Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-Graduação em Ciência da Computação

DEFINING AND PROVIDING PRAGMATIC INTEROPERABILITY - THE MIDAS MIDDLEWARE CASE

Elivaldo Lozer Fracalossi Ribeiro

TESE DE DOUTORADO

Salvador
10 de Dezembro de 2020

ELIVALDO LOZER FRACALOSSI RIBEIRO

DEFINING AND PROVIDING PRAGMATIC INTEROPERABILITY - THE MIDAS MIDDLEWARE CASE

Esta Tese de Doutorado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientadora: Daniela Barreiro Claro
Co-orientadora: Rita Suzana Pitangueira Maciel

Salvador
10 de Dezembro de 2020

Sistema de Bibliotecas - UFBA

Ribeiro, Elivaldo Lozer Fracalossi.
Defining and Providing Pragmatic Interoperability - The MIDAS Middleware Case / Elivaldo Lozer Fracalossi Ribeiro – Salvador, 2020.
165p.: il.

Orientadora: Prof. Dr. Daniela Barreiro Claro.
Co-orientadora: Prof. Dr. Rita Suzana Pitangueira Maciel.
Tese (Doutorado) – Universidade Federal da Bahia, Instituto de Matemática, 2020.

1. Syntactic interoperability. 2. Semantic interoperability. 3. Pragmatic interoperability. 4. Conceptual framework. 5. Cloud Computing. 6. Cloud services. I. Claro, Daniela Barreiro. II. Maciel, Rita Suzana Pitangueira. III. Universidade Federal da Bahia. Instituto de Matemática. IV. Título.

CDD – XXX.XX
CDU – XXX.XX.XXX

TERMO DE APROVAÇÃO

ELIVALDO LOZER FRACALOSSI RIBEIRO

DEFINING AND PROVIDING PRAGMATIC INTEROPERABILITY - THE MIDAS MIDDLEWARE CASE

Esta Tese de Doutorado foi julgada adequada à obtenção do título de Doutor em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 10 de Dezembro de 2020

Profa. Dra. Daniela Barreiro Claro
Orientadora/PGCOMP

Profa. Dra. Laís do Nascimento Salvador
Membro interno/PGCOMP

Prof. Dr. Ivan do Carmo Machado
Membro interno/PGCOMP

Prof. Dr. Frank Augusto Siqueira
Membro externo/UFSC

Profa. Dra. Elisa Yumi Nakagawa
Membro externo/USP

I dedicate this work to my God and my family.

ACKNOWLEDGEMENTS

I believe that most people think that this part is merely a formality. I am afraid I have to disagree. This is one of the most important parts of this thesis. I am sure that this Ph.D. was an extraordinary, curious, and exhaustive journey. And I would not have arrived here alone. I could write several pages of acknowledgments, but I will try to summarize it in a few words.

First, I thank God for the opportunity and strength to complete this work. Thank you for Your endless blessing, and thank you for the people in my life.

I would like to express my gratitude to all my family members, physically near or far. I thank my parents, Elizabeth and Olivaldo, and my brother, Elivelton, for their affection and constant support. To my beloved wife, Adriadna, I thank you for her unconditional love, understanding, incentive, and solidarity. I appreciate the conditions for finishing this work, and I apologize for the recent absence and possible lousy mood. I love you so much. I am also very grateful to my father-in-law and mother-in-law for love and prayers.

I thank all my friends for their support and brotherhood. Thank you for the laughs, for the conversations, and all the affection.

I am deeply indebted to my advisers Daniela B. Claro and Rita Suzana P. Maciel. Thank you for the valuable guidance, generosity, partnership, hundreds of advice, and patience. Thank you for patiently reviewing the same paper countless times. Thank you Daniela. Thank you Rita. Thank you so much.

This thesis could not be achieved without the support of the Formalisms and Semantic Applications (FORMAS) research group. Thank you very much to all my friends from the FORMAS group.

I am deeply thankful to professor Marlo Souza for his contributions, willingness, and support.

I also would like to thank the Federal University of Southern Bahia (UFSB) and *Fundação de Amparo à Pesquisa do Estado da Bahia* (Fapesb) for partially supporting this work.

My apologies for the ones I forgot. My sincere thanks.

If I have seen a little further, it is by standing on the shoulders of giants.

—ISAAC NEWTON

RESUMO

Sistemas de informação modernos estão se tornando cada vez mais complexos. Essa complexidade está relacionada com a necessidade de combinar softwares heterogêneos. Uma vez que um sistema pode conter diversos softwares, e cada software pode ser desenvolvido de maneira independente, não é uma tarefa trivial prover uma comunicação transparente entre sistemas distintos. A falta de padronização ocasiona um problema conhecido como *lock-in*. Situações de *lock-in* ocorrem quando usuários se tornam dependentes de um sistema devido à falta de interoperabilidade entre os provedores distintos. A interoperabilidade é a capacidade de um sistema se comunicar de forma transparente com outro sistema e tem sido classificada em três níveis: sintático, semântico e pragmático. O nível sintático permite que informações sejam trocadas entre sistemas a partir de uma codificação comum. A interoperabilidade semântica está relacionada com a capacidade dos sistemas compartilharem o mesmo significado dos dados. Por fim, a pragmática tem a intenção da mensagem que deve ser entendida pelos sistemas, de modo que o resultado produzido esteja dentro das expectativas comuns. Apesar dos diversos níveis, as soluções para interoperabilidade entre sistemas focam apenas em uma camada específica. A ausência de um modelo para a interoperabilidade pragmática dificulta a comunicação transparente entre sistemas, pois as informações necessárias para interoperar não são explícitas. Além disso, o nível pragmático requer o nível semântico que, por sua vez, necessita do nível sintático. Além da necessidade de interoperar sistemas heterogêneos, as tecnologias atuais apresentam os desafios de armazenar, processar e disponizar os dados gerados por essa comunicação. A computação em nuvem tem o objetivo de atender alguns desses requisitos. A computação em nuvem é um paradigma que permite acesso a uma rede ubíqua de aplicações, plataformas e *hardware* como serviços. Esses serviços são organizados em níveis e acessados sob demanda com uma política de pagamento baseado no uso. *Software as a Service* (SaaS), *Platform as a Service* (PaaS), *Infrastructure as a Service* (IaaS) e *Data as a Service* (DaaS) são exemplos de serviços em nuvem. Assim, essa tese apresenta um *framework* conceitual para interoperabilidade pragmática (CAPITAL) que considera os níveis sintático e semântico. Com o intuito de validar o *framework* CAPITAL, o modelo para interoperabilidade sintática fornece uma descrição detalhada dos elementos sintáticos do *middleware* MIDAS *(Middleware for DaaS and SaaS)*. O modelo para interoperabilidade semântica auxilia e formaliza a comunicação semântica entre SaaS e DaaS. O *framework* CAPITAL descreve o modelo para interoperabilidade pragmática. Três estudos foram realizados para avaliar nosso *framework* CAPITAL. No primeiro estudo, o *framework* CAPITAL foi simulado em quatro cenários distintos com o objetivo de fornecer um guia de modelagem e codificação. O segundo estudo é um experimento controlado que investiga se nosso *framework* auxilia a compreensão do conceito e interpretação de cenários com interoperabilidade pragmática. No terceiro estudo, nosso *framework* foi incorporado ao MIDAS como prova de conceito com o objetivo de discutir e apresentar uma versão do *middleware* para interoperabilidade pragmática. Nossos estudos sugerem que o CAPITAL *framework* influencia positivamente no entendimento, modelagem e padronização de cenários com interoperabilidade pragmática. Os resultados alcançados fornecem evidências que os modelos para interoperabilidade sintática, semântica e pragmática descrevem os elementos necessários para prover uma comunicação transparente.

Palavras-chave: Interoperabilidade sintática, interoperabilidade semântica, interoperabilidade pragmática, framework conceitual, computação em nuvem, serviços em nuvem

ABSTRACT

Modern information systems are becoming increasingly complex. This complexity is related to the need to combine heterogeneous software. Since a system may contain many software programs, and each software may be developed independently, providing transparent communication between heterogeneous systems is not a trivial task. The lack of standardization causes a problem known as lock-in. Lock-in situations occur when users are dependent on a system due to the lack of interoperability among different providers. Interoperability is heterogeneous systems' ability to communicate transparently, and it is classified into three levels: syntactic, semantic, and pragmatic. The syntactic level enables systems to exchange information based on standard coding. Semantic interoperability is concerned with ensuring that systems to share the same data meaning. Finally, pragmatic interoperability ensures that systems understand the message intention so that the result is within common expectations. Despite the various levels, solutions for interoperability among systems focus on a specific layer. The absence of a pragmatic interoperability model hinders transparent communication among systems because the mandatory information to interoperate is not explicit. The pragmatic level requires the semantic level that, in turn, requires the syntactic level. In addition to the need of interoperate heterogeneous systems, current technologies present the challenges of storing, processing, and making available the data generated by this communication. Cloud Computing aims to fulfill some of these requirements. Cloud Computing is a new paradigm that enables access to a ubiquitous network of applications, platforms, and hardware as services. These services are organized in levels, and they are accessed with a pay-per-use policy. Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Data as a Service (DaaS) are examples of cloud services. Therefore, this work presents a Conceptual frAmework for Pragmatic InTeroperAbiLity (CAPITAL). Although focused on pragmatic interoperability, the CAPITAL framework considers the syntactic and semantic levels. We evaluate our CAPITAL framework in threefolds, one for each model. The model for syntactic interoperability provides a detailed description of syntactic elements of MIDAS (Middleware for DaaS and SaaS). The model for semantic interoperability provides an ontology to formalize the communication between SaaS and DaaS. This ontology assists the semantic interoperability of MIDAS. The CAPITAL framework describes the model for pragmatic interoperability. We perform three studies to evaluate our CAPITAL framework. In the first study, we modeled the CAPITAL framework in four distinct scenarios that aim to provide a modeling and coding guide. The second study is a controlled experiment that investigates whether our framework eases to understand the concept and interpret scenarios with pragmatic interoperability. In the third study, we incorporated our framework into MIDAS as a proof of concept to discuss and present a middleware version for pragmatic interoperability. The three studies suggest that the CAPITAL framework positively influences the understanding, modeling, and standardization of scenarios with pragmatic interoperability. Our findings provide evidence that models for syntactic, semantic, and pragmatic interoperability describe mandatory elements to provide transparent communication.

Keywords: Syntactic interoperability, semantic interoperability, pragmatic interoperability, conceptual framework, cloud computing, cloud services

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACID	Atomicity, Consistency, Isolation, and Durability	24
API	Application Programming Interface	11
CAP	Consistency, Availability, and Partition	25
CAPITAL	Conceptual frAmework for Pragmatic InTeroperAbiLity	3
CSV	Comma Separated Values	13
DaaS	Data as a Service	1
DBaaS	Database as a Service	1
DBMS	Database Management System	24
DIS	Dataset Information Storage	31
DL	Description Logic	34
IaaS	Infrastructure as a Service	1
IoT	Internet of Things	1
JSON	JavaScript Object Notation	13
GQM	Goal-Question-Metric	98
MIDAS	Middleware for DaaS and SaaS	4
NIST	National Institute of Standards and Technology	19
NoSQL	Not Only SQL	24
OBQA	Ontology-Based Query Answering	64
OWL	Ontology Web Language	53
PaaS	Platform as a Service	1
SaaS	Software as a Service	1
SPARQL	SPARQL Protocol and RDF Query Language	63
SQL	Structured Query Language	24
SMS	Semantic Mapping Storage	35
URL	Uniform Resource Locator	31
XML	eXtensible Markup Language	13

Chapter

1

This chapter provides an overview of the research topic. Here, we describe the problem, research questions, hypothesis, objectives, research method, and the thesis outline.

INTRODUCTION

In the last decades, companies have been combining a high number of heterogeneous software to support their increasingly complex business rules. Nowadays, a single enterprise usually performs dozens of applications, such as different operating systems and databases (POKRAEV, 2009).

The advent of the Internet of Things (IoT), social networks, and mobile devices has boosted the combined use of heterogeneous software (ARMBRUST et al., 2010). A successful combination enables two or more systems to work together to achieve a specific objective. We define interoperability as the ability of multiple systems to work together to provide transparent communication regardless of the providers' differences (LOUTAS et al., 2011; ZHANG; WU; CHEUNG, 2013).

Interoperability is a challenge due to the systems' lack of standardization, and this lack of standardization causes a vendor lock-in problem (LOUTAS et al., 2011). The vendor lock-in problem occurs when a consumer becomes captive to a solution due to the lack of standardization among systems (SCOTT, 1996; OPARA-MARTINS; SAHANDI; TIAN, 2014).

Besides the challenge of interoperating heterogeneous systems, the volume of digital data generated by these collaborations grows exponentially (REINSEL; GANTZ; RYDNING, 2018). Consequently, this data needs to be stored and available to both consumers and organizations anytime and anywhere. Cloud Computing has emerged to fulfill some of these requirements (MELL; GRANCE, 2011).

Cloud Computing is a paradigm that enables the access to a ubiquitous and on-demand network of logical and physical resources (e.g., applications, platforms, and hardware) as services (ARMBRUST et al., 2010). Software as a Service (SaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Data as a Service (DaaS), and Database as a Service (DBaaS) are instances of cloud services (MELL; GRANCE, 2011; HACIGUMUS; IYER; MEHROTRA, 2002; TERZO et al., 2013).

Cloud Computing facilitates the combination, use, and management of heterogeneous resources through a pay-per-use policy. In this policy, end consumers pay only for actual consumption, and the entire infrastructure enables delivering these services, depending on the consumers' needs (VECCHIOLA; CHU; BUYYA, 2009).

The pay-per-use policy is the same concept applied to basic and essential services, such as water, gas, electricity, and telephone. A company provides a utility service and end-users consume this service anytime, anywhere, with payment according to their use. Although the user is unaware, a complex infrastructure allows the delivery of these services transparently (VECCHIOLA; CHU; BUYYA, 2009).

1.1 PROBLEM STATEMENT

Interoperability is an essential requirement for systems to perform together cooperatively (BRAVO; ALVARADO, 2008). Rimal et al. (2011) group the Cloud Computing requirements into providers, enterprises, and consumers. This classification emphasizes the importance of interoperability in the Cloud Computing domain since only the *quality of service*, *scalability*, and *interoperability* belong to more than one group.

Although not the only classification (ASUNCION; IACOB; SINDEREN, 2010; MACIEL et al., 2017; VALLE; GARCÉS; NAKAGAWA, 2019), interoperability may be classified into syntactic, semantic, and pragmatic levels (ASUNCION; SINDEREN, 2010). Despite the different levels, interoperability proposals provide solutions focused on a specific level, i.e., solutions are focused on the syntactic, semantic, or pragmatic level separately. In addition to making interoperability difficult, the absence of a solution involving different levels affects the quality and cost of the systems. An "isolated solution" does not detail the data necessary to interoperate. Therefore, the main problem investigated in the scope of this thesis is:

> *Communication among cloud services lacks adequate support for detailing data necessary to provide pragmatic interoperability.*

1.2 RESEARCH QUESTIONS AND HYPOTHESIS

This thesis is related to the challenges and opportunities of Cloud Computing. Based on interoperability among different clouds, we define the following main research question to guide this research:

> *How to provide pragmatic interoperability among cloud services?*

We decompose the above research question into the following sub-questions:

- *How to describe syntactic interoperability among different cloud services?*

  The objective of this research question is to describe syntactic interoperability among cloud services. Based on the bottom-up approach, we present a lightweight formal description of syntactic interoperability among different cloud services. Our syntactic description is based on canonical models and mathematical concepts. Chapter 5 focused on answering this question.

- *How to describe semantic interoperability among different cloud services?*

  The objective of this research question is to describe semantic interoperability among cloud services. Based on the bottom-up approach, we present an OWL-based ontology to represent communication among different cloud services formally. Our ontology enables explicitly expressing the semantic similarity among DaaS and DBaaS attributes. This semantic similarity enables semantic integration through queries rewriting. Chapter 6 focused on answering this question.

- *What elements are needed to provide pragmatic interoperability?*

  This research question aims to determine which elements are mandatory to provide pragmatic interoperability among systems. The heterogeneity of the scenarios modeled in our Conceptual frAmework for Pragmatic InTeroperAbiLity (CAPITAL) framework illustrated the mandatory information. We suggest these elements based on the intersection of the results of each scenario. Chapter 7 focused on answering this question.

- *How to provide a generic model capable of representing pragmatic interoperability among systems?*

  The objective of this research question is to provide a generic solution to represent pragmatic interoperability among systems. Initially, since there is no consensus on the literature, we present our pragmatic interoperability unified definition. Afterwards, we present CAPITAL, our Conceptual frAmework for Pragmatic InTeroperAbiLity. The CAPITAL framework is based on canonical models, textual definition, and schemas in Z notation. Our framework represents and details pragmatic scenarios. Chapter 7 focused on answering this question.

- *How to provide pragmatic interoperability among cloud services?*

  The objective of this research question is to achieve pragmatic interoperability among cloud services. In this stage, we apply our CAPITAL framework to the Cloud Computing domain. Few works are addressing pragmatic interoperability in Cloud Computing, as well as solutions to solve this problem. Generally, these researches are domain-specific, and the elements of the communication are non-consensual. To the best of our knowledge, CAPITAL framework is the first attempt to provide pragmatic interoperability among cloud services. Chapter 8 focused on answering this question.

This thesis is based on the hypothesis that it is possible to provide pragmatic interoperability among cloud services through our CAPITAL framework. Before that, it is necessary (i) to describe both syntactic and semantic interoperability and (ii) to provide a consensual definition of pragmatic interoperability.

1.3 OBJECTIVES

The main objective of this thesis is **to investigate and provide a model for pragmatic interoperability between SaaS and DaaS/DBaaS cloud services**. Since pragmatic interoperability requires both syntactic and semantic levels (ASUNCION; SINDEREN, 2010), our methodology envisions both interoperability levels.

The specific objectives of this work are described as follows:

1. present a lightweight formal description of syntactic interoperability between SaaS and DaaS cloud services;
2. evaluate the correctness of the description for syntactic interoperability;
3. develop an ontology to represent the semantic interoperability between SaaS and DaaS/DBaaS cloud services;
4. assess the consistency, acceptance, and correctness of our ontology;
5. determine data needed to provide pragmatic interoperability among systems;
6. provide a definition for pragmatic interoperability based on literature;
7. develop a framework to represent pragmatic interoperability among systems;
8. evaluate the effectiveness, completeness, correctness, understandability, consistency, conciseness, and performance of our framework; and
9. investigate the framework for pragmatic interoperability among cloud services;

Objectives 1 and 2 refer to the syntactic stage, objectives 3 and 4 refer to the semantic stage, and the other specific objectives refer to pragmatic interoperability.

