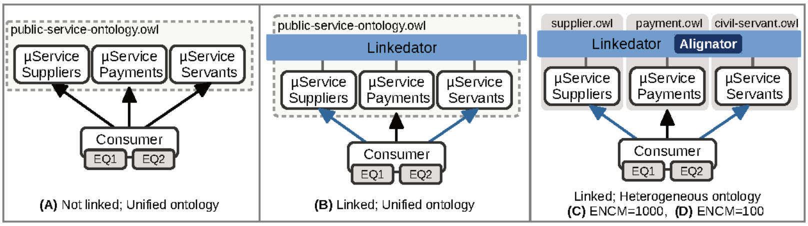

public-service-ontology.owl
µService Suppliers
µService Payments
µService Servants
Consumer
EQ1
EQ2
(A) Not linked; Unified ontology
public-service-ontology.owl
Linkedator
(B) Linked; Unified ontology
supplier.owl
payment.owl
civil-servant.owl
Alignator
Linked; Heterogeneous ontology
(C) ENCM=1000, (D) ENCM=100

Figure 57 Case study scenarios

The case study was built and evaluated in four different scenarios, as depicted in Figure 57:

(A) Not Linked: microservices are deployed without Linkedator.

(B) Linked: all microservices are deployed using the unified ontology *public-service-ontology.owl*. For that reason, Alignator is disabled.

(C) Linked (ENCM=1000): microservices are deployed using their own ontologies. The Alignator framework is enabled with ENCM set to 1000 entities in each ontology.

(D) Linked (ENCM=100): microservices are deployed using their own ontologies. The Alignator framework is enabled with ENCM set to 100 entities in each ontology.

We have also designed a Consumer in order to properly perform the evaluation. The Consumer's implementation relies on two evaluation questions based on the real-world data managed by microservices that compose the case study. The questions aim at finding the federal civil servants that operate more payments (EQ1) as well as suppliers that received the highest amounts of money from the government (EQ2). The Consumer basically retrieves the list with all payments available in the Payments µService and attempts to follow links to servants and suppliers associated with such payment.

In the first scenario, Linkedator is disabled. Therefore, the Consumer must know implementation and deployment details such as URI templates and server IP addresses of each microservice. As shown in Figure 57, the second scenario makes use of *public-service-ontology.owl*, which comprises all microservice data concepts in only one ontology. As Linkedator creates links, the Consumer can follow them and retrieve the suppliers and/or servants associated with a payment. The presence of heterogeneous ontologies is evaluated in the third and fourth scenarios, in which each microservice uses its own particular ontology carrying different concepts. The blue arrows depicted in Figure 57 indicate that HTTP requests are done by the Consumer using the links created by Linkedator. The black arrows represent the requests in which the Consumer previously knows the URIs.

In Figure 58 an example of a civil servant is presented, while Figure 59 presents an example of a supplier. To better understand the desired result of using Linkedator along with Alignator, Figure 60 shows the expected linked Web resource regarding a payment. It is worth mentioning that civil servants and payments share the same value of *servant:registrationNumber* and *payment:executorRegisterNumber* properties, without the information on how to access the related Web resources. Payments also share some property values with suppliers – e.g., *payment:providerRegisterNumber* with *supplier:partnerRegistrationID*, and *payment:providerName*) with *supplier:partnerName*. However, all this information is described by different properties. Additionally, the payment Web resource holds the object property *executedBy*, which can be linked to the corresponding civil servant. Similarly, the supplier Web resource holds the object property *payment:providedBy* that corresponds to the supplier. Thus, it is expected that Alignator would be able to identify the following property equivalences:

1. *payment:executorFullName* = *servant:name*;
2. *payment:executorRegisterNumber* = *servant:registrationNumber*;
3. *payment:providerRegisterNumber* = *supplier:partnerRegistrationID*.

As a result, these alignment triples are informed to Linkedator, which creates links associated with the concept *owl:sameAs* to related Web resources.

```
{
  "@id": "http://civil-servants/docID=1**14\&name=VALMIR M SILVA",
  "@context": { "servant": "http://civil-servant.owl#" },
  "@type": "servant:PublicServant",
  "servant:registrationNumber": "1**14",
  "servant:name": "VALMIR M SILVA",
  "servant:professionalPosition": "Administrator"
}
```

Figure 58 – Example of a civil servant

```
{
  "@id": "http://suppliers/5***8239***156",
  "@context": { "supplier": "http://supplier.owl#" },
  "@type": "supplier:Partner",
  "supplier:partnerRegistrationID": "5***8239***156",
  "supplier:partnerName" : "AUTO POSTO ESTONIA 3 LTDA.",
  "supplier:partnerTributationType" : "Comercio varejista..."
}
```

Figure 59 – Example of a supplier

```
{
  "@context": { "payment": "http://payment.owl#" },
  "@type": "payment:Expense",
  "@id": "http://government-payments/1",
  "payment:expenseValue": 743.73,
  "payment:expenseDate" : 05-12-2016,
  "payment:providedBy": {
    "payment:providerRegisterNumber": "5***8239***156",
    "payment:providerName": "AUTO POSTO ESTONIA 3 LTDA.",
    "@type": "payment:Provider",
    "owl:sameAs": ["http://suppliers/5***8239***156"]
  }
  "payment:executedBy": {
    "payment:executorRegisterNumber": "1**14",
    "payment:executorFullName": "VALMIR M SILVA",
    "@type": "payment:Executor",
    "owl:sameAs": ["http://civil-servants/docID=1**14\&name=VALMIR M SILVA" ]
   }
}
```

Figure 60 – Example of a supplier - linked with related Web resources

All components included in the scenarios have been deployed into the Amazon EC2 (Elastic Compute Cloud) environment. Five dedicated virtual machines have been created: three of them for the microservices of the case study, one for the Linkedator-API (which also runs Alignator as an internal module) and another one for the Consumer. The selected instance configuration was m3.medium, with Intel Xeon E5-2670 v2 (Ivy Bridge) processors running at 2.6 GHz, 3.75 GiB of memory, with Ubuntu Server 14.04 LTS (HVM) and Oracle Java Development Kit 8.

7.3.2 Experimental Results

The consumer application was executed in all four previously described scenarios, and the results are shown in Table 8. This table presents the mean response time/request spent for creating links in Linkedator, for aligning ontologies in Alignator and the time perceived by the Consumer. One can notice that the time perceived by the Consumer is increased in nearly 10 times due to the link creating process, when comparing scenarios (A) with (B). However, the impact of the ontology alignment is even higher, increasing response time in nearly 100 times, as can be seen in (C). On the other hand, as shown in (D), when the entity number control mechanism is properly used, it was possible to reduce the alignment time in 64%.

Table 8 also presents the total number of requests executed by the consumer to obtain all entities to answer the evaluation questions. It is worthwhile to note that the number of requests executed by the Consumer is different in each scenario. In scenario (A), for each record retrieved by the µService Payments, the consumer executed other two requests to µService Suppliers and to µService Servants. Nevertheless, there are payments that do not have correspondence to registered civil servants or suppliers, resulting in invalid links. However, when Linkedator is used, it validates such links and then only valid links are executed. In scenarios (C) and (D), it is necessary to perform alignments to properly create links among heterogeneous ontologies. This process requires a certain number of entities before producing alignments with a specific strength, which causes some link misses.