1.4 RESEARCH METHOD

Based on our objectives and research questions, we applied a combination of strategies to an in-depth understanding of our research problem (MIGUEL; MORABITO; PUREZA, 2010; WAZLAWICK, 2017).

Pragmatic interoperability depends on the previous consolidation of both syntactic and semantic interoperability (ASUNCION; SINDEREN, 2010). For this reason, the present investigation can be split into two main parts: literature review and models. Figure 1.1 presents both parts in detail.

Part I (Background) comprises an overview of the existing literature on interoperability and Cloud Computing. We performed this review focused on SaaS and DaaS cloud services. This analysis investigated the interoperability among different cloud services. Finally, we present in-depth the Middleware for DaaS and SaaS (MIDAS) (MARINHO et al., 2016; VIEIRA et al., 2017; SILVA; RIBEIRO; CLARO, 2018; RIBEIRO et al., 2018, 2019; MANE et al., 2020). MIDAS is a middleware to interoperate DaaS and SaaS among different clouds. We present MIDAS in detail because we use this middleware in the experiments performed with the three models.

Part II (CAPITAL Framework) corresponds to the construction of the three models. Initially, we present the model for syntactic interoperability. We describe the lightweight formal description of MIDAS syntactic interoperability, and we evaluate this

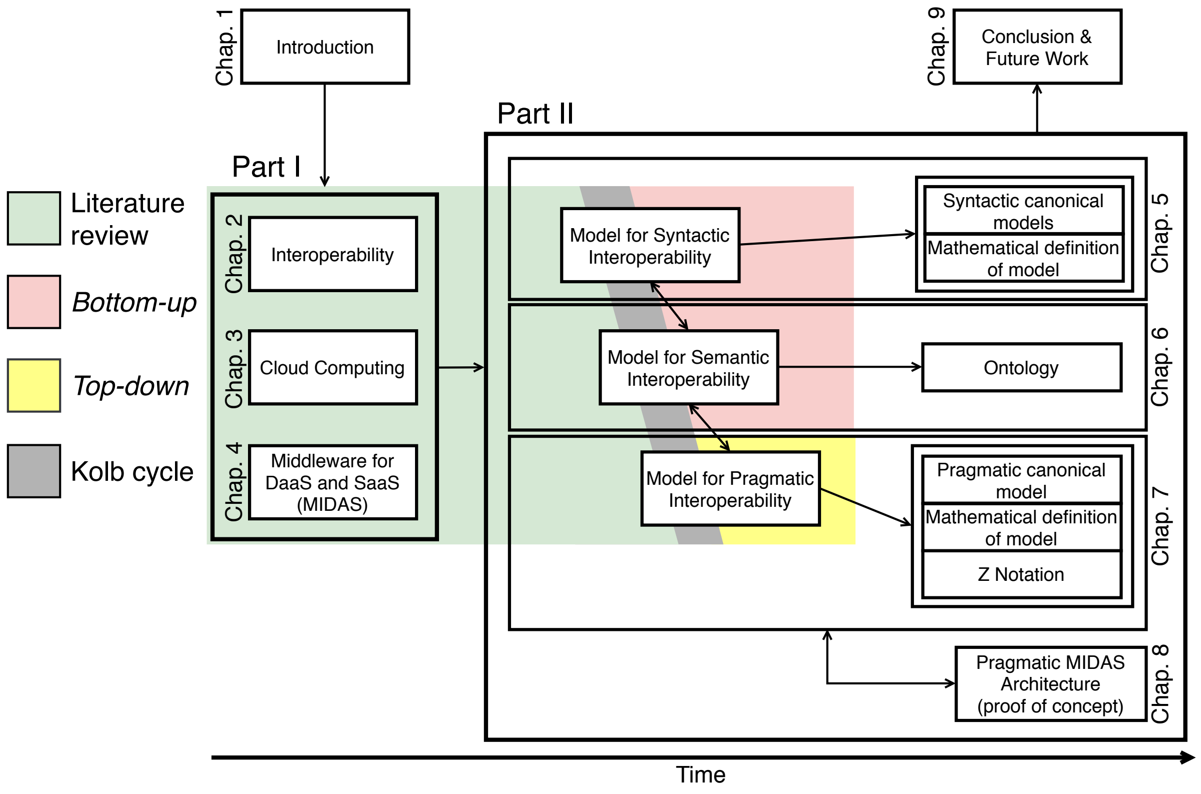

Chap. 1
Introduction
Chap. 9
Conclusion & Future Work
Part II
Part I
Literature review
Bottom-up
Top-down
Kolb cycle
Chap. 2
Interoperability
Chap. 3
Cloud Computing
Chap. 4
Middleware for DaaS and SaaS (MIDAS)
Model for Syntactic Interoperability
Syntactic canonical models
Mathematical definition of model
Chap. 5
Model for Semantic Interoperability
Ontology
Chap. 6
Model for Pragmatic Interoperability
Pragmatic canonical model
Mathematical definition of model
Z Notation
Chap. 7
Pragmatic MIDAS Architecture (proof of concept)
Chap. 8
Time

Figure 1.1 Research method of this thesis.

model through a set of canonical models and mathematical definitions. The model for semantic interoperability presents an OWL-based ontology of MIDAS semantic interoperability (*MIDAS-OWL*). We evaluate this model through SaaS queries submitted by MIDAS. Our ontology enables semantic integration through queries rewriting. Finally, we present the CAPITAL framework, our model for pragmatic interoperability. We describe this framework based on canonical models, textual definition, and schemas in Z notation. In the CAPITAL evaluation, we (i) apply our framework into different scenarios as modeling and implementation guidance and (ii) perform a controlled experiment. At the end of this part, we discuss the pragmatic MIDAS architecture. We also performed a proof of concept to evaluate the CAPITAL framework. We evaluated the three levels of interoperability together and separately.

According to Figure 1.1, this thesis applies three methodological approaches: bottom-up, top-down, and Kolb cycle. We apply Bottom-up to build the models for syntactic and semantic interoperability. At this stage, this research contributed to the development of the MIDAS middleware. We present and evaluate the models to syntactic and semantic interoperability based on this solution. We built the model for pragmatic interoperability based on the top-down approach, i.e., we developed this model independently of any solution.

We apply the Kolb cycle (KOLB, 1984) in all three models. Kolb's methodology describes that the construction or learning process is based on the continuous and cyclical execution of four stages: *act*, *reflect*, *conceptualize*, and *apply*. We adapted and performed this methodology in a continuous and cyclical execution of the first three steps (*act*, *reflect*, and *conceptualize*). We perform the fourth step (*apply*) after obtaining a model for syntactic, semantic, or pragmatic interoperability. This methodology allowed us to adjust any model at any stage.

1.5 THESIS OUTLINE

Figure 1.2 presents the thesis outline. Apart from the introduction chapter, this thesis is organized as follows.

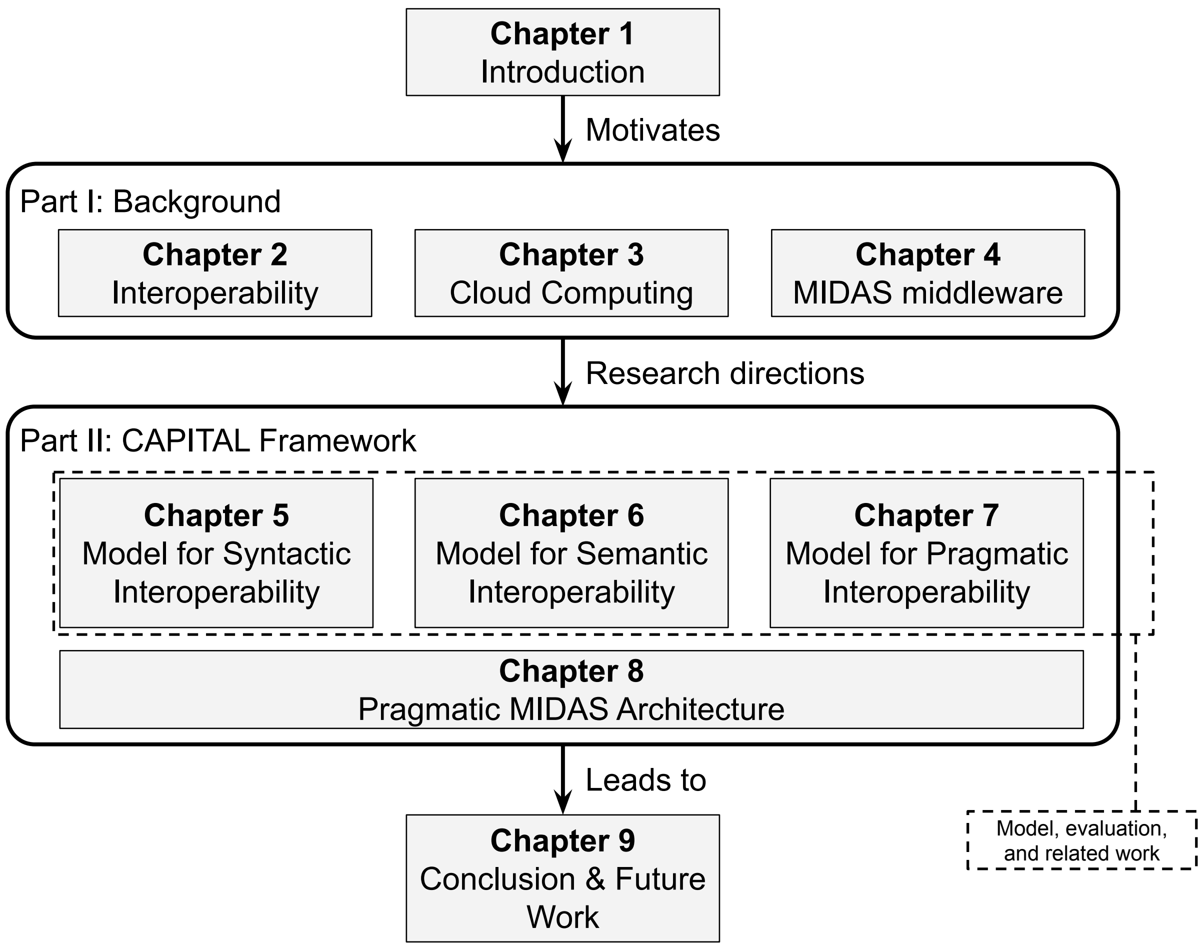


Figure 1.2 Overview of the thesis outline.

- **Background**: Part I provides background concepts on the topics involved in this study. This part presents state-of-the-art review of interoperability, Cloud Computing, and MIDAS middleware.

  - **Chapter 2** conceptualizes interoperability and details the syntactic, semantic and pragmatic levels.
  - **Chapter 3** defines Cloud Computing and describes characteristics, challenges, deployment models, and some services, e.g., SaaS, PaaS, IaaS, DaaS, and DBaaS.
  - **Chapter 4** introduces MIDAS middleware, a solution used as a basis for model evaluation. This chapter provides a MIDAS timeline and compares the MIDAS versions.

- **CAPITAL Framework**: Part II motivates and presents in detail the models for syntactic, semantic, and pragmatic interoperability.
  - **Chapter 5** details the model for syntactic interoperability. In this chapter, we report model evaluation into MIDAS and some related work.
  - **Chapter 6** presents our ontology to represents the MIDAS semantic interoperability (*MIDAS-OWL*). In this chapter, we report the ontology evaluation and some related work.
  - **Chapter 7** introduces CAPITAL, our conceptual framework for pragmatic interoperability. In this chapter, we report the CAPITAL framework evaluation (based on four distinct scenarios and a controlled experiment) and some related work.
  - **Chapter 8** presents the pragmatic MIDAS architecture. In this chapter, we discussed the workings of pragmatic MIDAS and then we perform the CAPITAL framework with the MIDAS middleware.

- **Conclusions**: Chapter 9 presents our concluding remarks, summarizes the contributions, and discusses future research directions.

PART I

BACKGROUND

Chapter

2

This chapter presents the interoperability concept. We describe syntactic, semantic, and pragmatic interoperability levels, and we discuss their challenges and benefits.

INTEROPERABILITY

Moderns solutions mean a variety of technologies, such as programming languages, operating systems, databases, Application Programming Interface (API), protocols, data formats, among others (ZHANG; WU; CHEUNG, 2013). The heterogeneity of technologies hinders two or more systems from working together to solve a specific problem (ZHANG; CHENG; BOUTABA, 2010; ARMBRUST et al., 2010; SAHANDI; ALKHALIL; OPARA-MARTINS, 2013). In this situation, users are dependent on the provider since they cannot easily change nor move their applications. This problem caused by dependence on the provider is known as lock-in (OPARA-MARTINS; SAHANDI; TIAN, 2016).

Vendor lock-in problem occurs when customers are dependent (i.e., locked-in) of a specific provider, among other factors, due to the different technologies adopted by different providers (SILVA; ROSE; CALINESCU, 2013; OPARA-MARTINS; SAHANDI; TIAN, 2014, 2016; LOUTAS et al., 2011). This heterogeneity hampers communication, and the absence of communication hinders portability, interoperability, and integration among systems (ARDAGNA et al., 2012; OPARA-MARTINS; SAHANDI; TIAN, 2016). Although portability, interoperability, and integration have similar objectives, they have some relevant differences.

Portability is the ability to migrate and reuse data or software components in a different provider (MELL; GRANCE, 2011; SILVA; ROSE; CALINESCU, 2013; OPARA-MARTINS; SAHANDI; TIAN, 2014). Portability may occur due to the difficulty of moving virtual machines, code, data, among others (OPARA-MARTINS; SAHANDI; TIAN, 2014).

Interoperability is the ability of heterogeneous systems to communicate transparently (MELL; GRANCE, 2011; CHEN; DOUMEINGTS; VERNADAT, 2008). Interoperability enables two or more systems to understand one another and use one another's services (CHEN; DOUMEINGTS; VERNADAT, 2008).

Integration is the process of combining multiple applications, resources, or services to function together as a unified whole. For instance, when an enterprise integrates various technologies into a single application to solve a specific problem. Integration is one solution to achieves interoperability (POKRAEV, 2009; GRAVINA et al., 2017).

Figure 2.1 depicts the difficulty of performing a) portability and b) interoperability between two providers. Consumers in Figure 2.1(a) endeavor to migrate their data from provider A to provider B, while consumers in Figure 2.1(b) attempt to establish communication between providers A and B. The lack of a standard makes both tasks difficult. Among the various causes of vendor lock-in, we are interested only in the lack of interoperability.

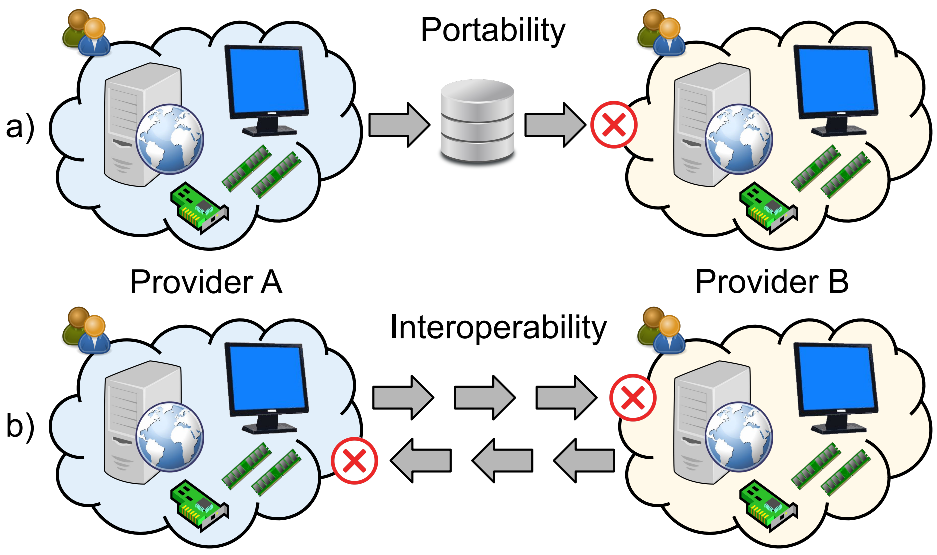


Figure 2.1 An overview of lack of a) portability and b) interoperability.

Successful transactions among systems require multiple layers of interoperability. Despite the absence of consensus, interoperability may be described into syntactic, semantic, and pragmatic levels. Asuncion, Iacob and Sinderen (2010) argue that interoperability levels surpass those presented by most works (syntactic, semantic, and pragmatic) with two other levels after the pragmatic: dynamic and organizational (or conceptual).

In summary, (i) syntactic level ensures the exchange of information among systems based on message standard, (ii) semantic interoperability is concerned with the communication meaning, (iii) pragmatic interoperability provides that systems share the same communication intention, (iv) at the dynamic level, systems take into account business rules, and (v) at the organizational level, systems perform all levels described previously (WANG; TOLK; WANG, 2009; ASUNCION; IACOB; SINDEREN, 2010; MACIEL et al., 2017; VALLE; GARCÉS; NAKAGAWA, 2019).

Another related interoperability is full interoperability (MACIEL et al., 2017). Full interoperability is achieved when a system reaches all desired interoperability levels. For instance, some systems may require only syntactic interoperability, while other systems may require syntactic, semantic, and pragmatic levels.

Although the literature presents different levels (ASUNCION; IACOB; SINDEREN, 2010; MACIEL et al., 2017; VALLE; GARCÉS; NAKAGAWA, 2019), this thesis contemplates syntactic, semantic, and pragmatic levels (ASUNCION; SINDEREN, 2010). The next sections describe these interoperability levels.

2.1 SYNTACTIC

Syntactic interoperability concerns the format of messages exchanged among systems. This level provides systems with an understanding of the message structure, and the messages exchanged require a well-defined syntax and encoding (ZARKO et al., 2019; KUBICEK; CIMANDER; SCHOLL, 2011).

When the sender encodes and sends a message, the receiver must be able to decode the message according to the same syntactic rules applied by the sender. As a result, the receiver obtains the message initially sent. Syntactic interoperability problems arise when the sender and receiver rules are incompatible (MACIEL et al., 2017).