Table 8 – Experiment scenarios' results - Mean times per request (time in milliseconds)

Scenario	**Consumer**	**Linkedator**	**Alignator**	**Requests**
(A)	4.10	-	-	26,010
(B)	39.83	26.49	-	22,318
(C)	386.06	45.35	363.47	22,297
(D)	139.44	37.95	99.33	22,310

Figure 61 shows how the use of the proposed composition method is perceived by the client application with respect to response time distribution. Figure 61 (a) presents the time distribution considering the first scenario (not linked Web resources). Most requests in this scenario are executed between 3 ms and 5 ms. Figure 61 (b) presents

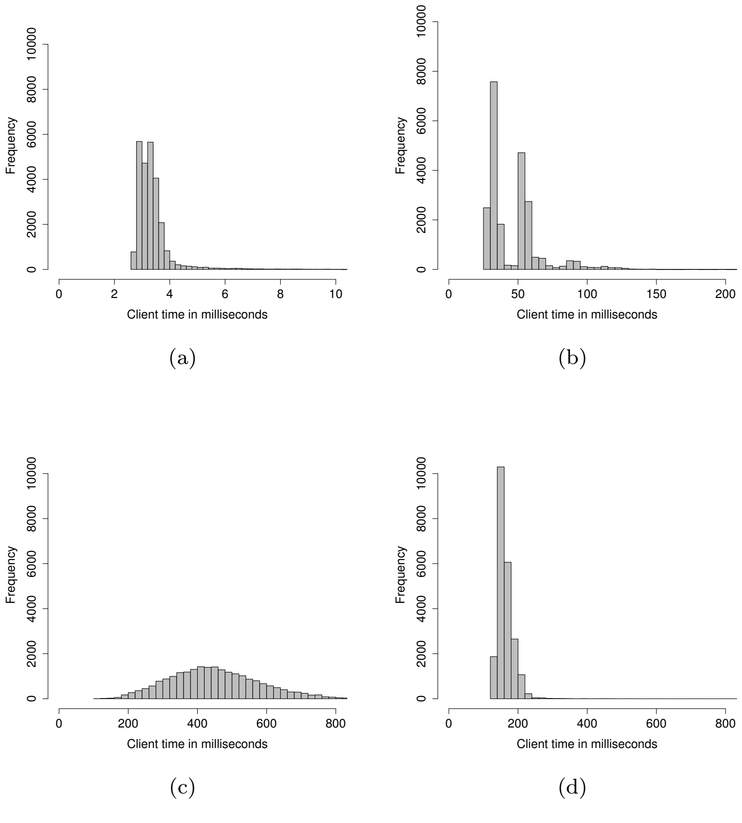


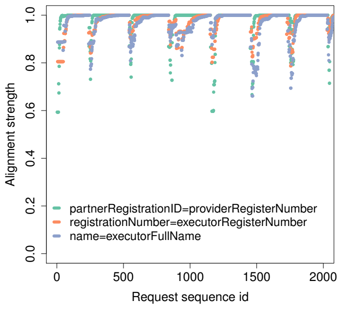
Figure 61 – Client response time distribution (a) Not linked; (b) Linked entities using unified ontology; (c) Linked entities using heterogeneous ontology and entity number control set to 1000 entities per ontology; (d) Linked entities using heterogeneous ontology and entity number control set to 100 entities per ontology

the time distribution considering the second scenario (linked Web resources using unified ontology). In this scenario, most requests are executed within two ranges, the first one between 30 ms and 40 ms, and the second one between 50 ms and 60 ms. Figure 61 (c) presents the distribution of requests executed in the third scenario (linked Web resources using heterogeneous ontology and ENCM set to 1000 entities per ontology). One can notice that the link ontology matching process distributes requests more evenly between 200 ms and 800 ms, reducing the spikes observed in previous scenarios. However, the use of a more suitable value for ENCM, presented in Figure (d), concentrates the execution of most requests between 140 ms and 200 ms.

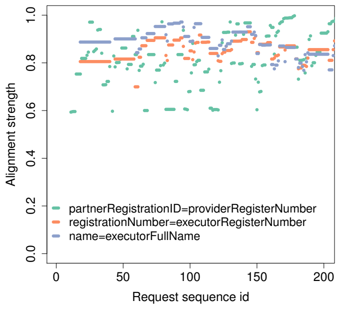
Figure 62 shows the results regarding two distinct entity number control mechanisms. Graphs (a), (c) and (e) present the alignment strength, number of entities in each ontology and the ontology matching time, respectively, achieved in the first 2000 requests performed by the consumer for scenario (C), while (b), (d) and (f) present the same results considering the first 200 requests for scenario (D). It is important to notice that in both scenarios there were several flushes throughout the execution. In (a), where the ENCM is set to 1000 entities per ontology, flushes are performed around each 250 requests, while in (b), where ENCM is set to 100 entities per ontology, flushes are performed quite more often. Furthermore, the alignment strength achieved the maximum value before all sequence of flushes in scenario (a). On the other hand, such alignment strength was not achieved before all flushes in (b). However, Alignator was developed to consider the highest alignment strength achieved in its runtime. Once the maximum value is achieved, lower strength values are not considered.

There is a direct relation between the number of entities considered in the ontology matching process and the time required to produce such alignments. In Figure 62 (e), there were three spikes within the range between the 1st and the 1,000th requests, that correspond to the exact moment in which flushes were performed, as can be seen in Figure62 (c). In contrast, considering scenario (D), in which flushes were performed around each 20 requests, as shown in Figure 62 (d), the time required to produce alignments was significantly reduced, as can be seen in Figure 62 (f).

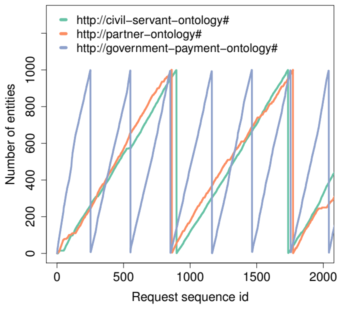
Figure 63 (a) shows an analysis of variance regarding the time spent by Linkedator to create links for scenarios (A), (B) and (C). It is possible to observe the impact of ontology alignments in the link creation process. Considering that scenario (B) uses an unified ontology, there is no need for ontology alignment, so there is no effect in the link



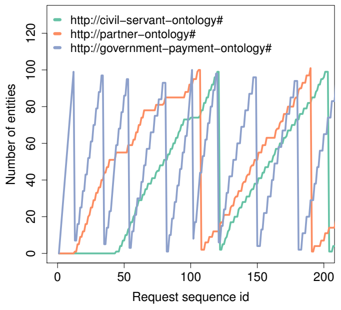
(a)



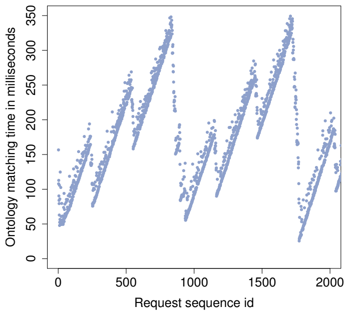
(b)



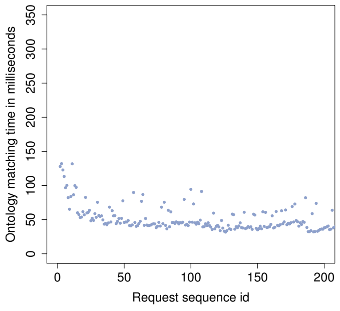
(c)



(d)



(e)



(f)

Figure 62 – Entity number control mechanism analysis; (a) alignment strength, (c) number of entities in ontologies, (e) matching time for ENCM=1000; (b) alignment strength, (d) number of entities in ontologies, (f) matching time for ENCM=100

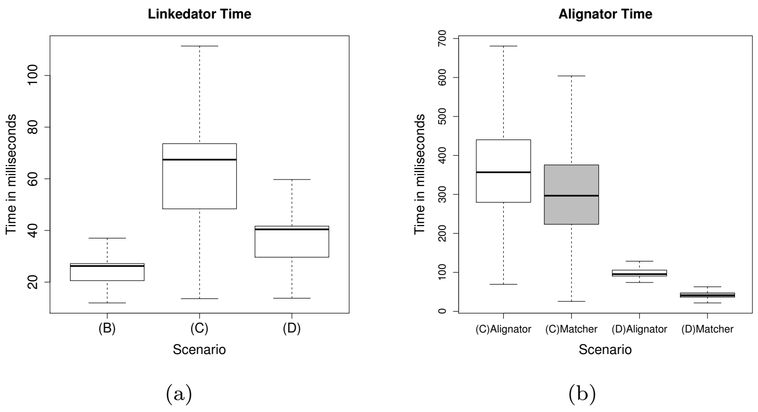


Figure 63 – Time variance analysis; (a) Linkedator; (b) Alignator

creation process. However, scenarios (C) and (D) require such alignments. One can notice that the required time for creating links in those scenarios had a slight increase. This is due to the fact that equivalent properties and classes are being considered in heterogeneous ontology scenarios. Furthermore, scenario (C) has a significant time variance. This is due to the interaction between Linkedator and Alignator, in which Alignator directly writes alignments into Linkedator's ontology model. Such write operation is affected by the number of entities of managed ontologies that Alignator has to deal with.

igure 63 (b) shows the time required to perform ontology alignments for scenarios (C) and (D). The time spent by the ontology matcher and by Alignator, which encompasses the first one, are shown separately. The first important thing to notice is that the ontology matcher corresponds to 83% of the time spent by Alignator in scenario (C) and 43% in scenario (D). In addition, the time spent by Alignator in scenario (D) was dramatically reduced when compared with scenario (C). This shows the importance of setting up a suitable ENCM to avoid such waste of time, since both scenarios have achieved the same alignment strength value. In addition, scenario (C) has higher time variance due to the larger number of entities, which grows in a non-continuous way up to a larger maximum of 3000 entities. This non-continuous growth impacts both the variance of alignment time as well as the time required to feed Linkedator with the updated alignments, as can be seen in Figure 62 (e) and (f).

7.4 L2R EVALUATION

This section presents evaluation results regarding the literal to resource conversion performed by L2R. This evaluation uses the same datasets previously described in section 7.3.1, which contain information about federal civil servants, payments for the acquisition and contracting of public works as well as government purchases, and suppliers that provide goods or services. These three datasets were materialized into RDF triples according to Figures 58, 59 and 60 and indexed by L2R for automatic creation of background information, since no external data was used to perform the conversion. Once indexed, a new materialization process was performed taking into account the resulting background information.

As shows Figure 64, a initial materialization resulted in 5.43 million of triples. However, only 17,340 triples were used to represent links between resources. After the conversion, a total of 1.6 million triples were used to allocate the new Web resources (Background) created by the conversion process. In this process, all literal values were replaced by the URIs of these new Web resources. As a result, the data overlapping between Web resources was drastically reduced. Instead, Web resources now share a common set of URIs that point to a background information.

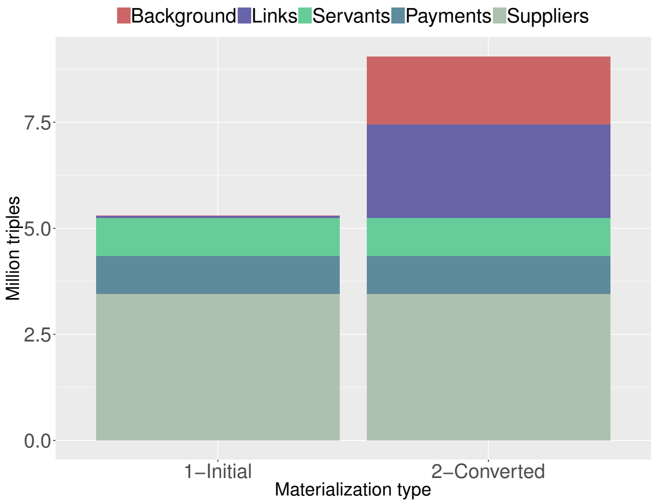


Figure 64 – L2R conversion results

Despite the increased number of triples required to represent the converted dataset, the dataset after conversion resulted in a much more connected graph. Originally, the respective 8,670 payments were connected with 3,992 civil servants and with 2014 suppliers. However, after the conversion, 804,244 new Web resources were created. As shown by Figure 65, besides the aforementioned links, payments are now connected with 6,201 Web resources. Similarly, servants and suppliers are connected with 673,012 and 137,781 background information resources respectively. Additionally, the intersection data between all the three types of Web resources are represented by 103 Web resources. It is important to mention that a background information resource contains backtrack links, which implies that it holds triples in which the URL is the subject of another Web resource. As a result, the converted dataset allows a given consumer to navigate among Web resources to find further related information.

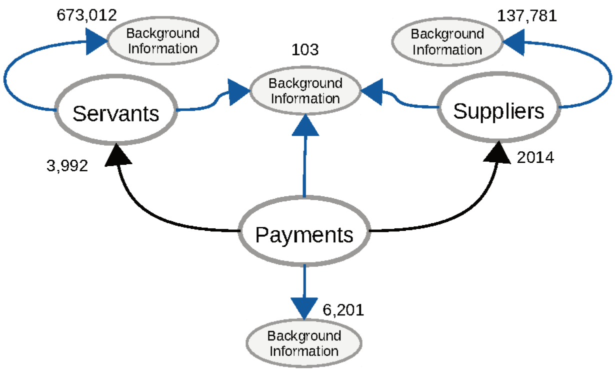


Figure 65   Converted graph

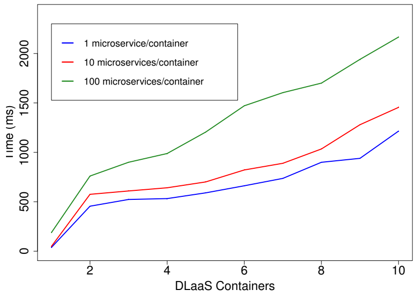
The conversion of literal values into independent and accessible Web resources represents an important initiative for data reuse. However, it is meant to be used after more sophisticated strategies, such as resource structure organization and explicit semantic definitions. It means that literal to resource conversion is better used as the last opportunity to connect resources, otherwise, it would result in less effective reuse of data.

7.5 AGENTS EVALUATION

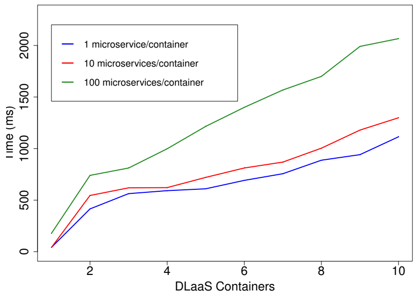
This section presents the evaluation of DLaaS agents. The objective of this evaluation is to find out the resulting delay in adopting multi-agent systems as a communication mechanism. This evaluation uses the same datasets previously described in section 7.3.1. DLaaS agents were evaluated according to three different scenarios: (i) a single sdd-µs instance running in a container; (ii) 10 sdd-µs instances running in a container and; (iii) 100 sdd-µs instances running in a container. These scenarios were executed according to 10 different configurations, each one of them deploying their respective microservices in a new DLaaS container, from 1 to 10 DLaaS containers. Each container was executed in an independent virtual machine. A JADE agent platform (version 4.5.0) is instantiated along with each DLaaS container and the dataset's entities were equally distributed among the microservices. This experiment aimed to measure the communication time between agents in each configuration. Linkedator and L2R processing times were not considered in this evaluation.

This experiment was executed using the computational infrastructure provided by the C3 Lab, with the following configuration: a dedicated server with 24 Intel Xeon X5690 processors at 3.47GHz frequency and 148 GiB of primary memory, running the CentOS-7 operating system and openjdk version 1.8.0_141 with maximum heap size set to 128 KB for each microservice and 2 GiB for each Linkedator and L2R instances.