Syntactic interoperability is ensured when systems are aware of data format and protocols. Systems should have a compatible and consistent approach to structure and standardize exchanged data (ASUNCION; SINDEREN, 2010).

Syntactic interoperability emphasizes two heterogeneities: syntax and structure. Syntax heterogeneity regards the data types and formats heterogeneities. For instance, DaaS returns data in different formats, such as eXtensible Markup Language (XML), Comma Separated Values (CSV), or JavaScript Object Notation (JSON). Given this heterogeneity, each return has a different file format, such as *file.xml*, *file.csv*, or *file.json.* The use of a standard format may help solve syntax differences. However, some cases can not be standardized, such as DaaS return. End-consumer may choose the return format based on the formats available by the DaaS provider. A solution to the syntax heterogeneity problem is the conversion among possible formats. In the DaaS example, the conversion ensures a standardized format in communication (SHETH, 1999).

Structure heterogeneity arises when the same data is represented using different schemas of metadata. For instance, systems could standardize a format in communication. However, each system may use the format differently. A solution to the structure heterogeneity problem is to standardize messages structure, e.g., with a schema (SHETH, 1999).

Syntactic interoperability is the foundation for higher levels of interoperability, such as semantic and pragmatic interoperability (ASUNCION; SINDEREN, 2010; KUBICEK; CIMANDER; SCHOLL, 2011; ZARKO et al., 2019).

2.2 SEMANTIC

Semantic interoperability ensures that systems understand the meaning of the exchanged messages. This level enables systems to combine data to process it in a meaningful manner (KUBICEK; CIMANDER; SCHOLL, 2011; STRASSNER; DIAB, 2016).

Semantic interoperability is ensured when systems understand the meaning of the syntactic elements, i.e., data exchanged must have the same meaning for the sender and the receiver (ASUNCION; SINDEREN, 2010). Any receiver must understand the data exchanged format since the meaning of messages must not change within the domain (MACIEL et al., 2017).

Some semantic representation problems arise when the sender and the receiver have a different understanding of the message (POKRAEV, 2009):

- Different systems use the **same symbol** to represent **different concepts**. For instance, *system A* uses the symbol "height" to represent "the height of a person", and *system B* uses the same symbol to represent "the height of a building" (see Figure 2.2(a)).

- Different systems use the **same symbol** to represent **overlapping concepts**. For instance, *system A* uses the symbol "blood" to represent "blood type A", and *system B* uses the same symbol to represent "blood Rh factor positive". This is important because not all "blood type A" is "Rh factor positive", and not all "Rh factor positive" is "blood type A" (see Figure 2.2(b)).

- Different systems use the **same symbol** to represent **more general (or specific) concepts**. For instance, *system A* uses the symbol "name" to represent "dog name", and *system B* uses the same symbol to represent "pet name". This causes doubt because "pet name" includes all "dog name" (see Figure 2.2(c)).

- Different systems use **different symbols** to represent the **same concept**. For instance, *system A* uses the symbol "car" and *system B* uses the symbol "automobile" to represent the same concept: "a wheeled motor vehicle used for transportation" (see Figure 2.2(d)).

- Different systems use **different symbols** to represent **overlapping concepts**. For instance, *system A* uses the symbol "student" to represent the concept of "someone enrolled in an educational institution" and *system B* uses the symbol "professor" to represent the concept of "someone who teaches at an educational institution". Both concepts may refer to the same entity in the real world since a "student" can be a "professor" and a "professor" can be a "student" (see Figure 2.2(e)).

- Different systems use **different symbols** to represent **more general (or specific) concepts**. For instance, *system A* uses the symbol "consumer" to represent the concept "client" and *system B* uses the symbol "partner" to represent the concept "client or seller" (see Figure 2.2(f)).

- Different systems use **different definition** of the **same concept**. For instance, *system A* defines "smartphone" as "a device that combines cell phone and handheld computer" and *system B* defines the same concept as "a device used as cell phone and handheld computer" (see Figure 2.2(g)).

Semantic interoperability is ensured when systems aware of messages language, taxonomy, and ontologies (ASUNCION; SINDEREN, 2010).

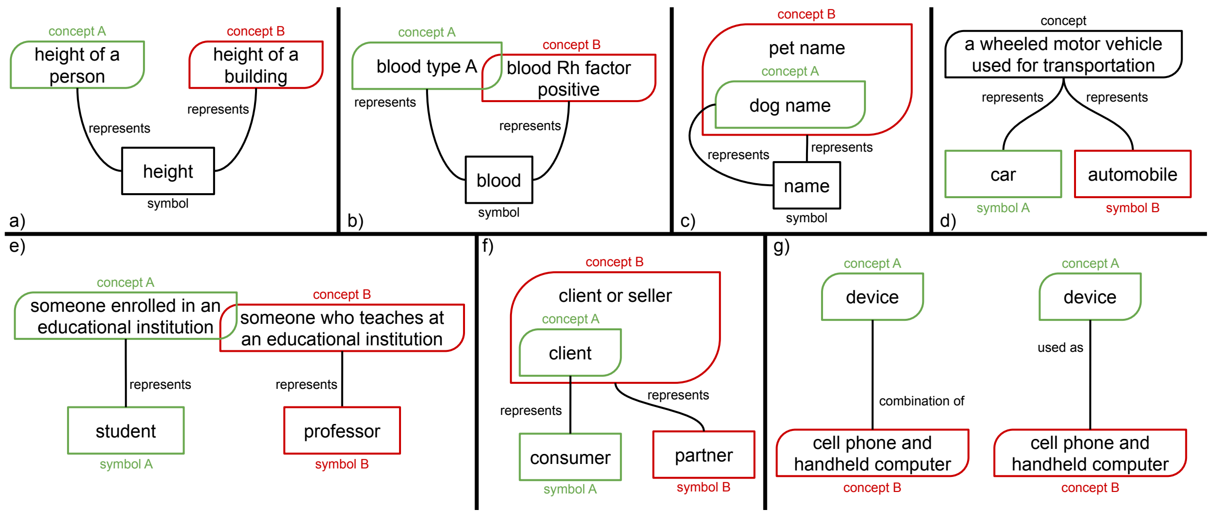


Figure 2.2 Some semantic representation problems: (a) same symbol for different concepts, (b) same symbol for overlapping concepts, (c) same symbol for more general (or specific) concepts, (d) different symbols for same concept, (e) different symbols for overlapping concepts, (f) different symbols for more general (or specific) concepts, and (g) different definition for same concept. Adapted from (POKRAEV, 2009).

2.3 PRAGMATIC

Although there is no consensus definition (ASUNCION; SINDEREN, 2010; MACIEL et al., 2017), pragmatic interoperability ensures that the sender and receiver understand the message context and intention in order to correctly use the message (MACIEL et al., 2017). This level enables a system to affect the state and behavior of another system (ASUNCION; IACOB; SINDEREN, 2010; ASUNCION; SINDEREN, 2010). Asuncion and Sinderen (2010) describe pragmatic interoperability as compatibility between the intended effect and actual effect of the message received within a context.

Differences between messages intended effect and actual effect occur because this level considers the context in which the messages are exchanged. Pragmatic interoperability problems arise when the sender and receiver differs (i) on messages *intended effect and actual effect*, or (ii) on the *interaction protocols* (MACIEL et al., 2017; POKRAEV, 2009). The most common problems with respect to interaction protocols are (POKRAEV, 2009):

- **Unexpected message**: *system A* sends a message that is not expected by *system B*. For instance, *system A* sends message M_1 and then message M_2 to *system B*. However, *system B* expects only the message M_2 (see Figure 2.3(a)).
- **Insufficient message**: *system B* intends to receive a message that is not sent by *system A*. For instance, *system B* expects to receive message M_1 and then message M_2 from *system A*. However, *system A* sends only the message M_2 (see Figure 2.3(b)).
- **Message order**: *system A* sends messages in a different order than expected by

system B. For instance, *system A* sends message M_1 and then message M_2. However, *system B* expects first the message M_2 and then message M_1 (see Figure 2.3(c)).

- **Message aggregation**: *system A* sends separately data that *system B* expects in a single message. For instance, *system A* sends message M_1 and then message M_2 to *system B*. However, *system B* expects only one message that aggregates M_1 and M_2 (see Figure 2.3(e)).
- **Message splitting**: *system B* intends to receive separately data that *system A* sent in a single message. For instance, *system B* intends to receive message M_1 and then message M_2 from *system A*. However, *system A* sends only one message that aggregates M_1 and M_2 (see Figure 2.3(f)).

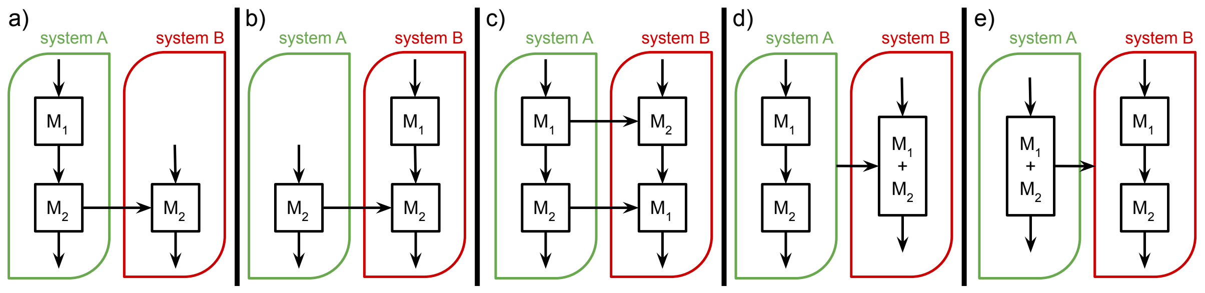


Figure 2.3 Some interaction protocol problems: (a) unexpected message, (b) insufficient message, (c) message order, (d) message aggregation, and (e) message splitting. Adapted from (POKRAEV, 2009).

Pragmatic interoperability can only be achieved after obtaining syntactic and semantic levels. Additionally, systems must be aware of the message context and intent (ASUNCION; SINDEREN, 2010).

2.4 CHAPTER SUMMARY

Interoperability is the ability of a system to share or exchange information with another system transparently.

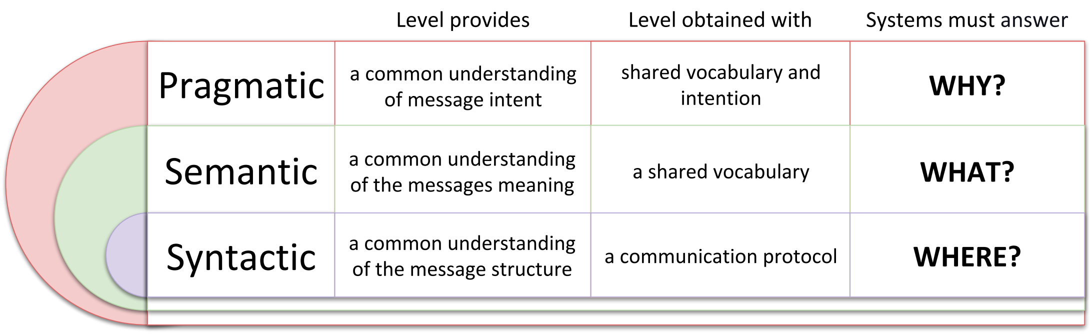
Some problems occur when systems communicate to share or exchange information, such as programming language differences or ambiguity in the development environment. Cooperation levels have been created to mitigate these problems. Despite the diversity of levels found in the literature, this thesis considers the syntactic, semantic, and pragmatic levels (ASUNCION; SINDEREN, 2010).

In syntactic interoperability, systems must be able to answer **where** other systems are. Systems achieve syntactic interoperability when, for example, a protocol defines the communication guidelines (including locations). In semantic interoperability, systems must be able to answer **what** other systems want unambiguously. Systems achieve semantic interoperability when, for example, an ontology represents all knowledge of the domain, and then the communication is carried out without ambiguity. In pragmatic interoperability, systems must be able to answer **why** a specific message was sent or received. Systems achieve pragmatic interoperability when they are aware of the intentions of the messages (van der VEER; WILES, 2008).

We noted that literature has advanced towards a hierarchical understanding of such levels. Nevertheless, although both syntactic and semantic levels have consensual definitions, the pragmatic interoperability definition is still diverse.

The lack of a standard understanding of pragmatic interoperability can result in (i) incompatible solutions, (ii) ambiguous interpretations of messages meaning, or (iii) difficulty in achieving the desired interoperability.

Figure 2.4 lists and relates the syntactic, semantic, and pragmatic interoperability. This idea of hierarchy among levels corroborates with the hypothesis that pragmatic interoperability is only achieved after obtaining syntactic and semantic interoperability (ASUNCION; SINDEREN, 2010).



	Level provides	Level obtained with	Systems must answer
Pragmatic	a common understanding of message intent	shared vocabulary and intention	**WHY?**
Semantic	a common understanding of the messages meaning	a shared vocabulary	**WHAT?**
Syntactic	a common understanding of the message structure	a communication protocol	**WHERE?**

Figure 2.4 Differences among interoperability levels. Adapted from (ADEBESIN et al., 2013).

The next chapter presents a general introduction of Cloud Computing, presenting their characteristics, challenges, business model, and some cloud services.

Chapter
3

This chapter presents the concepts about Cloud Computing. We describe the characteristics, challenges, deployment models, and service model.

CLOUD COMPUTING

According to the National Institute of Standards and Technology (NIST), Cloud Computing is a paradigm for enabling ubiquitous and on-demand access to a shared pool of logical and physical resources (e.g., applications, platforms, and hardware) as services. Cloud services can be rapidly provisioned and released with minimal management effort of providers (MELL; GRANCE, 2011). Zhang, Cheng and Boutaba (2010) state that Cloud Computing is a model that brings together a set of technologies to provide services differently. Cloud Computing leverages the advantage of existing technologies, such as *Grid Computing*, *Utility Computing*, *Virtualization*, and *Autonomic Computing*, to meet the requirements of today's demand. By 2025, about 49% of available data will be managed and stored by a cloud computing provider (REINSEL; GANTZ; RYDNING, 2018).

The major actors in traditional architecture are the consumers and providers. While consumers use, own, and maintain the services, the providers sell, install, license, consult, and maintain these services. In the Cloud Computing model, three new actors were added: auditors, brokers, and carriers. Auditors are responsible for assessing cloud services in terms of security, privacy, performance, among others. Brokers are entities that aim to integrate multiple services and intermediate the relationships between providers and consumers. Finally, carriers provide connectivity and transport among different cloud services (HOGAN et al., 2011).

One of the significant benefits of Cloud Computing is to enable consumers to use resources instead of installing them. Differently from traditional applications, modern applications must meet four major factors: *mobility*, *sharing*, *end of clock speed*, and *slow network speed growth* (SCHUBERT; JEFFERY, 2015). *Mobility* is the ability to access a service or application anytime and anywhere. *Sharing* is the ability of providers to deploy a service or distributed application and to consumers to access it remotely. *End of clock speed* concerns the strategies performed to overcome the limitations of current processors, such as parallelism. Finally, *slow network speed growth* involves the limitations of the network since the number of consumers (cores, processors, and devices) is growing faster than communication speed.

The following sections present the characteristics, deployment models, challenges, and service model.

3.1 ESSENTIAL CHARACTERISTICS

According to Mell and Grance (2011), the following characteristics distinguish the Cloud Computing model from traditional computing:

1. **On-demand self-service** enables consumers to unilaterally provision and release computing (e.g., server time and network storage) capabilities as needed without human interaction between the consumer and the provider. This feature provides efficiencies and cost savings to both consumers and cloud service providers since the consumers have the autonomy to manage resources.

2. **Broad network access** concerns the availability of services. This feature ensures that Cloud Computing services are available over the network to be accessed by heterogeneous devices, such as desktop, tablets, smartphones, smartwatch, among others.

3. **Resource pooling** enables Cloud Computing providers to pool services to serve multiple consumers based on the multi-tenant model, with physical and logical resources dynamically distributed. In most cases, consumers have no control or knowledge over the geographic location of the provided resources.

4. **Rapid elasticity** ensures that services can be elastically provisioned and released depending on demand, in some cases automatically. In the consumer's view, these services are available full-time and unlimitedly to be consumed anytime from anywhere.

5. **Measured service** concerns the payment method for the use of Cloud Computing resources. Cloud providers automatically control and optimize resource use based on the type of service, such as storage, processing, bandwidth, and active user accounts. Typically the payment is made on a pay-per-use policy. Resources are monitored, controlled, and reported, providing transparency for both the provider and consumer.

3.2 DEPLOYMENT MODELS

The structural models concern the models of development and availability of clouds to final consumers. In this classification, clouds can be public, private, community, or hybrid (MELL; GRANCE, 2011).

The **public cloud** infrastructure provides resources and services for open use by the general public. Although public, this cloud may be owned, managed, and operated by a business, academic, government organization, or some combination of them. Public cloud may be free, subscription-based, or provided on a pay-per-use model.

The **private cloud** infrastructure provides resources and services for the exclusive use of a specific public. Many organizations do not adopt public clouds because they are accessed over the Internet by the general public. A private cloud offers a higher degree of privacy and control over cloud services, resources, and data. Private clouds may be hosted on-premise or externally.

The **community clouds** infrastructure provides resources and services for exclusive use by a specific community of consumers, e.g., mission, security requirements, and policy. This cloud model supports a specific community that aims at a common goal. Community clouds may be owned, managed, and operated by one or more organizations, a third party, or some combination of them. Similar to the private clouds, community clouds may be hosted on-premise or externally.

Finally, **hybrid clouds** are composed of two or more distinct infrastructures (private, community, or public). As a result of this combination, each hybrid cloud has different properties, such as performance, cost, and security. Table 3.1 presents a comparison of the different models.

Table 3.1 Comparison of the four cloud distribution models

Type	Consumer	Access	Provider	Location
Public	General public	Shared	External	Externally
Private	Specific public	Private	Internal or External	On-premise or externally
Community	Specific community	Shared	Internal or External	On-premise or externally
Hybrid	Hybrid	Hybrid	Hybrid	Hybrid

3.3 CHALLENGES

Although cloud architecture appears as an attractive solution for companies worldwide, various challenges remain inadequately addressed (ARMBRUST et al., 2010; DILLON; WU; CHANG, 2010; ZHANG; CHENG; BOUTABA, 2010; KHORSHED; ALI; WASIMI, 2012; AN; ZAABA; SAMSUDIN, 2016; OPARA-MARTINS; SAHANDI; TIAN, 2016; NARANG; GUPTA, 2018):

1. **Security**: Since Cloud Computing available services over the Internet, access authentication, confidentiality, and auditability are examples of important requirements.