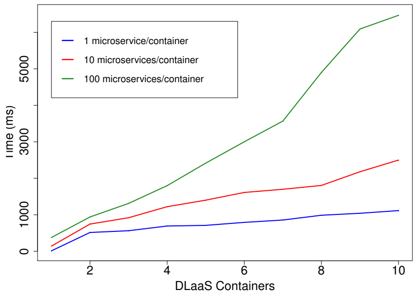
Figure 66 shows the communication times for Linkedator, L2R and sdd-µs agents. In Figure 66 (a) and (b), Linkedator and L2R agents are evaluated when performing the explicit semantic definition across multiple DLaaS containers and when performing the literal to resource conversion across multiple DLaaS containers, respectively. Initially, the three scenarios were executed in a single DLaaS container. In this configuration there is no inter-platform communication, however, a small delay can be observed when the JADE platform is enabled. Due to the need of activating FIPA MTP protocols for configurations in which more than one DLaaS container is instantiated, a delay ranging from 415 ms (single microservice per container scenario) to 740 ms (100 microservices per container scenario) can be observed. Once initiated, the time delay caused by FIPA MTP protocols is significantly reduced. The more DLaaS containers, the higher is the communication delay. However, it was required 2166.3 ms to perform all the communication for creating links based on the explicit semantic definition and 2013.8



(a)



(b)



(c)

Figure 66 – DLaaS agents communication time: (a) Linkedator Agent, (b) L2R Agent, (c) sdd-μs Agent

ms for converting literal values into resources across 10 DLaaS platforms, each one running 100 sdd-µs instances. The more microservices running in a container, the bigger the Hydra documentation and the background information. For this reason, more time is required by agents to communicate with each other and decide the next steps for scenarios (ii) and (iii).

Figure 66 (c) shows the communication times required by agents to spread the changes about their sdd-µs domain ontologies over multiple DLaaS containers. When a domain ontology of a given sdd-µs is updated, the attached agent establishes communications with agents that represent all the other running microservices. For this reason, the number of sdd-µs instances represents the most important factor for agent communication. With regard to scenario (i), it was required 1,114.5 ms to communicate to all other agents that a given ontology was updated, and 6,466.3 ms to spread this message for 1,000 microservices across 10 containers.

7.6 FINAL CONSIDERATIONS

This chapter presented the evaluation of the proposed interservice linking strategies and their implementation solutions. The evaluations were conducted according to appropriate statistical methods and used real-world datasets.

Experimental results showed that the adoption of Linkedator does not influence the overall client response time, considering the case study scenario. However, factors such as link validation, the number of potential links and the use of cache have a significant influence on the link creation process. The link validation process has an important influence on the response time of Linkedator-Core. However, in cases where microservices and the Linkedator-API are deployed in the same infrastructure, it reduces client time, due to link validation within the infrastructure being less expensive than providing occasional invalid links, forcing clients to waste time.

Additionally, it was analyzed the impact of performing ontology alignments using Alignator. Different scenarios that explore some characteristics, such as the adoption of a unified ontology or heterogeneous ontologies, were analyzed in order to identify their impact on the link creation process. Scenarios that explore specific characteristics of the adoption of heterogeneous ontologies were also considered. For those scenarios, the evaluation showed the importance of properly defining a threshold to be used by the entity number control mechanism, otherwise, the ontology matcher can become very time-consuming or result in alignment strengths that do not achieve the full potential of Alignator.

The evaluation of the Literal to Resource (L2R) component showed that the resulting materialization becomes significantly larger than the original dataset. However, after conversion of literal values into resources, a way more connected graph is obtained, following the principles of Linked Data. As a consequence, consumers are allowed to navigate among Web resources to find further related information.

Finally, the evaluation presented results about the adoption of multi-agent systems. It was evaluated in a large scale scenario deployed in a cloud computational infrastructure. Experimental results showed that agents are a suitable alternative communication solution for microservices that need exchange messages with one another. By adopting multi-agent systems, microservices delegate the communication to agents. As a result, microservices implementations can be exclusively focused on implementing their functionalities.

8 CONCLUSIONS

We are living in the age of big data, advanced analytics, and data science. Companies, governments, and even ordinary people are producing and publishing a huge amount of data on the Web. However, there is a lack of solutions for data publishing that follow the best practices for exposing, sharing and connecting data. With this regard, this work proposed DLaaS, a microservices infrastructure for publishing linked data. DLaaS is capable of interconnecting Web resources from multiple data sources. The proposed infrastructure is composed of several internal components responsible for performing a multitude of tasks for pro-actively connecting Web resources managed by the infrastructure.

8.1 CONTRIBUTIONS

The contributions resulting from this thesis are:

1. **A capacity model for semantic data providers.** It is broadly accepted that the adoption of Semantic Web techniques is a key factor for allowing automatic discovery, selection, and composition of Web Services. However, it is not clear how to apply such semantic features for implementing data providers. This work proposed a capacity model that describes how to properly adopt the most appropriate semantic Web features for implementing data-driven services.

2. **Conversion of legacy data into linked data.** This work presented sdd-µs, a specialized service capable of converting non-semantic data into linked data. It converts simple data entries into semantic Web resources that can be linked to other resources provided by different data sources.

3. **Data optimization and interlinking.** A novelty of this work refers to the capability of pro-actively interconnecting the data. Three different strategies for interlinking Web resources were proposed. The first strategy aims to change the Web resource structure to maximize data reuse. The second strategy aims at converting literal values into Web resources. Finally, the third strategy aims at creating links based on explicit semantic definitions.

By adopting the proposed data linking strategies, data-driven services implementations naturally follow principles such as universality and decentralization defined by Berners-Lee (2010). Such principles are concerned with avoiding the creation of data silos, then collaborating for making the Web a global linked data space.

4. **Data linking infrastructure.** This work proposed DLaaS, a microservices infrastructure for publishing linked data. It aims at facilitating the execution of necessary processes to properly publish high quality linked data, which includes semantic enrichment, data linking, and publication. In summary, DLaaS put together all the necessary tools for performing the conversion of legacy data into linked data as well as for performing data optimization and interlinking.

Taking into account the state of the art, the contributions of this Ph.D. thesis rely mainly on the proactive aspect of connecting the data spread over multiple data providers in a microservice architecture. The adoption of data mining techniques and data pattern recognition for connecting data is not unprecedented. However, the adopted strategy of using pattern recognition algorithms for optimizing the Web resource structure is innovative. The contributions regarding data integration followed patterns established by the Semantic Web community, which validates the hypothesis of this thesis that states that Semantic Web and Linked Data offer all the necessary principles for data services integration.

8.2 LIMITATIONS

Along with this thesis, we have highlighted some limitations regarding data optimization, the semantic capability achieved by DLaaS and the adoption of multi-agent systems. Below, they are discussed in further details.

Resource Structure Optimization. The current implementation adopts the A-Close algorithm for mining association rules. In addition, a variety of other algorithms with a similar purpose are available and may result in better results depending on characteristics of the dataset to be optimized. Additionally, adjustments in the algorithm parameters may yield better results in certain cases.

Semantic Capacities. DLaaS does not support any level of semantic capacities regarding authoring and information fusion dimensions. Instead, it is focused on Web resource re-design, knowledge and representational management, de-referenciability and data model. However, such features can be implemented in order to make DLaaS even more effective. For example, information fusion techniques could be applied to decide when a given link should be promoted to the resource referenced by it. As a result, it would benefit data consumers, since more information can be retrieved in a single request.

Multi-Agent Systems. DLaaS adopts multi-agents systems to tackle dynamic features regarding changes in the infrastructure. These features are addressed by spreading messages across multiples distributed platforms. These messages are sent and received by agents able to perform specialized tasks. However, agents communicate with one another exclusively based on informative and request messages. The capability in which agents are able to negotiate based on their own knowledge is not considered in this work. By allowing agents to negotiate, semantic inconsistency could be detected and avoided, and better semantic alignments and resource structure optimization could be produced, consequently resulting in higher levels of data reuse.

8.3 OPEN QUESTIONS AND FUTURE WORKS

Assuming that data can also be modeled as a monolith, microservices may be adopted as a suitable solution for distributing data across several small service providers. However, there are microservices characteristics that may conflict with some linked data principles. Disposability and native cloud-oriented are two important characteristics associated with microservices. Disposability means that a given microservice can be removed from the infrastructure if it is no longer useful. Native cloud-oriented applications refer to software designed to be deployed in cloud computing infrastructure. It often implies in elasticity, which makes use of transient computing infrastructure. However, linked data assumes that URLs that connect Web resources are permanent. That said, the adoption of microservices for publishing linked data requires special attention to these details, otherwise, the quality of linked data publishing may be drastically affected.

For future works, the Capacity Model for Semantic Data Providers may be used for proposing novel solutions that handle desirable semantic features. Authoring, licensing, versioning and provenance are good examples for the linked data scenario. These features represent aspects that are out of the scope of this work and could be properly addressed in future works. Additionally, these semantic features rise further discussions about Web resources immutability. Future works could consider handling immutability and all the issues associated with it.

8.4 PUBLICATIONS

This work has resulted in the following publications.

1. Oliveira, B. C. N., Salvadori, I., Huf, A., & Siqueira, F. (2016). A platform to enrich, expand and publish linked data of police reports. In 15th International Conference WWW/Internet. Mannheim, Germany: IADIS Press.

2. Salvadori, I., Huf, A., & Siqueira, F. (2016). An Agent-based Composition Model for Semantic Microservices. In 15th International Conference WWW/Internet. Mannheim, Germany: IADIS Press.

3. Salvadori, I., Huf, A., Mello, R. dos S., & Siqueira, F. (2016). Publishing linked data through semantic microservices composition. In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services - iiWAS '16 (pp. 443–452). New York, New York, USA: ACM Press.

4. Salvadori, I., Oliveira, B. C. N., Huf, A., Inacio, E. C., & Siqueira, F. (2017). An Ontology Alignment Framework for Data-driven Microservices. In Proceedings of the 19th International Conference on Information Integration and Web-based Applications and Services - iiWAS '17.

5. Salvadori, I. L., Huf, A., Oliveira, B. C. N., dos Santos Mello, R., & Siqueira, F. (2017). Improving entity linking with ontology alignment for semantic microservices composition. International Journal of Web Information Systems, 13(3), 302–323.

6. Salvadori, I. L., Huf & Siqueira, F. (2019). Semantic Data-Driven Microservices. IEEE Computer Society Signature Conference on Computers, Software and Applications. COMPSAC ´19.

Additionally, the following articles were published in collaboration with other authors.

1. de Camargo, A., Salvadori, I., Mello, R. dos S., & Siqueira, F. (2016). An architecture to automate performance tests on microservices. In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services - iiWAS '16 (pp. 422–429). New York, New York, USA: ACM Press.

2. Oliveira, B. C. N., Huf, A., Salvadori, I., & Siqueira, F. (2017). Automatic Semantic Enrichment of Data Services. In Proceedings of the 19th International Conference on Information Integration and Web-based Applications and Services - iiWAS '17.

3. Weingartner, R., Martins, P. H. P., Salvadori, I. L., Westphall, C. M., & Siqueira, F. (2017). Improving OpenID Connect Federation's Interoperability with Web Semantics. In 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA) (pp. 1269–1276). IEEE.

4. Oliveira, B. C. N., Huf, A., Salvadori, I. L., & Siqueira, F. (2018). OntoGenesis: an architecture for automatic semantic enhancement of data services. International Journal of Web Information Systems, IJWIS-04-2018-0020.

REFERENCES

AALST, W. M. van der; WESKE, M.; GRÜNBAUER, D. Case handling: a new paradigm for business process support. *Data And Knowledge Engineering*, v. 53, n. 2, p. 129–162, may 2005. ISSN 0169023X.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. (VLDB '94), p. 487–499. ISBN 1-55860-153-8.

AHMED, R.; BOUTABA, R. A Survey of Distributed Search Techniques in Large Scale Distributed Systems. *IEEE Communications Surveys & Tutorials*, v. 13, n. 2, p. 150–167, 2011. ISSN 1553-877X. <http://ieeexplore.ieee.org/document/5473882/>.

AL-SHARGABI, B.; SHEIKH, A.; SABRI, A. Web service composition survey: State of the art review. *Recent Patents on Computer Science*, v. 3, n. 2, p. 91–107, 2010.

ALARCON, R.; WILDE, E. Linking data from RESTful services. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2010. v. 628. ISSN 16130073.

ALONSO, G. et al. *The Semantic Web*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. 123–149 p. ISBN 978-3-540-76451-9. <http://link.springer.com/10.1007/978-3-662-10876-5_5 http://link.springer.com/10.1007/978-3-540-76452-6>.

ANGELE, J. OntoBroker: Mature and approved semantic middleware. *Semantic Web*, v. 5, n. 3, p. 221–235, 2014. ISSN 22104968.

ASSAF, A.; SENART, A. Data Quality Principles in the Semantic Web. In: *2012 IEEE Sixth International Conference on Semantic Computing*. IEEE, 2012. p. 226–229. ISBN 978-1-4673-4433-3. <http://ieeexplore.ieee.org/document/6337108/>.

BAGOZI, A. et al. Interactive Data Exploration as a Service for the Smart Factory. *2017 IEEE International Conference on Web Services (ICWS)*, p. 293–300, 2017. <http://ieeexplore.ieee.org/document/8029774/>.

BANG, A. et al. The Role of Hypothesis in Constructive Design Research. In: *The Art of Research 2012: Making, Reflecting and understanding*. Helsinki - Finland: [s.n.], 2012. p. 1–11.

BANOUAR, O.; RAGHAY, S. Comparative study of the systems of semantic integration of information: A survey. In: *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, 2015. v. 2016-July, p. 1–8. ISBN 978-1-5090-0478-2. ISSN 21615330. <http://ieeexplore.ieee.org/document/7507235/>.

BARHAMGI, M.; BENSLIMANE, D. Composing Data-Providing Web Services. *VLDB PhD Workshop*, p. 0–5, 2009.

BARTALOS, P.; BIELIKOVA, M. Automatic dynamic web service composition: A survey and problem formalization. *Computing and Informatics*, v. 30, n. 4, p. 793–827, 2011.

BATINI, C.; SCANNAPIECO, M. *Data and Information Quality*. Cham: Springer International Publishing, 2016. 87–112 p. (Data-Centric Systems and Applications). ISBN 978-3-319-24104-3.

BATTLE, R.; BENSON, E. Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). *Web Semantics*, v. 6, n. 1, p. 61–69, fev. 2008. ISSN 15708268.

BEEK, W. et al. LOD Lab: Scalable Linked Data Processing. In: PAN, J. Z. et al. (Ed.). Cham: Springer International Publishing, 2017, (Lecture Notes in Computer Science, v. 9885). p. 124–155. ISBN 978-3-319-49492-0.

BELLIFEMINE, F.; CAIRE, G.; GREENWOOD, D. *Developing Multi-Agent with JADE Systems*. [S.l.: s.n.], 2007. 286 p. ISSN 00380644. ISBN 9780470057476.

BERGAMASCHI, S. et al. From Data Integration to Big Data Integration. In: FLESCA, S. et al. (Ed.). Cham: Springer International Publishing, 2018, (Studies in Big Data, v. 31). p. 43–59. ISBN 978-3-319-61892-0.