2. **Availability**: Cloud providers should allow consumers to access and to utilize the services anytime and anywhere.

3. **Auto scaling**: Resources and services (physical and logical) should be (re)configured automatically whenever possible since Cloud Computing is an autonomous system managed transparently.

4. **Data management**: Many challenges are associated with data management in cloud applications. For instance, services should ensure data integrity, backup, availability, or completeness. These characteristics are critical factors for cloud success and depend on the service. Data processing must guarantee service scalability without compromising data consistency.

5. **Financial cost**: Clouds should fulfill business rules without overloading consumers' budgets. The cost reduction happens as cloud providers rapidly provisioned and released services on demand.

6. **Standardization**: One of the leading Cloud Computing challenges is to perform transparent communication among services. This communication is a challenge due to the high number of technologies involved. Providers develop, manage, and provide clouds using different technologies without any standardization. The lack of standardization causes a vendor lock-in problem, and vendor lock-in problem hinders interoperability and portability among systems (LOUTAS et al., 2011).

3.4 SERVICE MODEL

The most common service model in the literature for Cloud Computing is formed by three levels: IaaS, PaaS, and SaaS. Figure 3.1 illustrates the relationship among the layers: system architects manage the IaaS infrastructure to provide a platform in PaaS. Developers utilize the PaaS platform to provide services for SaaS. Finally, SaaS level provides services to end-users (RINGS; GRABOWSKI, 2012; TARIQ; KHAN; IFTIKHAR, 2014). Each layer employs and provides a set of physical or logical resources as services.

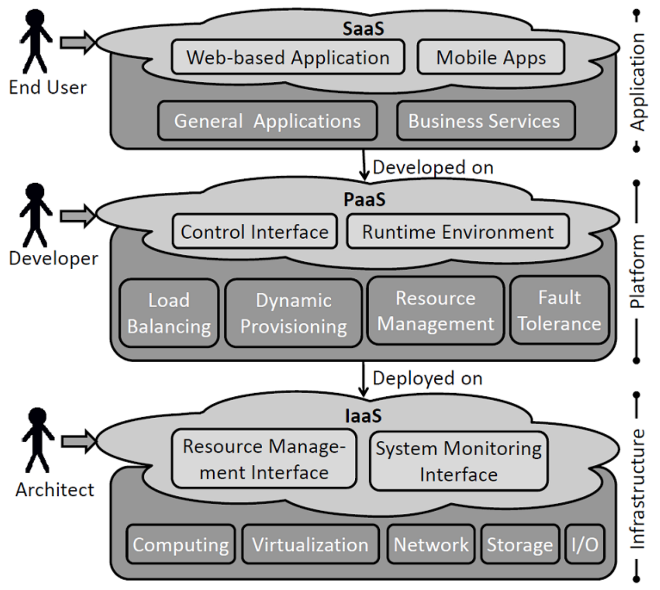


Figure 3.1 The most common service model in the literature for Cloud Computing (RINGS; GRABOWSKI, 2012).

Some cloud providers create other cloud services. These novel services were incorporated into the three layers model to fulfill more specific business rules. For instance, DBaaS (MALLIGA, 2012), DaaS (DIKAIAKOS et al., 2009), Communication as a Service (CaaS) (SCHAFFER, 2009), Confidentiality as a Service (FAHL et al., 2012), Privacy as a Service (PaaS or DPaaS) (SCHAFFER, 2009), and Test as a Service (TaaS) (TUNG; LIN; SHAN, 2014) are examples of these novel services. Those models are commonly defined as *Everything as a Service* (XaaS). This nomenclature indicates that any computational resource can be created and provided as a service (RIMAL; CHOI; LUMB, 2009; SIMMON, 2018). The next subsections detail some cloud services.

3.4.1 Software as a Service (SaaS)

SaaS is a delivery model of applications in a cloud environment (BUYYA; BROBERG; GOSCINSKI, 2011). Cloud environment enables multiple devices to access these applications (SaaS) through a thin client[1]. The consumer does not manage or control the cloud infrastructure (e.g., network, servers, operating systems, or storage). The consumers can only configure user-specific settings (LIU et al., 2011).

Examples of SaaS are Dropbox[2] and Google Docs[3]. Both applications are available over the Internet through a web browser or through an API that can be accessed by their respective mobile and desktop clients or by third-party applications.

3.4.2 Platform as a Service (PaaS)

PaaS provides a platform to develop, manage, and host applications. PaaS platform provides databases, development environments, operating systems, among others, to facilitate application development. This layer offers developers a transparent environment without quantifying the total of processors or memory that a given application may require (MELL; GRANCE, 2011; BUYYA; BROBERG; GOSCINSKI, 2011).

Heroku[4] and Google Colab[5] are examples of PaaS services.

3.4.3 Infrastructure as a Service (IaaS)

IaaS affords virtualized resources, such as virtualization, storage, and network on-demand. This infrastructure provides heterogeneous servers transparently and facilitates the organizations' expansion based on infrastructure on-demand (MELL; GRANCE, 2011; BUYYA; BROBERG; GOSCINSKI, 2011).

Examples of IaaS are VMware[6] and Amazon Web Services[7].

[1]In a client-server architecture, a thin client is a simple and low-performance computer optimized for a remote connection. The client depends mostly on the server for access and processing of resources.

[2]<https://dropbox.com>

[3]<https://docs.google.com>

[4]<https://www.heroku.com>

[5]<https://colab.research.google.com/>

[6]<https://www.vmware.com>

[7]<https://aws.amazon.com>

3.4.4 Data as a Service (DaaS)

DaaS is a data delivery and management model on-demand regardless of the geographic or organizational location of provider and consumer (TRUONG; DUSTDAR, 2009).

As previously stated, data may be stored in non-structured (e.g., text), semi-structured (e.g., XML, JSON, and CSV), or structured format (e.g., Relational Database). Additionally, different cloud providers employ different modeling and technologies to provide the same resources (LOUTAS et al., 2011). For these reasons, Cloud Computing manages and processes heterogeneous DaaS (TERZO et al., 2013).

DaaS services are dynamic since they must fulfill requests from different consumers. Terzo et al. (2013) state that a DaaS can be separated into three layers: (i) discovery layer finds or collects data from heterogeneous sources, (ii) process layer concerns the processing of massive datasets, and (iii) the store layer regards the issues posed by Big Data, such as the problem of the store and manage a large amount of data.

Brazilian Open Data Portal[8] and DATA.GOV[9] bring together some DaaS.

3.4.5 Database as a Service (DBaaS)

DBaaS provides databases as a service to consumers or organizations on-demand. DBaaS service is scalable with some traditional model features, such as replication, data integrity, and data authentication (HACIGUMUS; IYER; MEHROTRA, 2002; MALLIGA, 2012). Although confusing, DaaS and DBaaS are different concepts (ZHENG; ZHU; LYU, 2013).

Some benefits of the DBaaS model compared to the traditional model are: (i) on-demand *scalability* allows simple increasing the computational resources when necessary as opposed to on-site databases; (ii) *rapid provisioning*; (iii) *enhanced security* with the remote monitoring and database being off-site; (iv) *outsourcing* the administration enables different services (e.g., backup, optimization, and upgrading) managed by experts; (v) *cost savings* since the DBaaS service is cheaper than a commercial DBMS; and (vi) easy *tracking* of features as usage time, space, availability, and resource consumption (HACIGUMUS; IYER; MEHROTRA, 2002; MALLIGA, 2012; SEIBOLD; KEMPER, 2012).

PaaS and DBaaS are similar due to the contracting model. In the PaaS model, the consumer purchases an environment with a Database Management System (DBMS) and all necessary computational resources, such as processor and memory (SEIBOLD; KEMPER, 2012). Similar to DBMS model, DBaaS service consists of a Relational, Not Only SQL (NoSQL), or NewSQL database.

Relational databases are composed of simple structures according to Atomicity, Consistency, Isolation, and Durability (ACID) properties. The standard language of relational databases is Structured Query Language (SQL). SQL queries allow the selection, insertion, updating, and deletion of data (ELMASRI; NAVATHE, 2010).

Although popular[10], relational databases do not fully fulfill modern application re-

[8]<http://dados.gov.br>

[9]<https://www.data.gov>

[10]<https://db-engines.com/en/ranking>, last accessed: November 6th, 2020

quirements, such as processing large volumes of data, scalability, and availability. NoSQL has emerged to fulfill some of these requirements (STOREY; SONG, 2017).

NoSQL is an approach to manage non-tabular and non-relational databases. Contrary to Relational models, NoSQL databases use flexible schemes without supporting ACID properties. NoSQL databases are based on Consistency, Availability, and Partition (CAP) Theorem. According to CAP Theorem, a NoSQL (or Relational) system can support only two of these proprieties simultaneously and not much more (HAN et al., 2011). Figure 3.2 presents NoSQL databases according to the two guarantees of the CAP Theorem: CA, CP, or AP. These databases are classified into Key-value, Column-oriented, Document, and Graph model.

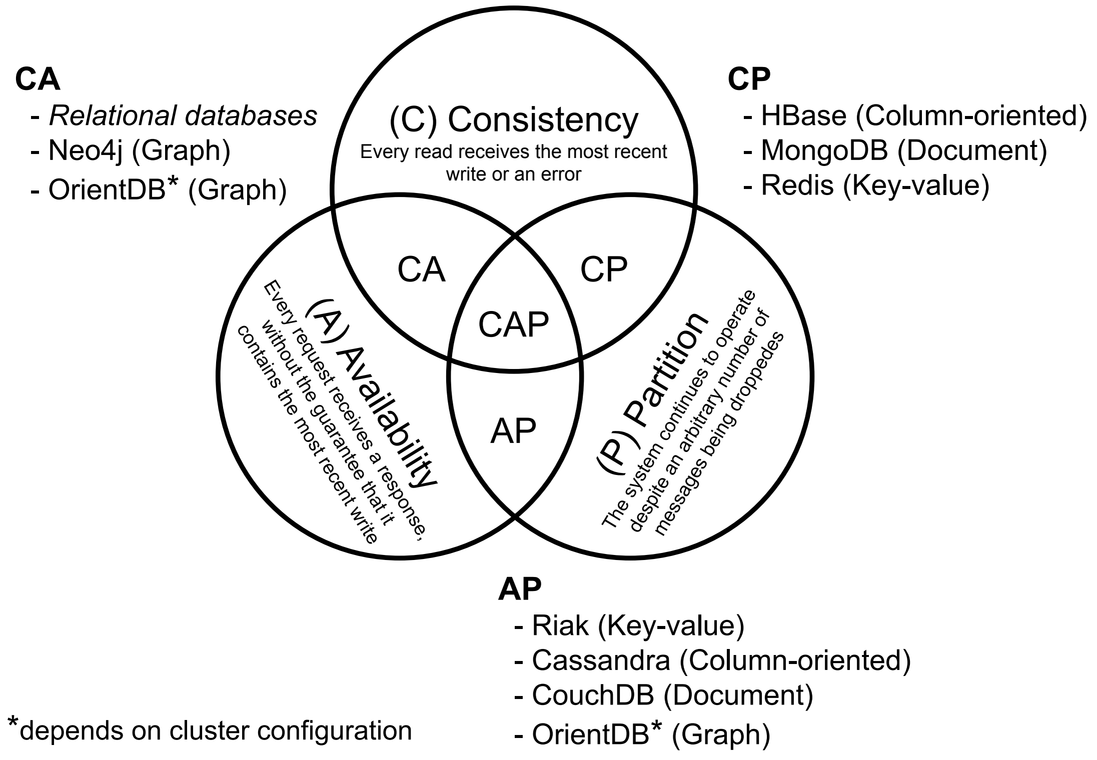

CA
- Relational databases
- Neo4j (Graph)
- OrientDB* (Graph)
(C) Consistency
Every read receives the most recent write or an error
CP
- HBase (Column-oriented)
- MongoDB (Document)
- Redis (Key-value)
CA
CP
CAP
AP
(A) Availability
Every request receives a response, without the guarantee that it contains the most recent write
(P) Partition
The system continues to operate despite an arbitrary number of messages being droppedes
AP
- Riak (Key-value)
- Cassandra (Column-oriented)
- CouchDB (Document)
- OrientDB* (Graph)
*depends on cluster configuration

Figure 3.2 NoSQL databases based on CAP Theorem. Adapted from (WANG; TANG, 2012; MONIRUZZAMAN; HOSSAIN, 2013).

In the key-value model, data is stored without schema, and each data consists of an indexed key and a value. The column-oriented model is an extension of the key-value model. This model enables complex structures with nested lists. The document model stores a set of key-values with unique keys in JSON format. Graph-based databases are based on graph theory and are useful for interconnected relationship data (HAN et al., 2011; FERNANDES; BERNARDINO, 2018). Figure 3.3 presents the same dataset in four different models.

NoSQL model uses a proprietary API to queries data on the database. For instance, given the same dataset, the following three queries return the same data:

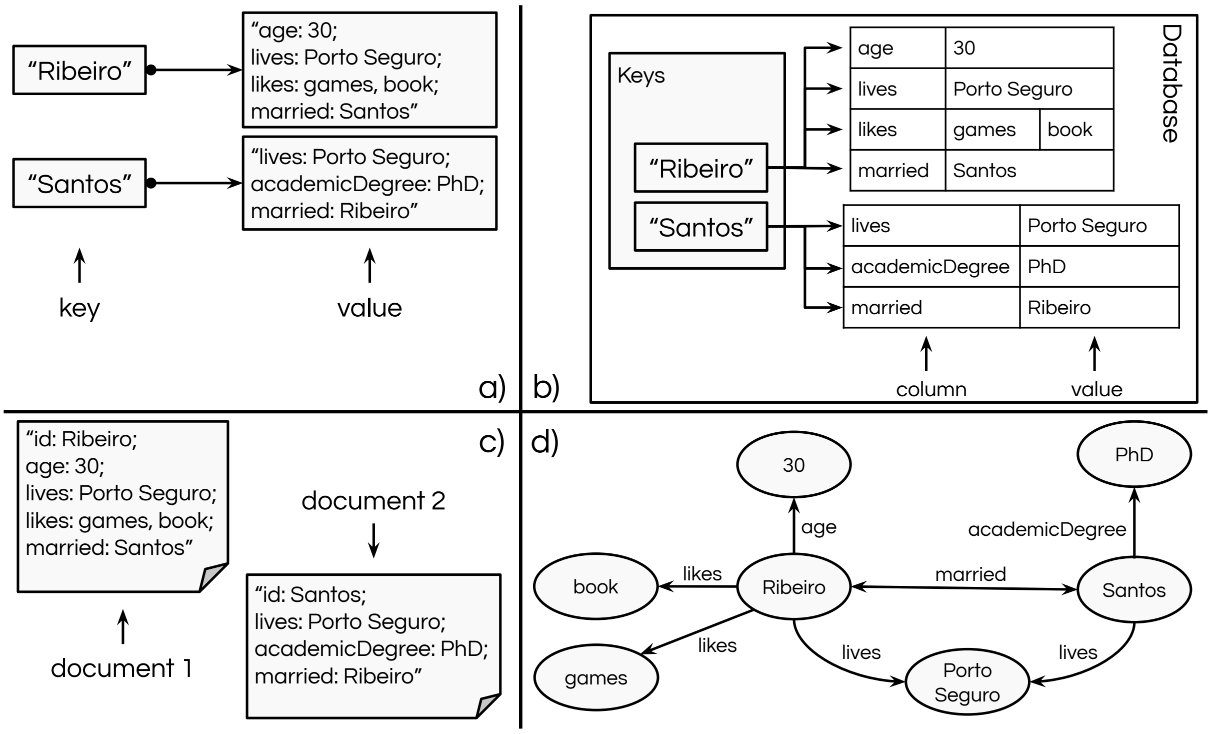


Figure 3.3 NoSQL models: (a) Key-value, (b) Column-oriented, (c) Document, and (d) Graph.

- **MySQL** (Relational database):

```
SELECT id
FROM people
WHERE married = "Santos"
```

- **MongoDB**:

```
db.people.aggregate([
    { $match:{married: "Santos"} },
    { $project:{"id":1} }
])
```

- **Neo4j**:

```
MATCH (t:people)
WHERE t.married = 'Santos'
RETURN t
```

Many organizations continue to use traditional relational databases due to uncertainties about NoSQL. The major impediments to companies adopt NoSQL solutions are the lack of full ACID transaction support and the absence of SQL. NewSQL databases have emerged to the strengths of the relational and NoSQL models. NewSQL is a relational and highly scalable database with support for ACID properties (GROLINGER et al., 2013).

Table 3.2 compares the Relational, NoSQL, and NewSQL databases (GROLINGER et al., 2013; FATIMA; WASNIK, 2016; MALHOTRA et al., 2017). We evaluate according to the following criteria: (i) *data model*, (ii) use of *keys*, (iii) availability for Big Data applications *(Big Data apps)*, (iv) query *language*, (v) *ACID* properties, (vi) *atomicity*, (vii) *consistency*, (viii) *isolation*, (ix) *durability*, (x) *CAP* properties, (xi) possibility to *partition*, and (xii) *scheme* flexibility.

Table 3.2 Comparison of Relational, NoSQL, and NewSQL databases (GROLINGER et al., 2013; FATIMA; WASNIK, 2016; MALHOTRA et al., 2017)

Criteria	Relational	NoSQL	NewSQL
Data model	Relational	Depends on database	Relational
Keys	Yes	No	Yes
Big Data apps	No	Yes	Yes
Language	SQL	Specific API	SQL
ACID	Yes	No	Yes
Atomicity	Yes	Depends on database	Yes
Consistency	Yes	Eventual	Yes
Isolation	Yes	Manual	Yes
Durability	Yes	Depends on database	Yes
CAP	CA	AP or CP	AP
Partition	Weak	Strong	Strong
Scheme	Hard	Light	Hard

Oracle[11] and MySQL[12] are example of Relational databases, MongoDB[13] and Redis[14] are example of NoSQL databases, and MariaDB[15] and VoltDB[16] are example of NewSQL databases.

[11]<https://www.oracle.com>
[12]<https://www.mysql.com>
[13]<https://www.mongodb.com>
[14]<https://redis.io>
[15]<https://mariadb.org>
[16]<https://www.voltdb.com>

3.5 THE LOCK-IN PROBLEM AND INTEROPERABILITY IN CLOUD COMPUTING

Cloud Computing provides a business strategy for small, medium, or large companies to remain competitive and meet consumer needs (ARMBRUST et al., 2010). Security and vendor lock-in are significant challenges for the growth and adoption of Cloud Computing (DILLON; WU; CHANG, 2010; SAHANDI; ALKHALIL; OPARA-MARTINS, 2013; LOUTAS et al., 2011).