BERNERS-LEE, T. Long Live The Web. *Scientific American*, v. 303, n. 6, p. 80–85, dec 2010. ISSN 0036-8733.

BERNERS-LEE, T. *Linked Data – Design Issues*. 2011. <https://bit.ly/21MR3Zt>.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data-the story so far. In: _____. [S.l.]: IGI Global, 2009. p. 205–227.

BORST, W. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse.* Tese (Doutorado) — University of Twente, Netherlands, 9 1997.

BRANDT, E.; BINDER, T. Experimental Design Research : Genealogy – Intervention - Argument. In: *International association of societies of design research 2007.* [S.l.: s.n.], 2007. p. 17.

CAMPOS, G. M. M.; ROSA, N. S.; PIRES, L. F. A Survey of Formalization Approaches to Service Composition. In: *Services Computing (SCC), 2014 IEEE International Conference on.* [S.l.: s.n.], 2014. p. 179–186. ISBN VO -.

CAREY, M. J.; ONOSE, N.; PETROPOULOS, M. Data services. *Communications of the ACM*, v. 55, n. 6, jun 2012. ISSN 00010782.

CASTELFRANCHI, C. Guarantees for autonomy in cognitive agent architecture. In: *Proceedings of the Workshop on Agent Theories, Architectures, and Languages on Intelligent Agents.* Berlin, Heidelberg: Springer-Verlag, 1995. (ECAI-94), p. 56–70. ISBN 3-540-58855-8. <http://dl.acm.org/citation.cfm?id=201157.201178>.

CHEATHAM, M.; PESQUITA, C. Semantic Data Integration. In: *Handbook of Big Data Technologies.* Cham: Springer International Publishing, 2017. p. 263–305.

DAVID, J. Association Rule Ontology Matching Approach. *Int. Journal on Semantic Web and Information Systems*, IGI Global, v. 3, n. 2, p. 27–49, 2007. ISSN 1552-6283.

DOAN, A.; HALEVY, A.; IVES, Z. *Principles of data integration.* [S.l.]: Elsevier, 2012. ISBN 9780124160446.

DONG, H.; HUSSAIN, F. K.; CHANG, E. Semantic Web Service matchmakers: state of the art and challenges. *Concurrency and Computation: Practice and Experience*, v. 25, n. 7, p. 961–988, may 2013. ISSN 15320626.

DUAN, Q.; YAN, Y.; VASILAKOS, A. A survey on service-oriented network virtualization toward convergence of networking and cloud computing. *IEEE Transactions on Network and Service Management*, v. 9, n. 4, p. 373–392, 2012.

ELMAGARMID, A. K.; IPEIROTIS, P. G.; VERYKIOS, V. S. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, v. 19, n. 1, p. 1–16, jan. 2007. ISSN 1041-4347.

ELSAYED, D.; SALAH, A. Semantic Web Service discovery: A systematic survey. In: *2015 11th International Computer Engineering Conference: Today Information Society What's Next?, ICENCO 2015*. [S.l.: s.n.], 2015. p. 131–136.

EUZENAT, J.; SHVAIKO, P. *Ontology Matching*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN 3540496114.

FEITOSA, D. et al. A systematic review on the use of best practices for publishing linked data. *Online Information Review*, v. 42, n. 1, p. 107–123, feb 2018. ISSN 1468-4527. <http://www.emeraldinsight.com/doi/10.1108/OIR-11-2016-0322>.

FELLAH, A.; MALKI, M.; ELçI, A. Web services matchmaking based on a partial ontology alignment. *Int. Journal of Information Technology and Computer Science (IJITCS)*, MECS Publisher, v. 8, n. 6, p. 9–20, jun. 2016. ISSN 2074-9007.

FERRARA, A.; NIKOLOV, A.; SCHARFFE, F. Data Linking for the Semantic Web. *International Journal on Semantic Web and Information Systems*, v. 7, n. 3, p. 46–76, jan. 2011. ISSN 1552-6283.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. 162 p. Tese (Doutorado) — University of California, Irvine, 2000. <http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm>.

FOURNIER-VIGER, P. et al. The SPMF Open-Source Data Mining Library Version 2. In: *Machine Learning and Knowledge Discovery in Databases*. [S.l.]: Springer International Publishing, 2016. p. 36–40. ISBN 978-3-319-46131-1.

GAO, L.; URBAN, S.; RAMACHANDRAN, J. A survey of transactional issues for Web Service composition and recovery. *International Journal of Web and Grid Services*, v. 7, n. 4, p. 331–356, 2011.

GARRIGA, M. c. et al. RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications*, v. 60, p. 32–53, 2016.

GARTNER, R. *Metadata*. Cham: Springer International Publishing, 2016. ISBN 978-3-319-40891-0. <http://link.springer.com/10.1007/978-3-319-40893-4>.

GENESERETH, M. R.; KETCHPEL, S. P. Software agents. *Commun. ACM*, ACM, New York, NY, USA, v. 37, n. 7, p. 48–ff., jul. 1994. ISSN 0001-0782. <http://doi.acm.org/10.1145/176789.176794>.

GIROLAMI, M. b.; CHESSA, S. b.; CARUSO, A. On service discovery in mobile social networks: Survey and perspectives. *Computer Networks*, v. 88, p. 51–71, 2015.

GRIFFITHS, N.; CHAO, K.-M. *Agent-Based Service-Oriented Computing*. London: Springer London, 2010. ISBN 978-1-84996-040-3. <http://link.springer.com/10.1007/978-1-84996-041-0>.

GROLINGER, K. et al. Integration of business process modeling and Web services: A survey. *Service Oriented Computing and Applications*, v. 8, n. 2, p. 105–128, 2014.

GRUBER, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition*, v. 5, n. 2, p. 199 – 220, 1993. ISSN 1042-8143. <http://www.sciencedirect.com/science/article/pii/S1042814383710083>.

GUPTA, A. *Data Provenance*. Boston, MA: Springer US, 2009. 608–608 p.

GUPTA, S. et al. Karma: A System for Mapping Structured Sources into the Semantic Web. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.: s.n.], 2015. v. 7540, p. 430–434. ISBN 9783662466407.

HANG, F.; ZHAO, L. Supporting End-User Service Composition: A Systematic Review of Current Activities and Tools. In: *Proceedings - 2015 IEEE International Conference on Web Services, ICWS 2015*. [S.l.: s.n.], 2015. p. 479–486.

HOGAN, A. et al. An empirical survey of Linked Data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, v. 14, p. 14–44, jul 2012. ISSN 15708268. <http://dx.doi.org/10.1016/j.websem.2012.02.001 http://linkinghub.elsevier.com/retrieve/pii/S1570826812000352>.

IMMONEN, A.; PAKKALA, D. A survey of methods and approaches for reliable dynamic service compositions. *Service Oriented Computing and Applications*, v. 8, n. 2, p. 129–158, jun 2014. ISSN 1863-2386. <http://link.springer.com/10.1007/s11761-013-0153-3>.

ISLAM, N.; ABBASI, A. Z.; SHAIKH, Z. a. Semantic web: Choosing the right methodologies, tools and standards. In: *2010 International Conference on Information and Emerging Technologies, ICIET 2010.* [S.l.]: IEEE, 2010. p. 1–5. ISBN 9781424480012.

ISSARNY, V. et al. Service-oriented middleware for the Future Internet: State of the art and research directions. *Journal of Internet Services and Applications*, v. 2, n. 1, p. 23–45, 2011.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* [S.l.]: Wiley, 1991. 720 p. ISBN 978-0-471-50336-1.