Vendor lock-in problem in cloud computing occurs when cloud customers are dependent on a specific cloud provider (OPARA-MARTINS; SAHANDI; TIAN, 2016). Vendor lock-in problems may occur within a single cloud (i.e., vertical heterogeneity) or among distinct clouds (i.e., horizontal heterogeneity) (RANABAHU; SHETH, 2010).

Cloud users hardly interoperate their services with a different vendor due to substantial costs, legal constraints, or technical incompatibilities (ARMBRUST et al., 2010; RODERO-MERINO et al., 2010; OPARA-MARTINS; SAHANDI; TIAN, 2016). The cost to interoperate clouds can be substantial since different clouds often use proprietary protocols and interfaces. Communication between different services requires cloud providers to interact with other providers using different API (DILLON; WU; CHANG, 2010).

Interoperability in Cloud Computing can be classified into four categories (ARUNKUMAR; VENKATARAMAN., 2015):

- **Application** interoperability is at risk when a service or resource needs to be accessed by platform or provider, and this service or resource is bound to provider-specific API.

- **Platform** interoperability is at risk when the platforms on which services are running are not able to be moved to another cloud.

- **Storage** interoperability is at risk when specific RAID configurations and partitioning made by a cloud provider prevents data transference to another cloud provider.

- **Management** interoperability is at risk when details of cloud configuration (e.g., network adapters and open ports) may not be replicated at another cloud provider.

The importance of interoperability is related to Software Engineering from another perspective. According to Rana, Dauren and Kumaran (2015), the first phase in developing applications for Cloud Computing is to identify the structural model's specific characteristics. For instance, access authentication is an optional requirement in a public cloud and a mandatory requirement in a private cloud.

Tariq, Khan and Iftikhar (2014) state that these requirements are fundamental to successful cloud adoption. Figure 3.4 summarizes some cloud requirements from the perspective of providers, enterprises, and consumers. According to the figure, there are only three requirements concerning more than one stakeholder: quality of service, scalability, and interoperability (RIMAL et al., 2011).

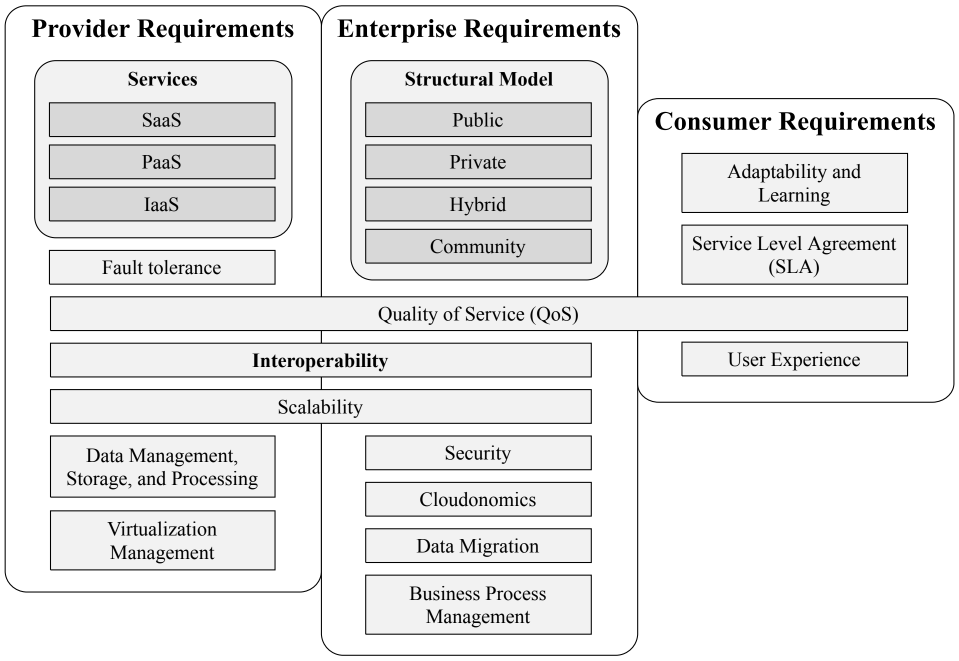


Figure 3.4 Relevant requirements in cloud applications (RIMAL et al., 2011).

3.6 CHAPTER SUMMARY

This chapter introduced the general concepts of Cloud Computing, their characteristics, challenges, deployment models, and service model. Cloud Computing provides physical and logical resources as a service on-demand in a pay-per-use policy. The cloud services are usually organized into three levels: IaaS, PaaS, and SaaS. However, other services may be incorporated into the three levels model to fulfill more specific requirements, such as DaaS and DBaaS. Five significant characteristics enable Cloud Computing to meet consumers' expectations: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The cloud environment provides services based on four deployment strategies: public, private, community, and hybrid models. This paradigm's most significant challenges are security, availability, auto scaling, data management, financial cost, and standardization.

The concepts presented in this chapter are the basis for understanding the thesis domain. The next chapter presents a MIDAS in detail.

Chapter

4

This chapter presents a Middleware for DaaS and SaaS (MIDAS). We describe MIDAS in detail because this solution is used as a basis for our model evaluation. We compare the different MIDAS versions and provide a MIDAS timeline.

MIDDLEWARE FOR INTEROPERABILITY AMONG CLOUDS

MIDAS is a solution to provide interoperability among different SaaS, DaaS, and DBaaS. Proposed initially by Marinho et al. (2016), some works have improved MIDAS middleware over time (VIEIRA et al., 2017; RIBEIRO et al., 2018; SILVA; RIBEIRO; CLARO, 2018; RIBEIRO et al., 2019; MANE et al., 2020). We present MIDAS in detail because the evaluation of our models is based on this middleware.

MIDAS is based on open-source technologies with PHP support. The middleware is hosted by Heroku, an open cloud that provides a complete PaaS, sufficient storage space, and some programming languages (e.g., PHP, Java, Ruby, Node, Python, and Go).

The next sections introduce MIDAS in detail, from the initial version (MIDAS 1.0) to the current version (MIDAS 2.0). We describe the improvements provided by each version.

4.1 MIDAS 1.0

MIDAS 1.0 (see Figure 4.1) is formed by four components: (i) *query decomposer*, (ii) *query builder*, (iii) *Dataset Information Storage (DIS)*, and (iv) *result formatter* (MARINHO et al., 2016). *Query decomposer* breaks the original query submitted by SaaS (in this version, only SQL) into an independent format. This format is more interoperable than the SQL query because it enables *query builder* to build the DaaS request[1] independently of the query language.

After receiving the query decomposed by *query decomposer*, the *query builder* assembles the Uniform Resource Locator (URL) to query DaaS. For this, *query builder* searches in *DIS* the information about DaaS requested.

[1]Each DaaS is accessed by a URL. The URL parameters vary according to the API.

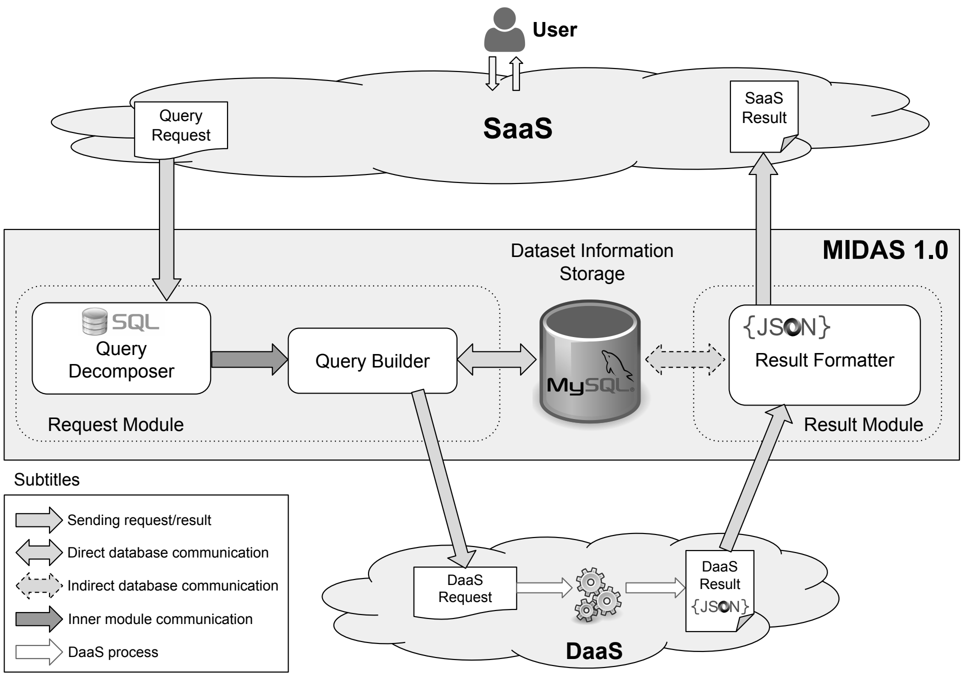


Figure 4.1 MIDAS 1.0 architecture (MARINHO et al., 2016).

DIS stores information about data and DaaS API. MIDAS 1.0 used a relational database to manage *DIS* information. These data were added and updated manually.

After performing the query on DaaS, *result formatter* handles the returned data. This component sends to SaaS only relevant information. MIDAS 1.0 received (from DaaS) and returned (to SaaS) data only in JSON format.

This release recognized SQL queries from a single DaaS.

4.2 MIDAS 1.6

MIDAS 1.6 (see Figure 4.2) is an enhanced version of MIDAS 1.0 (VIEIRA et al., 2017). This version grouped the MIDAS 1.0 components into DIS and two modules: *Request* and *Result*. As the names suggest, the *Request Module* handles the incoming SaaS query, while the *Result Module* process the DaaS return. *Request Module* combines *query decomposer* and *query builder*, and *Result Module* covers the *result formatter*.

Besides rearrange the MIDAS 1.0 components, MIDAS 1.6 presented some significant improvements. MIDAS 1.6 enables (i) more robust support to SQL queries, including `join` statement and (ii) simple MongoDB (NoSQL) queries. Additionally, this release (iii) recognizes returned data by DaaS in other formats (e.g., XML and CSV) in addition to JSON files. Finally, (iv) authors changed the DIS from a relational database (MySQL) to a JSON file. This improves the MIDAS efficiency and avoids lock-in problems in the future.

MIDAS 1.6 addresses important issues left open in the first version. The authors

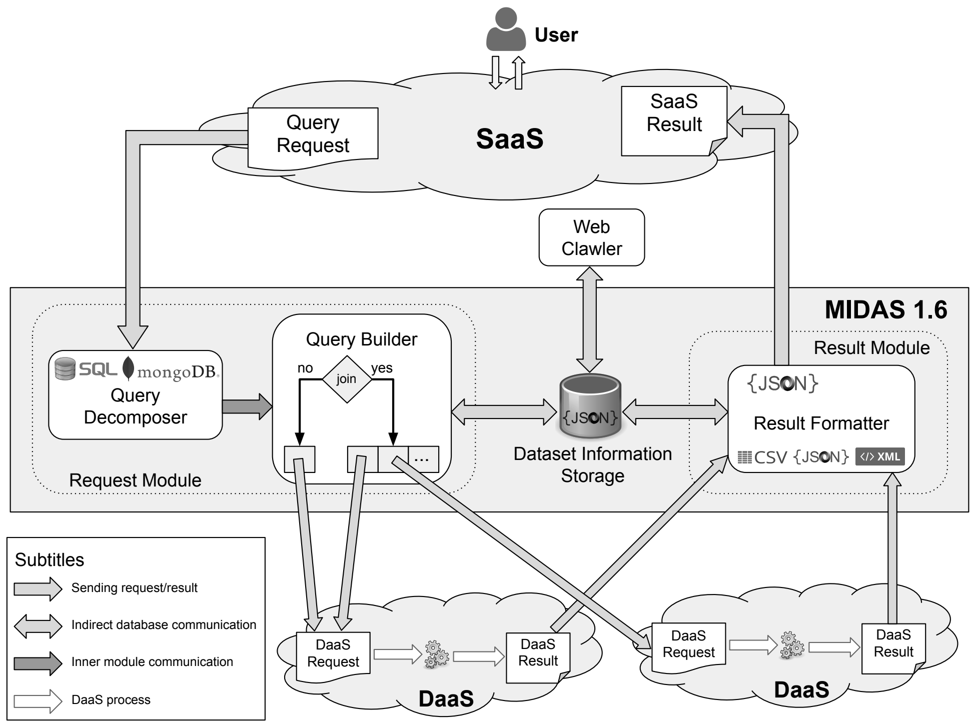


Figure 4.2 MIDAS 1.6 architecture (VIEIRA et al., 2017).

claim that results show the effectiveness of new MIDAS for tackling interoperability issues among Cloud Computing services. Nonetheless, a critical problem is presented: the absence of a crawler to automate the search for DIS information. This problem is significant because changes in DaaS parameters are updated manually in MIDAS 1.6. A Web Crawler was incorporated into MIDAS 1.6 to update DIS automatically (SILVA; RIBEIRO; CLARO, 2018). This Web Crawler enhances consistencies MIDAS 1.6 communication among cloud services.

4.3 MIDAS 1.8

MIDAS 1.8 (see Figure 4.3) recognizes DBaaS as a data storage service (RIBEIRO et al., 2018). In contrast to previous versions, MIDAS 1.8 enables (i) queries to DBaaS service, (ii) to manipulate different data (DaaS and/or DBaaS) into a single query, (iii) aggregation operation (`lookup`[2] clause) in MongoDB queries, and (iv) to return data to SaaS in other formats (e.g., XML and CSV) in addition to JSON files.

Queries in DBaaS are possible due to a new module: *Resource Module*. This module contains one component: *data mapping*. When the query sent by SaaS has a DBaaS as a data source, *data mapping* is activated, and it simulates a DBaaS as a DaaS. This feature allows query DBaaS without changing MIDAS components.

MIDAS 1.8 presents a formal description of the middleware components. The formal

[2]MongoDB `lookup` is similitar to SQL `LEFT OUTER JOIN`.

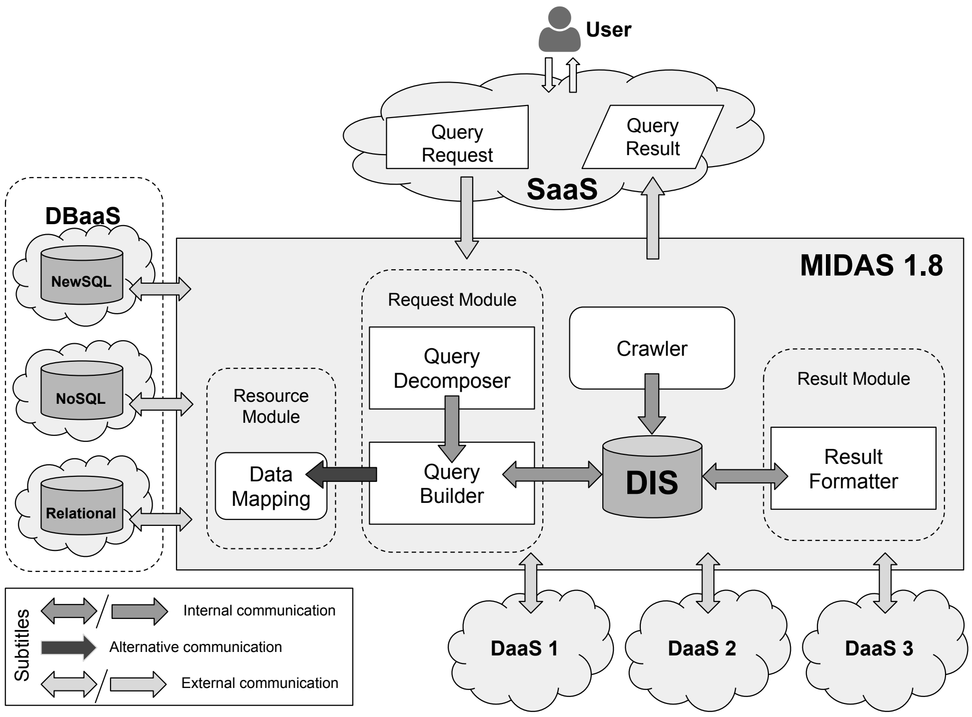


Figure 4.3 MIDAS 1.8 architecture (RIBEIRO et al., 2018).

model aims to explain and to document the communication among MIDAS components.

4.4 MIDAS 1.9

MIDAS 1.9 (see Figure 4.4) is an attempt to move towards concerning the formalization of communication (RIBEIRO et al., 2019). This version presents the formal model in Description Logic (DL).

Additionally, MIDAS 1.9 presented some minor adjustments. *Resource Module* was renamed to *Data Module*. Opposite to the previous version, the *Data Module* consists of two components: *data mapping* and *data join*. *Data mapping* identifies, obtains, and simulates a DBaaS as a DaaS. *Data join* runs when SaaS submits queries with the clause join. *Data join* receives data from *data mapping* separately, and then it performs the data merger. This module was split to improve the understanding of the components. *Resource Module* (MIDAS 1.8) and *Data Module* (1.9) have the same functionality.

About *Result Module*, *result formatter* was also split into two other components: *formatter* and *filtering*. *Formatter* is responsible for data formatting, while *filtering* performs the data association and selection. *Result Module* was split to detail the data flow.

The other modules (*DIS*, *Web Crawler*, and *Request Module*) remained unchanged.