JU, Y. Leveraging levels of information services and developing knowledge services. *Library Management*, v. 27, n. 6/7, p. 354–361, jul 2006. ISSN 0143-5124. <http://www.emeraldinsight.com/doi/10.1108/01435120610702341>.

JULA, A.; SUNDARARAJAN, E.; OTHMAN, Z. Cloud computing service composition: A systematic literature review. *Expert Systems with Applications*, v. 41, n. 8, p. 3809–3824, jun 2014. ISSN 09574174. <http://linkinghub.elsevier.com/retrieve/pii/S0957417413009925>.

KAPURUGE, M.; HAN, J.; COLMAN, A. Support for business process flexibility in service compositions: An evaluative survey. In: *Proceedings of the Australian Software Engineering Conference, ASWEC.* [S.l.: s.n.], 2010. p. 97–106.

KASANEN, E.; LUKKA, K.; SIITONEN, A. The Constructive Approach in Management Accounting Research. *Journal of Management Accounting Researc*, v. 5, n. June 1991, p. 243–264, 1993. ISSN 1049-2127.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, v. 33, n. TR/SE-0401, p. 28, 2004. ISSN 13537776.

KLUSCH, M. Information agent technology for the internet: A survey. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The

Netherlands, The Netherlands, v. 36, n. 3, p. 337–372, mar. 2001. ISSN 0169-023X. <http://dx.doi.org/10.1016/S0169-023X(00)00049-5>.

KöPCKE, H.; RAHM, E. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 69, n. 2, p. 197–210, fev. 2010. ISSN 0169-023X. <http://dx.doi.org/10.1016/j.datak.2009.10.003>.

LANTHALER, M. Creating 3rd Generation Web APIs with Hydra. In: *Proceedings of the 22nd International World Wide Web Conference (WWW2013)*. [S.l.]: International World Wide Web Conferences Steering Committee, 2013. p. 35–37.

LANTHALER, M.; GüTL, C. On Using JSON-LD to Create Evolvable RESTful Services. In: *Proceedings of the Third International Workshop on RESTful Design*. New York, NY, USA: ACM, 2012. (WS-REST '12), p. 25–32. ISBN 978-1-4503-1190-8. <http://doi.acm.org/10.1145/2307819.2307827>.

LEITE, L. et al. A systematic literature review of service choreography adaptation. *Service Oriented Computing and Applications*, v. 7, n. 3, p. 199–216, 2013.

LEMOS, A. L.; DANIEL, F.; BENATALLAH, B. Web Service Composition: A Survey of Techniques and Tools. *ACM Computing Surveys*, v. 48, n. 3, p. 1–41, dec 2015. ISSN 03600300. <http://dl.acm.org/citation.cfm?doid=2856149.2831270>.

LIRA, H. A. et al. Semantic data services: An approach to access and manipulate Linked Data. In: *2014 XL Latin American Computing Conference (CLEI)*. [S.l.]: IEEE, 2014. p. 1–12. ISBN 978-1-4799-6130-6.

MARJIT, U. et al. Publishing legacy data as linked data: a state of the art survey. *Library Hi Tech*, v. 31, n. 3, p. 520–535, sep 2013. ISSN 0737-8831.

MÁRMOL, F. G.; KUHNEN, M. Q. Reputation-based Web service orchestration in cloud computing: A survey. *Concurrency Computation*, v. 27, n. 9, p. 2390–2412, 2015.

MARTIN-FLATIN, J. P.; LöWE, W. Special Issue on Recent Advances in Web Services. *World Wide Web*, v. 10, n. 3, p. 205–209, ago. 2007. ISSN 1386-145X. <http://link.springer.com/10.1007/s11280-007-0035-8>.

MATTHIAS, K.; KANE, S. P. *Docker: Up and Running.* First edit. Gravenstein Highway North, Sebastopol, CA 95472, USA.: O'Reilly Media, Inc., 2015. ISBN 9781491917572.

MAZUREK, C. et al. CLEPSYDRA Data Aggregation and Enrichment Framework: Design, Implementation and Deployment in the PIONIER Network Digital Libraries Federation. In: BEMBENIK, R. et al. (Ed.). *Studies in Computational Intelligence.* Cham: Springer International Publishing, 2014, (Studies in Computational Intelligence, v. 541). p. 275–288. ISBN 978-3-319-04713-3.

MCILRAITH, S.; SON, T.; ZENG, H. Z. H. Semantic Web services. *IEEE Intelligent Systems*, v. 16, n. 2, p. 46–53, mar. 2001. ISSN 1541-1672. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=920599>.

MEYMANDPOUR, R.; DAVIS, J. G. A semantic similarity measure for linked data: An information content-based approach. *Knowledge-Based Systems*, v. 109, n. October, p. 276–293, oct 2016. ISSN 09507051.

MURGUZUR, A. et al. Process flexibility in service orchestration: A systematic literature review. *International Journal of Cooperative Information Systems*, v. 23, n. 3, 2014.

NACER, H.; AISSANI, D. Review: Semantic Web Services: Standards, Applications, Challenges and Solutions. *J. Netw. Comput. Appl.*, Academic Press Ltd., London, UK, UK, v. 44, p. 134–151, sep 2014. ISSN 1084-8045. <http://dx.doi.org/10.1016/j.jnca.2014.04.015>.

NAZMUDEEN, M.; BUHARI, S. A survey on distributed service discovery mechanisms with the focus on topology awareness. *Advances in Intelligent Systems and Computing*, v. 331, p. 315–326, 2015.

NEUMAIER, S. et al. Data Integration for Open Data on the Web. In: IANNI, G. et al. (Ed.). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Cham: Springer International Publishing, 2017, (Lecture Notes in Computer Science, v. 10370). p. 1–28. ISBN 978-3-319-61032-0.

NEWMAN, S. *Building Microservices.* Gravenstein Highway North, Sebastopol, CA. USA: O'Reilly Media, 2015. ISBN 9781491950357.

NGAN, L. D.; KANAGASABAI, R. Semantic Web service discovery: State-of-the-art and research challenges. *Personal and Ubiquitous Computing*, v. 17, n. 8, p. 1741–1752, 2013.

OLIVEIRA, B. C. N. et al. Automatic Semantic Enrichment of Data Services. In: *International Conference on Information Integration and Web-based Applications and Services - iiWAS '17*. [S.l.: s.n.], 2017. ISBN 9781450352994.

OLIVEIRA, B. C. N. et al. A platform to enrich, expand and publish linked data of police reports. In: ISAÍAS, P. (Ed.). *15th International Conference WWW/Internet*. Mannheim, Germany: IADIS Press, 2016. p. 111–118. ISBN 978-989-8533-57-9. <http://iadisportal.org/digital-library/a-platform-to-enrich-expand-and-publish-linked-data-of-police-reports>.

ORDÓNEZ, A. et al. Towards automated composition of convergent services: A survey. *Computer Communications*, v. 69, p. 1–21, 2015.

PAIK, H.-y. et al. Web Services – Data Services. In: *Web Service Implementation and Composition Techniques*. [S.l.]: Springer, 2017. p. 93–147. ISBN 978-3-319-55540-9.

PASQUIER, N. et al. Discovering frequent closed itemsets for association rules. In: *Proceedings of the 7th International Conference on Database Theory*. London, UK, UK: Springer-Verlag, 1999. (ICDT '99), p. 398–416. ISBN 3-540-65452-6.

PAVEL, S.; EUZENAT, J. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 25, n. 1, p. 158–176, jan. 2013. ISSN 1041-4347.

PINKEL, C. et al. DataOps: Seamless End-to-End Anything-to-RDF Data Integration. In: *The Semantic Web: ESWC 2015 Satellite Events*. [S.l.]: Springer, 2015. p. 123–127. ISBN 978-3-319-25639-9.