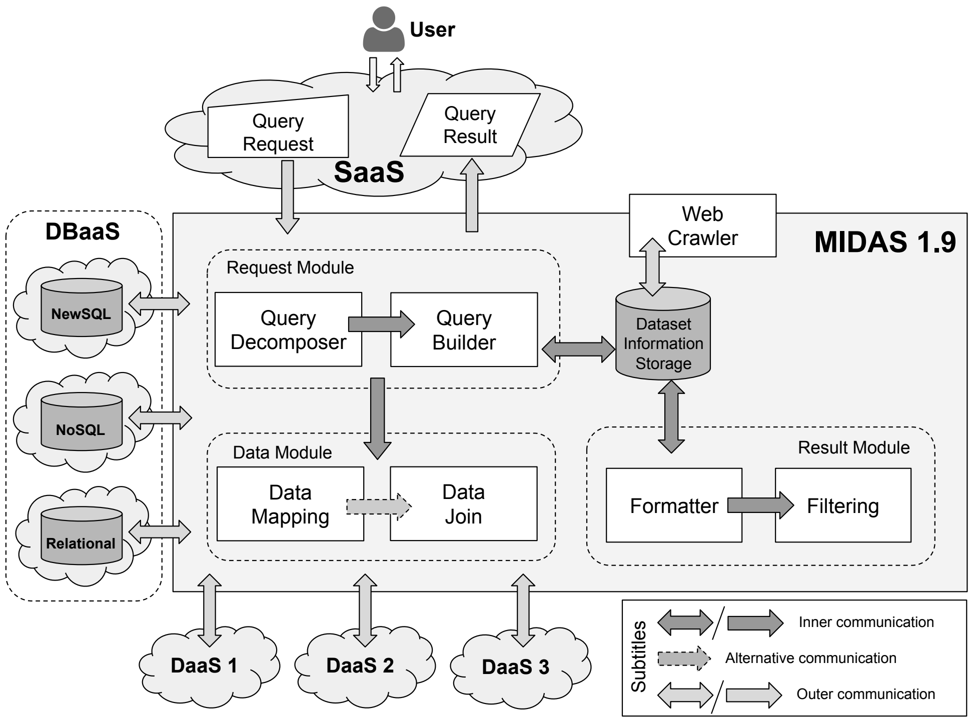


Figure 4.4 MIDAS 1.9 architecture (RIBEIRO et al., 2019).

4.5 MIDAS 2.0

MIDAS 2.0 (see Figure 4.5) is the first attempt to incorporate a semantic approach into MIDAS middleware (MANE et al., 2020). The semantic approach aims to maximize MIDAS availability even if DaaS parameters changed over time.

The authors developed a method to semantically combine parameters from DaaS and data from the original query. The similarity method is based on several knowledge sources. This approach identifies a conceptual proximity degree between two parameters based on Cosine similarity, Jaccard index, and WordNet.

A semantic approach is possible due to a new module: *Semantic Module*. This new module is composed of two components: *semantic mapping* and *Semantic Mapping Storage (SMS)*. *Semantic mapping* accesses *Web Crawler* logs after each *DIS* update to identify semantic similarity among DaaS parameters. *SMS* stores the *semantic mapping* results results in a tree structure. The parameter in each result points to its semantically similar word.

Although the authors pointed out that results present some overhead due to the middleware structure, experiments show that MIDAS 2.0 is a good attempt towards semantic interoperability among cloud services.

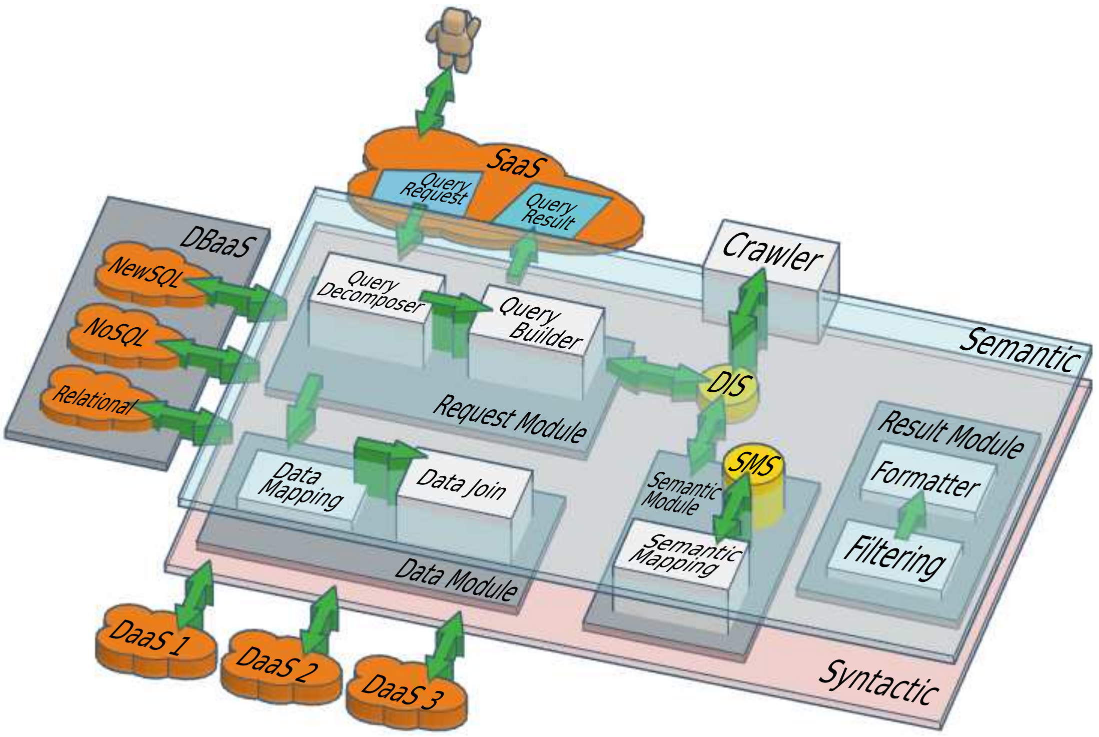


Figure 4.5 MIDAS 2.0 architecture (MANE et al., 2020).

4.6 CHAPTER SUMMARY

MIDAS is a middleware to provide interoperability among cloud services. In this chapter, we described the MIDAS and presented all five middleware versions published till today. We present the MIDAS evolution because the evaluation of this thesis is based on the middleware versions. This evolution emphasizes the contribution of this thesis in the middleware improvements. Figure 4.6 depicts the main similarities and differences among all MIDAS versions.

In Part III, we present our CAPITAL. The next chapter describes the model for syntactic interoperability. We also report model evaluation into MIDAS.

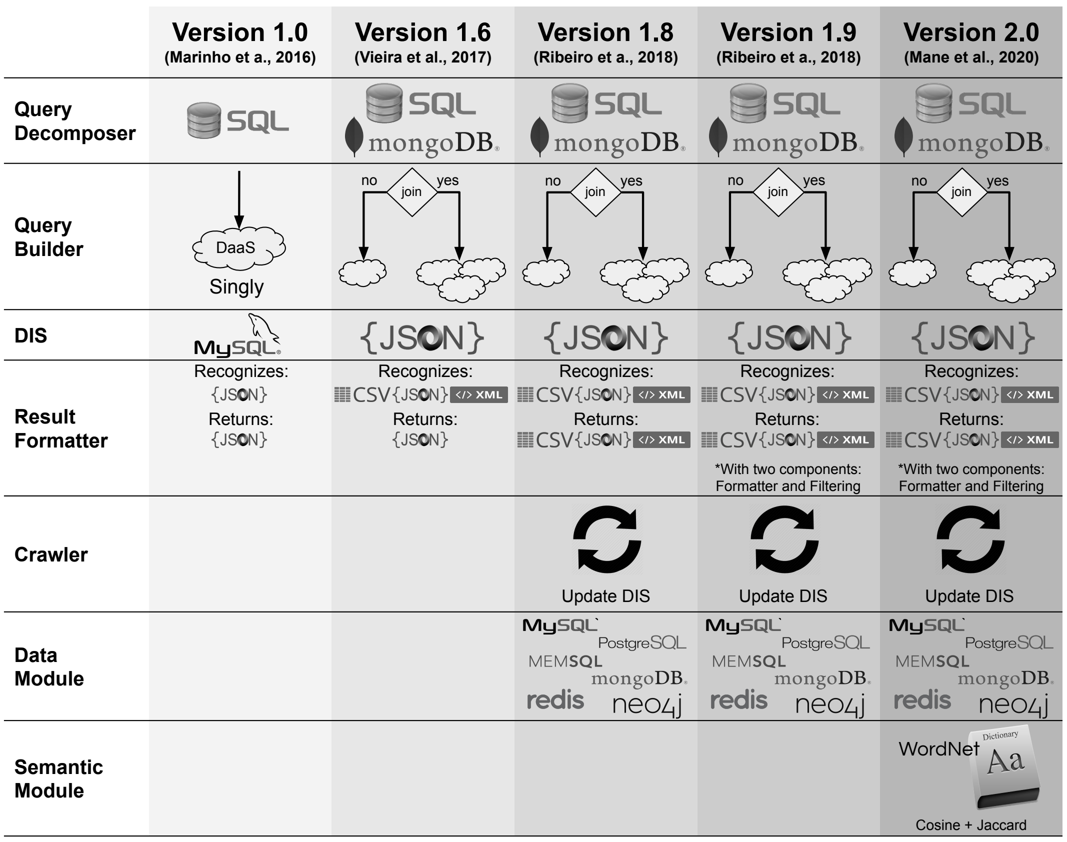


Figure 4.6 Similarities and differences among the MIDAS versions.

PART II

CAPITAL FRAMEWORK

Chapter

5

This chapter presents the model for syntactic interoperability. We detail the syntactic interoperability of MIDAS, and then we present the evaluation of this model. Finally, we present some related work.

MODEL FOR SYNTACTIC INTEROPERABILITY

The objective of this thesis is to investigate and provide a model for pragmatic interoperability among cloud services. Since pragmatic interoperability is achieved after the consolidation of syntactic and semantic interoperability (ASUNCION; SINDEREN, 2010), this thesis describes the syntactic and semantic levels before presenting the pragmatic interoperability. Our models consider horizontal heterogeneity (i.e., among distinct clouds) and application interoperability (i.e., when a service needs to be accessed by another service).

Our model for syntactic interoperability is based on MIDAS 1.8 (RIBEIRO et al., 2018) for chronological reasons. We employ a bottom-up strategy. This model aims to explain and detail the syntactic interoperability among SaaS, DaaS, and DBaaS.

Generically, we assumed that p is a solution (e.g., a middleware) to provide syntactic interoperability between cloud services s_1 and s_2. Figure 5.1 depicts the basic scenario for the syntactic interoperability model. Given a solution p to interoperate s_1 and s_2, we describe the syntactic interoperability based on three canonical models: (i) m_1 describes the flow $s_1 \rightarrow p \rightarrow s_2$, (ii) m_2 describes the flow $s_2 \rightarrow p \rightarrow s_1$, and (iii) m_3 describes p. Each canonical model m_i includes a set of mathematical definitions.

The p solution may be generalized to interoperate s_1 with more than one service, i.e., s_2, s_3, …, s_n. For instance, we can assume that our solution interoperates a SaaS with multiple DaaS (DaaS$_1$, DaaS$_2$, …, DaaS$_n$). In this case, two situations may occur: SaaS communicates with a single or multiple DaaS, i.e., the communication occurs with or without data join. Regardless of the situation, the models m_1, m_2, and m_3 store and manipulate information necessary to provide syntactic interoperability, such as projection, selection, data sources, join condition, among others. When the solution requires to interoperate SaaS and multiple DaaS, m_1 describes the flow SaaS $\rightarrow p \rightarrow$ DaaS$_1$, DaaS$_2$, …, DaaS$_n$; m_2 describes the flow DaaS$_1$, DaaS$_2$, …, DaaS$_n \rightarrow p \rightarrow$ SaaS; and m_3 describes information about each DaaS, such as attributes, formats, domain, among others.

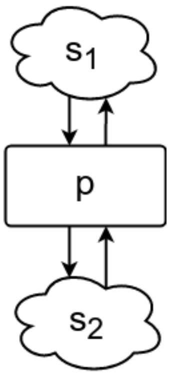


Figure 5.1 Generic solution p to interoperate s_1 and s_2 services.

The next section details our model for syntactic interoperability (RIBEIRO et al., 2018). After that, we present the evaluation of this model. Finally, we present some related work.

5.1 MODEL

Definition 5.1 (MIDAS internal structure)**.** *The structure used internally by MIDAS (MIDASql) is a tuple MIDASql* $= (mDIS, mSaaS, mDaaS)$*, where: mDIS is the canonical model of DaaS presented in DIS; mSaaS is the canonical model that maps the query sent by SaaS (i.e., the flow SaaS* $\rightarrow$ *MIDAS* $\rightarrow$ *DaaS*$_1$*, DaaS*$_2$*, …, DaaS*$_n$*); and mDaaS is the canonical model that maps DaaS return (i.e., the flow DaaS*$_1$*, DaaS*$_2$*, …, DaaS*$_n$ $\rightarrow$ *MIDAS* $\rightarrow$ *SaaS).*

We name m_1 as *mSaaS*, m_2 as *mDaaS*, and m_3 as *mDIS*. The following subsections detail each canonical model.

5.1.1 Canonical Model mDIS

Definition 5.2 (mDIS)**.** *The canonical model mDIS stores information present in the DIS. This model is a tuple mDIS* $= (N_{root}, DAAS)$*, where:* N_{root} *is the name of the model; and* $DAAS$ *is a set of DaaS models (daas set).*

Definition 5.3 (daas)**.** *The canonical model for a specific DaaS (daas* $\in DAAS$*) is a tuple daas* $= (N_{root_{daas}}, K)$*, where:* $N_{root_{daas}}$ *is the name of DaaS; and* K *is a predefined set of keys (k) for each DaaS, where K = {domain, search_path, query, sort, limit, dataset, records, fields, format}.*

Definition 5.4 (k)**.** *A key* k *(*$k \in K$*) is an information about* $daas$*. Each* k *is defined as* $k = (N_{root_k}, i)$*, where:* N_{root_k} *specifies information* k *(*$N_{root_k} \in K$*); and* i *is the information about* k*. Information* i *may be empty, atomic, or multivalued.*

5.1.2 Canonical Model mSaaS

Definition 5.5 (mSaaS). *The canonical model mSaaS converts the query submitted by SaaS in a set with n URL, where n ($n \geq 1$) is the number of join in the query. For instance, $n = 2$ indicates a data join with 2 sources. Each DaaS is accessed by a URL. The model mSaaS is a tuple mSaaS $= (N_{root}, C_1)$, where: N_{root} is the value of n; and C_1 is a set of first-level clauses (c_1) that map and identify queries and operations.*

Definition 5.6 (c_1). *A first-level clause c_1 ($c_1 \in C_1$) stores specific information about a query OR about an operation. This clause is a tuple $c_1 = (N_{root_{c_1}}, C_2)$, where: $N_{root_{c_1}}$ identifies the query OR the operation; and C_2 is a set of second-level clauses (c_2) that map and identify the query attributes and operations. Some important observations: (i) $N_{root_{c_1}} \in \{q_1, q_2, \ldots, q_n, param\}$, where q_i is an i-th query of a join with n data sources and param stores data about join, order by and limit clauses; and (ii) given n, there are $n + 1$ clauses c_1.*

Definition 5.7 (c_2). *A second-level clause c_2 ($c_2 \in C_2$) stores information about query clauses (q_i) OR about operation clauses (param). This clause is a tuple $c_2 = (N_{root_{c_2}}, V)$, where: $N_{root_{c_2}}$ identifies the query clause OR operation clause; and V is a set of values (v) for each c_2. Some important observations: (i) if c_1 represents a query q_i, then $N_{root_{c_2}}$ concerns j attributes ($j \geq 0$) of q_i, where $N_{root_{c_2}} \in \{Projection, Selection, Dataset\}$; (ii) if c_1 represents param, then $N_{root_{c_2}}$ comprises j attributes ($j \geq 0$) of all n relations, where $N_{root_{c_2}} \in \{OrderBy, Limit, TypeJoin, CondJoin, Return\}$; and (iii) given n, there are $3n + 5$ clauses c_2.*

Definition 5.8 (v). *A value v ($v \in V$) represents an information about c_2. Depending on c_2, V may be empty, atomic, or multivalued, i.e, $V = \varnothing$ or $V = \{v_1, v_2, \ldots, v_w\}$, where: v_i is the i-th value v for each c_2; and w is the number of values v for key c_2.*

After generating *mDIS* and *mSaaS*, our model for syntactic interoperability needs to create a set of URLs to retrieve data from DaaS. *MIDASql* converts both *mDIS* and *mSaaS* into URLs according to the function `generateURLs`. This function has the following prototype: `URLs generateURLs(mDIS, mSaaS, n)`. Given *mDIS*, *mSaaS*, and n, `generateURLs` returns a set of `URLs`. Each URL is a concatenation sequence of *mDIS* and *mSaaS* elements. The function `generateURLs` creates n URLs ($n \geq 1$), where n is the number of joins in the query, i.e., each q_i (in *mSaaS*) generates URL$_i$. For this, we assume that (i) `+` is the concatenation operator for two literal or variable strings, and (ii) `ch(p)` is a function that returns the contents of the child(ren) of p node. Code in Listing 5.1 describes the function `generateURLs`.

Some important remarks about function `generateURLs`: (i) when `ch(p)` does not return elements, the corresponding code line for `URL`$_i$ must be disregarded; (ii) multivalued results for `ch(p)` are separated by commas; and (iii) if $n \geq 2$, then `ch(q`$_i$`.Projection)` must include `ch(param.CondJoin)` when the join attribute is not included in the projection attributes.

Listing 5.1 Function `URLs generateURLs(mDIS, mSaaS, n)`

```
URLs generateURLs(mDIS, mSaaS, n)
    for each q_i in mSaaS
        SName = mDIS.ch(q_i.dataset)
        URL_i =
                    ch(SName.domain)      +
                    ch(SName.search_path) + '?' +
                    ch(SName.dataset)     + '=' + ch(q_i.Dataset)
            + '&' + ch(SName.records)     + '=' + ch(q_i.Projection)
            + '&' + ch(SName.query)       + '=' + ch(q_i.Selection)
            if n > 1    // n is the number of join in the query
                URL_i = URL_i +
                    '&' + ch(SName.sort)  + '=' + ch(param.OrderBy) +
                    '&' + ch(SName.limit) + '=' + ch(param.Limit)
        return URL_i
```

5.1.3 Canonical Model mDaaS

Each DaaS returns data through a URL. Before sending the results to SaaS, MIDAS performs some operations to make the data "presentable", such as join, order by, and limit, when applicable. These steps are carried out by employing the canonical model *mDaaS*.

Definition 5.9 (mDaaS)**.** *The canonical model mDaaS maps the returns of n DaaS. DaaS sends a return for each URL in a format described in the mDIS. If $n = 1$, then mDaaS just converts the format returned by DaaS into format desired by SaaS, i.e., mDaaS just converts $ch(DIS.ch(q_1.dataset).format)$ into $ch(param.Return)$. When $n \geq 2$ (i.e., when there is a join), the relations are mapped two-by-two and mDaaS generates n canonical mappings. In this case ($n \geq 2$), mDaaS is a tuple $mDaaS = (N_{root}, CJ)$, where: N_{root} is the name of the DaaS model; and CJ is a distinct set of $ch(param.CondJoin)$ (cj) values in the corresponding relation. The name (N_{root}) of the i-th model is q_iD.*

Definition 5.10 (cj)**.** *An information cj ($cj \in CJ$) is a value that satisfies the join condition $ch(param.CondJoin)$ in the relation. Thus, cj is a tuple $cj = (N_{root_{cj}}, L)$, where: $N_{root_{cj}}$ is the name that identifies the value cj; and L is a set of lists (l) with all attributes that contain cj.*

Definition 5.11 (l)**.** *A list l ($l \in L$) contains all elements of the same tuple in which cj is part, in the same order that occurs in the relation: from left to right. The amount of $l \in L$ is equal to the amount of occurrences of cj in the relation, thus $l = \{a_1, a_2, a_3, \dots, a_m\}$, where: a_i is the i-th attribute a for each l in cj; and m is the number of attributes $a \in l$.*

The data join must be done after generating *mDaaS*. The next step depends on join type ($ch(param.TypeJoin)$). In addition to the functions and operator mentioned previously, we assume that: `lch(p)` is a function that returns the last child of a p node; and `con(p_1, p_2)` is a function that connects the node p_1 to the node p_2.

The joins are performed as follows:

1. When $ch(param.TypeJoin) =$ '*left outer*':

   (a) $\forall\, cj_1 \in ch(q_1D)$ and $\forall\, cj_2 \in ch(q_2D)$, $con(lch(q_1D.cj_1),\ ch(q_2D.cj_2))$, $\forall\, cj_1 = cj_2$;

   (b) if $cj_1 \notin ch(q_1.Projection)$, then (i) perform $con(q_1D,\ ch(q_1D.cj_1))$ and (ii) remove cj_1;

2. When $ch(param.TypeJoin) =$ '*right outer*':

   (a) $\forall\, cj_1 \in ch(q_1D)$ and $\forall\, cj_2 \in ch(q_2D)$, $con(lch(q_2D.cj_2),\ ch(q_1D.cj_1))$, $\forall\, cj_1 = cj_2$;

   (b) if $cj_2 \notin ch(q_2.Projection)$, then (i) perform $con(q_2D,\ ch(q_2D.cj_2))$ and (ii) remove cj_2;

3. When $ch(param.TypeJoin) =$ '*inner join*':

   (a) $\forall\, cj_1 \in ch(q_1D)$, if $cj_1 \notin ch(q_2D)$, then remove cj_1;

   (b) $\forall\, cj_2 \in ch(q_2D)$, if $cj_2 \notin ch(q_1D)$, then remove cj_2;

   (c) $\forall\, cj_1 \in ch(q_1D)$ and $\forall\, cj_2 \in ch(q_2D)$, perform $con(lch(q_1D.cj_1),\ ch(q_2D.cj_2))$, $\forall\, cj_1 = cj_2$ and $\forall\, cj_2 = cj_1$;

   (d) $\forall\, cj_1 \notin ch(q_1.Projection)$, (i) perform $con(q_1D,\ ch(q_1D.cj_1))$ and (ii) remove cj_1;

4. When $ch(param.TypeJoin) =$ '*full join*':

   (a) $\forall\, cj_1 \in ch(q_1D)$ and $\forall\, cj_2 \in ch(q_2D)$, $con(q_1D, q_2D.cj_2)$;

   (b) $\forall\, cj_1 \notin ch(q_1D.Projection)$, (i) perform $con(q_1D, ch(q_1D.cj_1))$ and (ii) remove cj_1;

5. Regardless of the join type, the following steps must be performed after the previous steps:

   (a) if $ch(param.OrderBy)$ node exists, then this node must be ordered;

   (b) if $ch(param.Limit)$ node exists, then this must be the total of $ch(q_1D)$;

   (c) q_1D should be converted into $ch(param.Return)$ format and the data is sent to SaaS.

The evaluation of the model for syntactic interoperability is presented in Section 5.2.

5.2 EVALUATION

We evaluated the model for syntactic interoperability with a proof of concept, and we performed our model based on two different queries: with and without data join. Figure 5.2 depicts a DIS with two DaaS: *w7* and *vz*. Both queries consider the same DIS. The main node (*DIS*) references two subtrees, and each subtree stores data about a specific DaaS. At the last level (i), each node stores information about the k level immediately above.

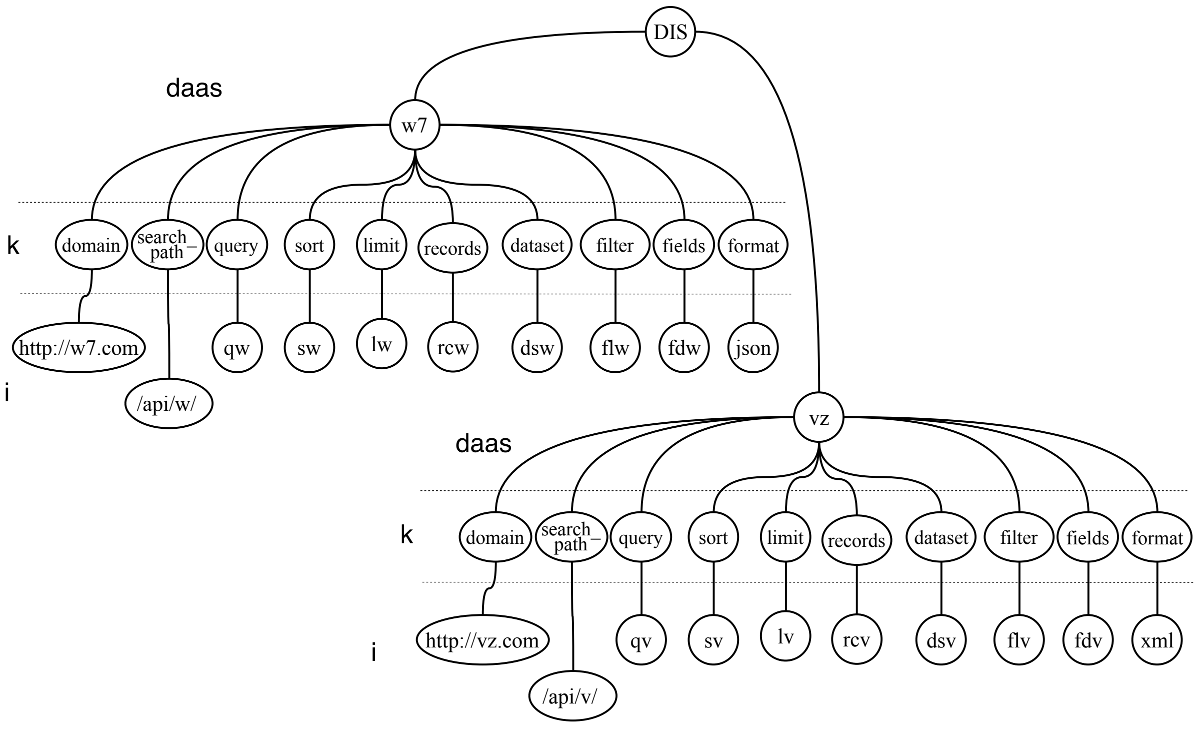


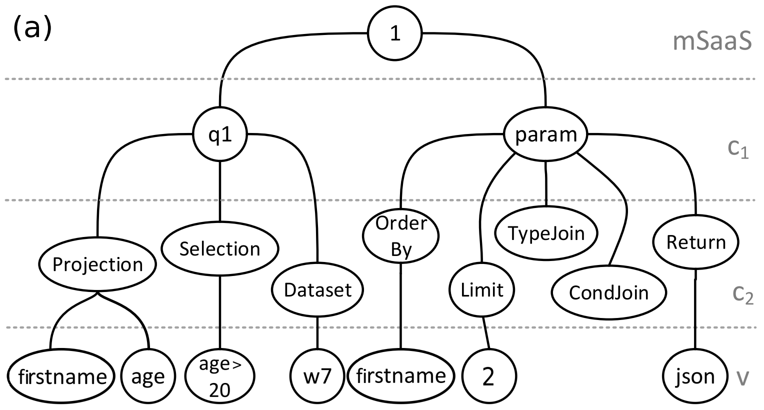
Figure 5.2 Example of *mDIS* with two DaaS: *w7* and *vz*.

Originally, *w7* stores data about New York City hospitals, and *vz* stores data on New York department of education borough enrollment offices. Table A.1 and Table A.2 in Appendix A illustrate the data considered in the evaluation of our model for syntactic interoperability. Queries are presented in the Subsection 5.2.1 (without data join) and Subsection 5.2.2 (with data join).

5.2.1 Query without data join

The query present in Figure 5.3(b) generates the canonical model *mSaaS* described in Figure 5.3(a). The query is described in SQL and NoSQL (MongoDB) without data join.

The main node of *mSaaS* in Figure 5.3(a) points to two subtrees: *q1* and *param*. The first subtree (*q1*) stores data about query: (i) projection attributes (*firstname* and *age*), (ii) selection condition (*age* > 20), and (iii) data source name (*w7*). The second subtree (*param*) stores data about (i) sort attribute (*firstname*), (ii) limit clause value



(b)

SQL	NoSQL (MongoDB)

`SELECT firstname,`
`   age`
`FROM w7`
`WHERE age > 20`
`ORDER BY firstname`
`LIMIT 2`
	`db.w7.find(`
	`   {'age>20'},`
	`   {'firstname':1, 'age':1})`
	`.limit(2)`
	`.sort({'firstname':1});`

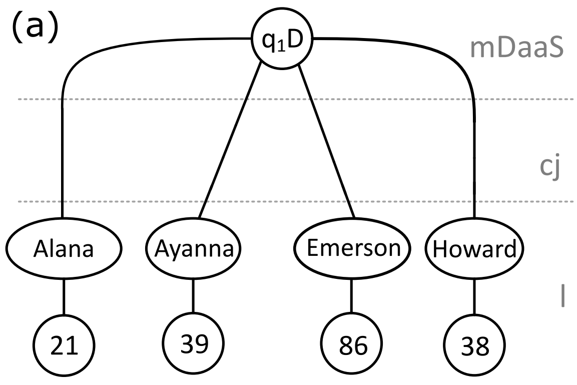
Figure 5.3 Example of *mSaaS* (a) for an SQL/NoSQL query without data join (b).

(*2*), and (iii) return format (*json*). Join type and condition are empty since the query (Figure 5.3(b)) have no data join. The values present in the *v* level are independent of the language used in the query.

DaaS return data through a URL. Given *mDIS* (Figure 5.2) and *mSaaS* (Figure 5.3(a)), the function `generateURLs` (Listing 5.1) generates the following URL:

```
URL1 = http://w7.com/api/w/?dsw=w7&rcw=firstname,age&qw=age>20&sw=
    firstname&lw=2
```

Considering that the query without data join (Figure 5.3(b)) returns the data present in Figure 5.4(b), *mDaaS* creates the canonical model in Figure 5.4(a).

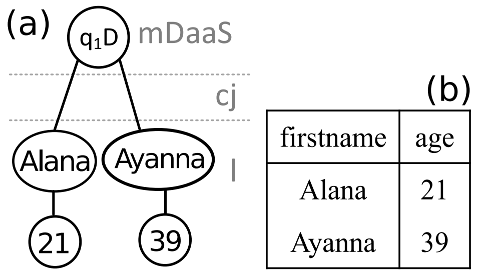


(b)

w7	
firstname	age
Alana	21
Ayanna	39
Emerson	86
Howard	38

Figure 5.4 Example of *mDaaS* (a) for data returned by query without join (b).

After MIDAS receives the return from a DaaS, middleware performs *limit* clause on the data before sending the results to SaaS. We assume that *mDaaS* (Figure 5.5(a)) represents the data that should be returned to the SaaS (Figure 5.5(b)).

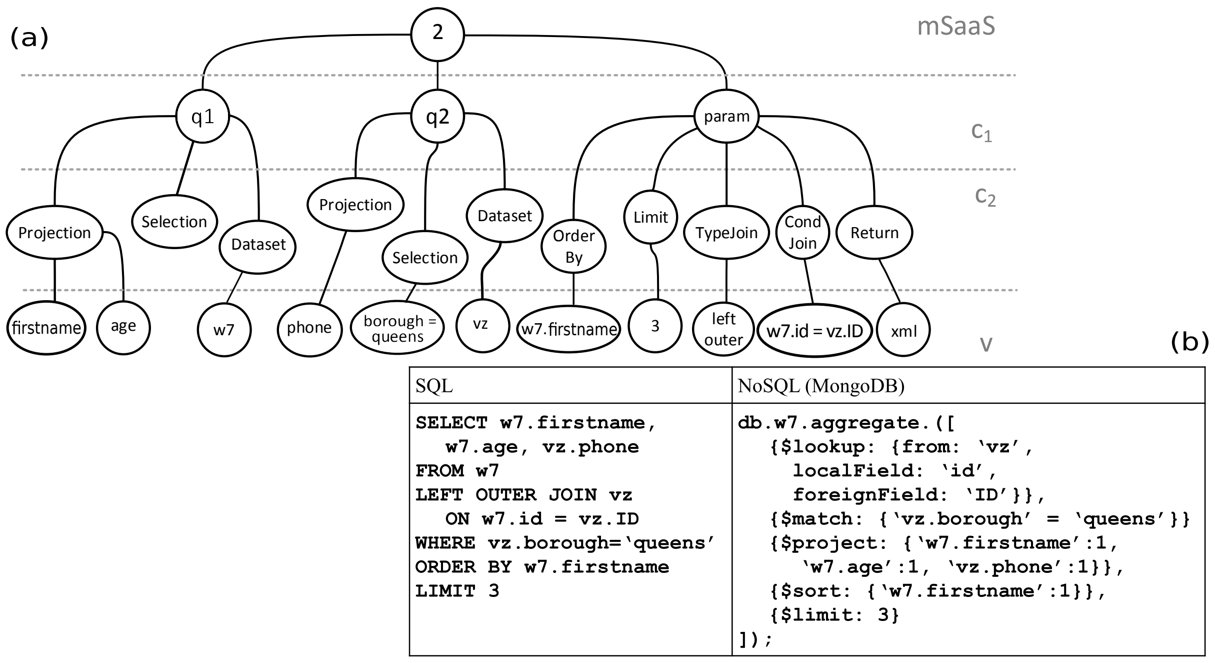


firstname	age
Alana	21
Ayanna	39

Figure 5.5 Final result after model performs *limit* clause for the query without data join.

5.2.2 Query with data join

The query present in Figure 5.6(b) generates the canonical model *mSaaS* described in Figure 5.6(a). The query is described in SQL and NoSQL (MongoDB) with data join.



SQL

```
SELECT w7.firstname,
   w7.age, vz.phone
FROM w7
LEFT OUTER JOIN vz
   ON w7.id = vz.ID
WHERE vz.borough='queens'
ORDER BY w7.firstname
LIMIT 3
```

NoSQL (MongoDB)

```
db.w7.aggregate.([
   {$lookup: {from: 'vz',
     localField: 'id',
     foreignField: 'ID'}},
   {$match: {'vz.borough' = 'queens'}}
   {$project: {'w7.firstname':1,
      'w7.age':1, 'vz.phone':1}},
   {$sort: {'w7.firstname':1}},
   {$limit: 3}
]);
```

Figure 5.6 Example of *mSaaS* (a) for a SQL/NoSQL query with data join (b).

The main node of *mSaaS* in Figure 5.6(a) points to three subtrees: *q1*, *q2*, and *param*. The first two sub-trees (*q1* and *q2*) store data about the query. A query with data join relates two or more data sources (in this case, two data sources). Each data source is accessed through a URL. Consequently, our model must convert a query with join into multiple URLs, so that each data source is accessed by one distinct URL. Since the query performs data join between two data sources (*w7* and *vz*), the model groups the data sources information separately:

- *q1* subtree stores (i) projection attributes (*firstname* and *age*) and (ii) data source name (*w7*).
- *q2* subtree stores (i) projection attribute (*phone*), (ii) selection condition (*borough* = *'queens'*), and (iii) data source name (*vz*).
- *param* subtree stores data about (i) sort attribute (*w7.firstname*), (ii) limit clause value (*3*), (iii) join type (*left outer*), (iv) join condition (*w7.id* = *vz.ID*), and (v) return format (*xml*).

Similarly to the query without data join, the values present in the *v* level are independent of the language used in the query.

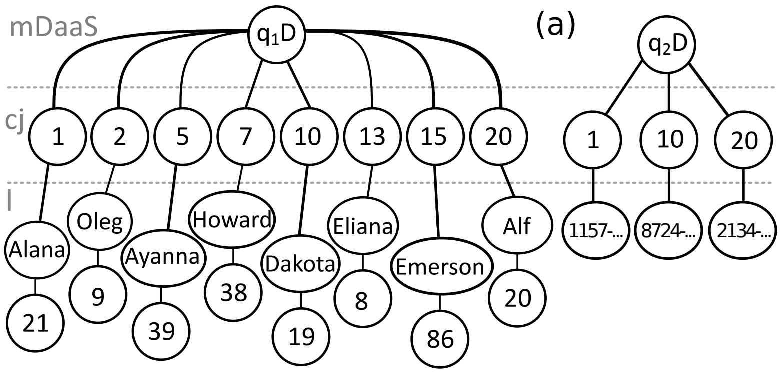
Given *mDIS* (Figure 5.2) and *mSaaS* (Figure 5.6(a)), the function `generateURLs` (Listing 5.1) generates the following URLs:

```
URL1 = http://w7.com/api/w/?dsw=w7&rcw=id,firstname,age
```

and

```
URL2 = http://vz.com/api/v/?dsv=vz&rcv=ID,phone&qv=borough='queens'
```

Considering that the query with join (Figure 5.6(b)) returns the two datasets present in Figure 5.7(b), *mDaaS* creates the canonical models is Figure 5.7(a).