PLATENIUS, M. et al. A survey of fuzzy service matching approaches in the context of on-the-fly computing. In: *CBSE 2013 - Proceedings of the 16th ACM SIGSOFT Symposium on Component Based Software Engineering*. [S.l.: s.n.], 2013. p. 143–152.

POSTMAN, N. Book. *Building a bridge to the eighteenth century : how the past can improve our future / Neil Postman*. [S.l.]: Scribe Publications Carlton North, Vic, 1999. 213 p. ; p. ISBN 0908011407.

RICHARDS, M. *Microservices vs. Service-Oriented Architecture.* 1005 Gravenstein Highway North, Sebastopol, CA 95472. O'Reilly: O'Reilly Media, Inc., 2015. 1–55 p. ISBN 9781491952429. <https://www.nginx.com/microservices-soa/>.

RIETVELD, L. et al. Linked Data-as-a-Service: The Semantic Web Redeployed. In: GANDON, F. et al. (Ed.). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Cham: Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9088). p. 471–487. ISBN 978-3-319-18817-1.

ROSENBERG, F. et al. Composing restful services and collaborative workflows: A lightweight approach. *IEEE Internet Computing*, v. 12, n. 5, p. 24–31, Sept 2008. ISSN 1089-7801.

RUBENSTEIN-MONTANO, B. et al. A systems thinking framework for knowledge management. *Decision Support Systems*, v. 31, n. 1, p. 5 – 16, 2001. ISSN 0167-9236. Knowledge Management Support of Decision Making.

SALVADORI, I. et al. Publishing linked data through semantic microservices composition. In: *Proc. of Int. Conf. on Information Integration and Web-based Applications & Services.* [S.l.]: ACM, 2016. ISBN 978-1-4503-4807-2/16/11.

SALVADORI, I.; HUF, A.; SIQUEIRA, F. An Agent-based Composition Model for Semantic Microservices. In: ISAÍAS, P. (Ed.). *15th International Conference WWW/Internet.* Mannheim, Germany: IADIS Press, 2016. p. 75–82. ISBN 978-989-8533-57-9. <http://iadisportal.org/digital-library/an-agent-based-composition-model-for-semantic-microservices>.

SALVADORI, I. et al. An Ontology Alignment Framework for Data-driven Microservices. In: *Proceedings of the 19th International Conference on Information Integration and Web-based Applications and Services - iiWAS '17.* [S.l.: s.n.], 2017. ISBN 9781450352994.

SALVADORI, I. L. et al. Improving entity linking with ontology alignment for semantic microservices composition. *International Journal of Web Information Systems*, v. 13, n. 3, p. 302–323, aug 2017. ISSN 1744-0084. <http://www.emeraldinsight.com/doi/10.1108/IJWIS-04-2017-0029>.

SERRANO, D. et al. Linked REST APIs: A Middleware for Semantic REST API Integration. In: *2017 IEEE International Conference on Web Services (ICWS)*. [S.l.]: IEEE, 2017. p. 138–145. ISBN 978-1-5386-0752-7.

SIBBEL, R. *Cooperative Agents*. Dordrecht: Springer Netherlands, 2001. ISSN 978-90-481-5902-4. ISBN 978-90-481-5902-4. <http://link.springer.com/10.1007/978-94-017-1177-7>.

SINGH, M. P.; HUHNS, M. N. *Service-Oriented Computing*. Chichester, UK: John Wiley & Sons, Ltd, 2004. 1–549 p. (Lecture Notes in Computer Science, v. 4504). ISSN 0001-0782. ISBN 9780470091500. <http://doi.wiley.com/10.1002/0470091509 http://link.springer.com/10.1007/978-3-540-72619-7>.

SPEISER, S.; HARTH, A. Integrating Linked Data and Services with Linked Data Services. In: *The Semantic Web: Research and Applications*. [S.l.]: Springer, 2011. ISBN 978-3-642-21033-4.

STADTMÜLLER, S.; SPEISER, S.; HARTH, A. Future challenges for linked APIs. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2013. v. 1056, p. 20–27.

STRUNK, A. QoS-aware service composition: A survey. In: *Proceedings - 8th IEEE European Conference on Web Services, ECOWS 2010*. [S.l.: s.n.], 2010. p. 67–74.

SUCHANEK, F. M.; ABITEBOUL, S.; SENELLART, P. Paris: Probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 5, n. 3, p. 157–168, 2011.

SUN, L.; DONG, H.; ASHRAF, J. Survey of service description languages and their issues in cloud computing. In: *Proceedings - 2012 8th International Conference on Semantics, Knowledge and Grids, SKG 2012*. [S.l.: s.n.], 2012. p. 128–135.

SUN, L. et al. Cloud service selection: State-of-the-art and future research directions. *Journal of Network and Computer Applications*, v. 45, p. 134–150, 2014.

SYU, Y.; FANJIANG, Y. Y. *A Survey to Service Composition Methods Using Aspects Classification*. 2013. 170–181 p.

TAHERIYAN, M. et al. Rapidly integrating services into the linked data cloud. In: CUDRÉ-MAUROUX, P. et al. (Ed.). *The Semantic Web – ISWC 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 559–574. ISBN 978-3-642-35176-1.

TEKA, A. Y.; CONDORI-FERNANDEZ, N.; SAPKOTA, B. A systematic literature review on service description methods. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 7195 LNCS, p. 239–255, 2012.

TOSI, D.; MORASCA, S. Supporting the semi-automatic semantic annotation of web services: A systematic literature review. *Information and Software Technology*, v. 61, p. 16–32, May 2015. ISSN 09505849.

TRINH, T.-D. et al. Distributed mashups: a collaborative approach to data integration. *Int. Journal of Web Information Systems*, Emerald Group Publishing Limited, v. 11, n. 3, p. 370–396, aug 2015. ISSN 1744-0084.

VACULÍN, R. et al. Modeling and discovery of data providing services. In: *Proceedings of the IEEE International Conference on Web Services, ICWS 2008*. [S.l.]: IEEE, 2008. p. 54–61. ISBN 9780769533100.

WANG, L.; SHEN, J. A systematic review of bio-inspired service concretization. *IEEE Transactions on Services Computing*, PP, n. 99, 2015.

WANG, Y.; WANG, Y. A survey of change management in service-based environments. *Service Oriented Computing and Applications*, v. 7, n. 4, p. 259–273, 2013.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, v. 10, n. 02, p. 115, jun 1995. ISSN 0269-8889. <http://www.journals.cambridge.org/abstract_S0269888900008122>.

WU, X. b.; CHEN, C.; HUANG, H. A survey on web service composition: From service description, automatic process generation to process evaluation. *International Journal of Digital Content Technology and its Applications*, v. 6, n. 17, p. 483–495, 2012.

XIE, Z. et al. An evolvable and transparent data as a service framework for multisource data integration and fusion. *Peer-to-Peer Networking and Applications*, apr 2017. ISSN 1936-6442. <http://link.springer.com/10.1007/s12083-017-0555-7>.

ZAVERI, A. et al. Quality assessment for Linked Data: A Survey. *Semantic Web*, v. 7, n. 1, p. 63–93, mar 2015. ISSN 22104968.

ZILCI, B. I.; SLAWIK, M.; KUPPER, A. Cloud Service Matchmaking Approaches: A Systematic Literature Survey. In: *Proceedings of the 2015 26th International Workshop on Database and Expert Systems Applications (DEXA)*. Washington, DC, USA: IEEE Computer Society, 2015. (DEXA '15), p. 181–185. ISBN 978-1-4673-7582-5. <http://dx.doi.org/10.1109/DEXA.2015.50>.

ZIMMERMANN, O. Microservices tenets: Agile approach to service development and deployment. *Computer Science - Research and Development*, Springer Berlin Heidelberg, nov 2016. ISSN 1865-2034.