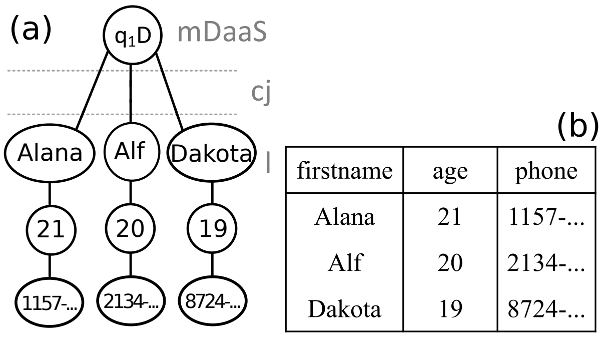
w7		
id	firstname	age
1	Alana	21
2	Oleg	9
5	Ayanna	39
7	Howard	38
10	Dakota	19
13	Eliana	8
15	Emerson	86
20	Alf	20

vz	
ID	phone
1	1157-...
10	8724-...
20	2134-...

(b)

Figure 5.7 Example of *mDaaS* (a) for data returned by query with data join (b).

After MIDAS receives the return from two DaaS, the middleware performs the steps for *left outer join* (Subsection 5.1.3). We assume that *mDaaS* (Figure 5.8(a)) represents the data that should be returned to the SaaS (Figure 5.8(b)).



firstname	age	phone
Alana	21	1157-...
Alf	20	2134-...
Dakota	19	8724-...

Figure 5.8 Final result after model performs *order*, *limit* and *left outer join* clauses for the query with join.

We evaluate the correctness of our model for syntactic interoperability based on two situations: query with and without data join. In this evaluation, we perform both queries with our model and then we perform both queries on MIDAS. We noted that our model represented without loss of information the syntactic interoperability between SaaS and DaaS provided by MIDAS.

5.3 RELATED WORK

Our model for syntactic interoperability is based on SQLtoKeyNoSQL (SCHREINER; DUARTE; MELLO, 2015). SQLtoKeyNoSQL is a layer for relational to NoSQL database mapping. This solution enables users to perform NoSQL databases similar to relational databases. A hierarchical canonical model maps relational schemes into NoSQL schemes. This canonical model is composed of trees and key-values. We adapted SQLtoKeyNoSQL's canonical model to represent syntactic interoperability in the cloud environment.

Benzadri, Belala and Bouanaka (2014), Ma et al. (2012) formally specify cloud services, customers, and their interactions. Although Benzadri, Belala and Bouanaka (2014) focuses on the most common service model (i.e., SaaS, PaaS, and IaaS), the authors aim to reason about concepts focusing on cloud deployment models. Furthermore, this work does not focus on the interoperability concern.

Ma et al. (2012) specifies the semantics of services with a focus on web services composition in the form of ontology. These descriptions represent the types of service, conditions, and functionality of the service. In contrast to our work, the authors focused on service specification, discovery, composition, and orchestration.

5.4 CHAPTER SUMMARY

This chapter presented the lightweight formal description of MIDAS syntactic interoperability (RIBEIRO et al., 2018). Based on the bottom-up approach, our model describes the syntactic interoperability of MIDAS 1.8. We introduced the canonical model and mathematical definition of each MIDAS component. As proof of concept, we performed two different queries in our model: with and without data join. Our results demonstrate that our model was able to describe both queries.

In the next chapter, we present our model for semantic interoperability. We also report model evaluation into MIDAS middleware.

Chapter 6

This chapter presents our model for semantic interoperability. We detail the semantic interoperability of MIDAS, and then we present the evaluation of this model. Finally, we present some related work.

MODEL FOR SEMANTIC INTEROPERABILITY

Our model for semantic interoperability is based on MIDAS 2.0 (MANE et al., 2020) for chronological reasons. We employ a bottom-up strategy, similar to syntactic interoperability. This model aims to detail the semantic interoperability among SaaS, DaaS, and DBaaS.

Although MIDAS (1.0, 1.6, 1.8, and 1.9) provides syntactic interoperability among cloud layers, semantic interoperability has only recently been addressed in MIDAS 2.0. *MIDAS-OWL* provides semantic interoperability between SaaS and DaaS/DBaaS, which was not possible in the previous versions of MIDAS. *MIDAS-OWL* is an OWL-based ontology to reason about the communication among SaaS, DaaS, and DBaaS. With such an ontology, we can translate queries independently of changes in the technology, format, and information structure transparently. Our ontology represents some structural aspects of DaaS and DBaaS, as well as queries. This ontology enables explicitly expressing the semantic similarity between different queries and DaaS/DBaaS attributes.

We create a query vocabulary considering some query clauses and DaaS information. Our ontology is an artifact of knowledge representation that aims to (i) formalize the query taxonomy; (ii) establish relationships between different DaaS explicitly; and (iii) separate business rules from knowledge rules, i.e., how MIDAS works from how MIDAS integrates data.

MIDAS-OWL eases the query building through of (re)writing process. Our ontology performs semantic integration based on this (re)writing queries. *MIDAS-OWL* represents queries and DaaS structure. Our ontology facilitates semantic interoperability between SaaS and DaaS.

We develop *MIDAS-OWL* in Ontology Web Language (OWL) based on Methontology (FERNÁNDEZ-LOPEZ; GÓMEZ-PEREZ; JURISTO, 1997). We present an implementation and evaluation using Protégé 5.5.0[1] and Pellet[2] reasoner. Pellet is an OWL-DL reasoner, i.e., the DL $\mathcal{SROIQ}^{(D)}$ (SIRIN et al., 2007).

[1]https://protege.stanford.edu/

[2]https://www.w3.org/2001/sw/wiki/Pellet

We perform some experiments to evaluate the consistency, correctness, acceptance, and integration of our model for semantic interoperability. Our results pointed out that *MIDAS-OWL* provides (i) data select through query rewriting, (ii) integrating data-based on semantically similar attributes, and (iii) a higher abstraction of SaaS and DaaS layers.

The next section details our model for semantic interoperability. Afterwards, we present the evaluation of this model. Finally, we briefly discuss our findings and related work.

6.1 MODEL

The literature presents some methodologies for the development and maintenance of ontologies, such as Enterprise (USCHOLD; KING, 1995), On-To-Knowledge (STAAB et al., 2001), TOVE (GRÜNINGER; FOX, 1995), and Methontology (FERNÁNDEZ-LOPEZ; GÓMEZ-PEREZ; JURISTO, 1997).

Methontology provides a set of elements to support ontology development, e.g., knowledge acquisition, specification, conceptualization, formalization, and others. The development of our ontology considers the (i) specification, (ii) conceptualization, (iii) formalization/implementation, and (iv) evaluation.

We apply qualitative research approach for the specification, conceptualization, and evaluation, and Protégé tool and reasoner Pellet for the formalization/implementation and evaluation.

MIDAS-OWL ontology is available at Appendix B. A complete documentation is accessible at `<https://elivaldolozer.github.io/ontmidas/>`.

6.1.1 Specification

The focus of the specification stage was to understand the project scope. We carried out interviews and meetings with MIDAS experts to obtain MIDAS specific terms and how these terms relate to each other. MIDAS experts have different degrees levels (e.g., bachelor, master, and Ph.D.) and different middleware roles (e.g., developers, requirement analysts, and project manager). Table 6.1 shows some competency questions which were carried out in the interviews and meetings with experts.

The answer to each competency question is presented as follows:

- CQ_1: MIDAS receives queries through a SaaS;
- CQ_2: middleware recognizes SQL and MongoDB (NoSQL) queries;
- CQ_3: MIDAS recognizes projection (`SELECT` in SQL and `$project` in MongoDB), selection (`WHERE` in SQL and `$match` in MongoDB), order (`ORDER BY` in SQL and `$sort` in MongoDB), limit (`LIMIT` in SQL and `$limit` in MongoDB), and join (`LEFT JOIN` in SQL and `$lookup` in MongoDB) clauses. The clauses recognized by MIDAS are subclasses of the *clauses* class (blue rectangle in Figure 6.1);
- CQ_4: when MIDAS receives a query, the middleware (i) converts the query to a specific pattern and (ii) based on this format creates a URL;

Table 6.1 Competency questions (CQ) applied in the specification of our ontology

	Competency Question (CQ)
CQ_1	How to submit queries in MIDAS?
CQ_2	Which languages are recognized by MIDAS?
CQ_3	Which clauses are recognized by MIDAS?
CQ_4	What does MIDAS perform when it receives a query?
CQ_5	How does MIDAS receive data return from DaaS?
CQ_6	How are the joins performed since data is returned from different DaaS/DBaaS?
CQ_7	What is a relevant information about each DaaS?

- CQ_5: MIDAS experts state that middleware recognizes data returned by DaaS in CSV, XML, and JSON files (red rectangle in Figure 6.1);
- CQ_6: middleware (i) receives the data by source (DaaS or DBaaS), (ii) standardizes the results in a single format (e.g., XML), (iii) performs the join of these data based on the query, and (iv) sends the result to the SaaS;
- CQ_7: DaaS name, return formats, domain address, search path, attribute names, and the parameter associated with the projection, selection, order, and limit clauses (green rectangle in Figure 6.1). This information allows the ontology to performs the exchange of messages between a SaaS and different DaaS.

From the responses, we establish the following specification:

- based on the interviews and meetings with experts, the domain of *MIDAS-OWL* is queries in cloud computing;
- the focus of *MIDAS-OWL* is to represent queries submitted to MIDAS by a SaaS, i.e., knowledge representation; and
- *MIDAS-OWL* will contribute to the implementations of semantic MIDAS version from (re)writing queries.

6.1.2 Conceptualization

After the specification step, and before implementing *MIDAS-OWL* on Protégé, we depict a representation of the abstract view of our ontology. This representation emphasizes the classes and relationships. Figure 6.1 shows this representation of *MIDAS-OWL*. Based on this representation, we define 44 classes and 41 relations. *Clauses* and *attribute* are examples of classes, while *has_attrb* and *composed_of* are examples of relations.

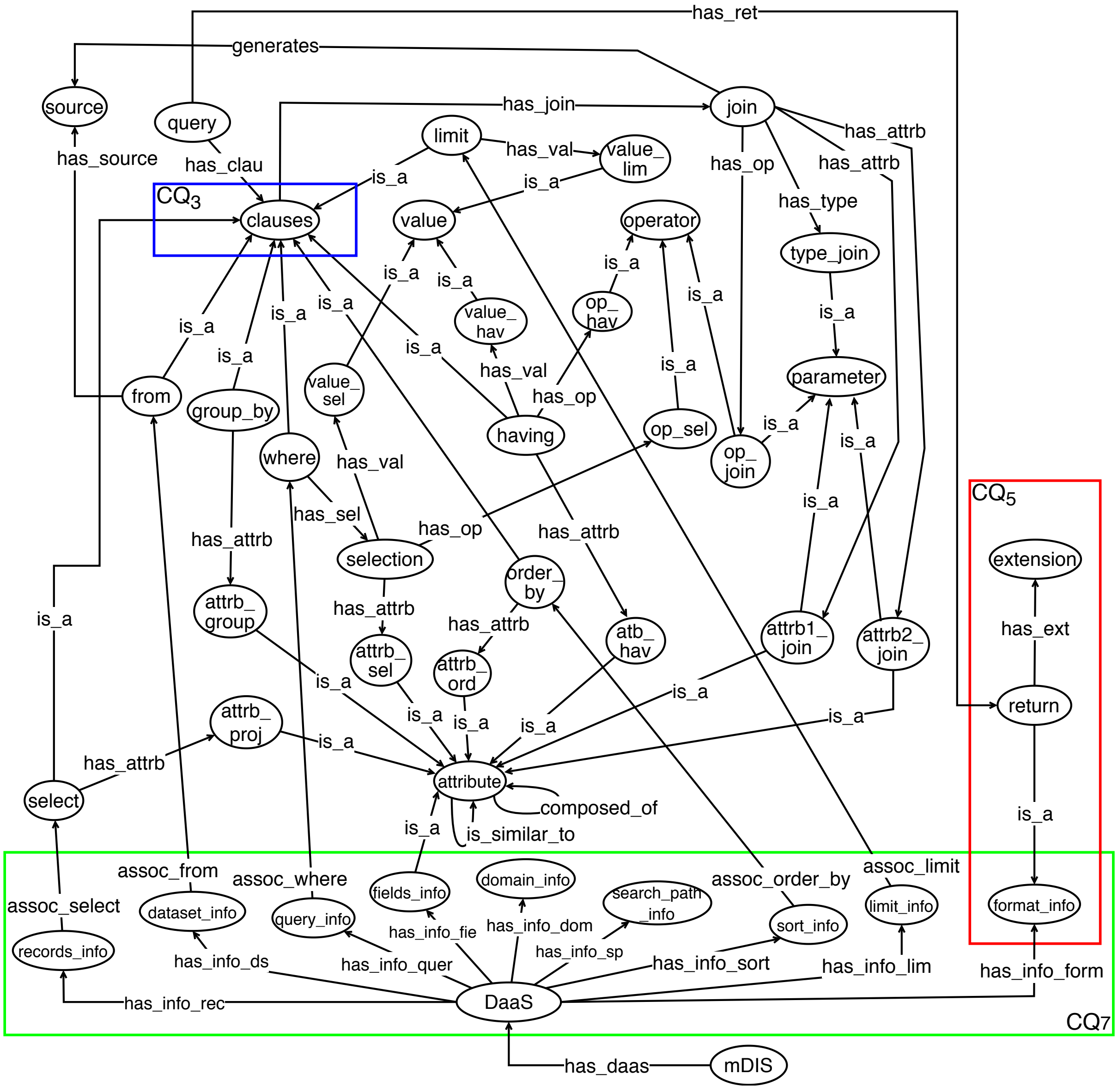


Figure 6.1 *MIDAS-OWL* overview.

6.1.3 Formalization and Implementation

In this step, we provide our representation into a Protégé format. We create 14 classes, 30 subclasses, 41 object properties, and 59 individuals.

6.1.3.1 Classes and Subclasses. Class is a group of individuals that share some properties. A class can be divided into subclasses (ANTONIOU; HARMELEN, 2004). We create classes and subclasses represented in Figure 6.1. Table 6.2 presents some classes of our ontology.

Table 6.2 Examples of *MIDAS-OWL* classes

Class	Subclass of	Meaning
clauses	—	this class represents all the clauses that a query may have, such as *select*, *from*, *limit*, *order by*, among others
select	*clauses*	this class maps the projected attributes by a query
where	*clauses*	this class specifies criteria for selecting data
op_sel	*operator*	this class maps the operator present in the where clause, such as *greater than* (>), *not equal to* (<>), among others

Figure 6.2 presents the taxonomy of *clauses* class. This class represents types of clauses that may appear in a query to a DaaS: *where*, *select*, *order_by*, *having*, *group_by*, *limit*, and *from*. Each of these clauses can be differently performed for each specific DaaS, depending on its foundational technology, structure, among others. For instance, a *select* clause may be performed in SQL as `SELECT firstname FROM w7` and in MongoDB (NoSQL) as `db.w7.find({}, {"firstname": 1})`.

6.1.3.2 Object Properties. Object property relates individuals of one class with individuals of another class (ANTONIOU; HARMELEN, 2004). We define object properties based on Figure 6.1. Table 6.3 presents some object properties of *MIDAS-OWL* ontology.

6.1.3.3 Individuals. Protégé recognizes individuals as instances. Instances are real data from the knowledge base. An instance is the real value of a class (ANTONIOU; HARMELEN, 2004).

Since *MIDAS-OWL* aims to represent a domain conceptually, instantiating individuals is not the scope of this work. In the future, these instances will be populated by MIDAS from each DaaS. Nevertheless, we instantiate some individuals to facilitate the understanding of our model for semantic interoperability.

The evaluation of the model for semantic interoperability is presented in Section 6.2.

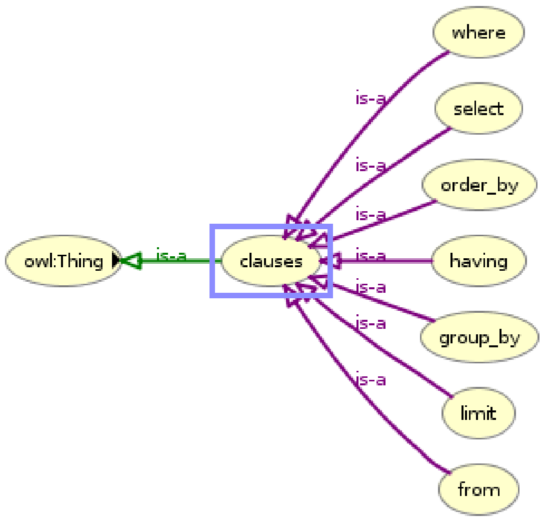


Figure 6.2 Taxonomy of the *Clauses* class.

6.2 EVALUATION

We evaluate our model for semantic interoperability based on the qualitative research approach (LOVRENCIC; CUBRILO, 2008).

We performe three distinct experiments. Our first experiment concerns with *MIDAS-OWL* concepts and taxonomy. This experiment aims to assess the correctness and acceptance of concepts to construct classes and object properties. In the second experiment, we depict a motivating example. This motivating example evaluates the correctness and performance of *MIDAS-OWL* within MIDAS middleware. The third experiment performs two queries through SaaS to MIDAS: with and without a join clause. This experiment aims to evaluate whether the elements meet all clauses in different queries. Experiments 1 and 3 evaluate intrinsic aspects of the ontology, and experiment 2 performs our ontology inside the MIDAS middleware. Experiments 2 and 3 address interoperability issues between SaaS and DaaS cloud services (according to Table A.1 and Table A.2 in Appendix A).

6.2.1 Experiment 1: Evaluation of Concepts

The first experiment checks MIDAS-OWL consistency. We perform the consistency check test through the Pellet reasoner, looking for inconsistencies in logic and concepts. As a result, the Pellet reasoner found no inconsistencies in the ontology. Afterwards, MIDAS experts evaluated our ontology. This evaluation was carried out during two meetings. Each meeting took three hours approximately.

A group of eight MIDAS experts evaluates the ontology based on the produced documentation. Experts evaluate the classes and object properties of the ontology. After evaluating the ontology model, the experts indicate which items of the ontology are in compliance with MIDAS 2.0. Experts approved all 14 classes and 30 subclasses. Addi-

Table 6.3 Examples of *MIDAS-OWL* object properties

Property	Domain	Range	Meaning
has_clau	*query*	*clauses*	this property indicates that a *query* has some *clauses*, such as *select*, *from*, *limit*, among others
has_attrb	*selection*	*attrb_sel*	this property indicates that a *selection* has a set of attributes (*attrb_sel*)
assoc_from	*dataset_info*	*from*	this property associates the DaaS name (*dataset_info*) with the *from* clause of the query
is_similar_to	*attribute*	*attribute*	this property represents that an *attribute* can be semantically similar to another *attribute*

tionally, the experts suggested the inclusion of one object property (upgrading it to 42). No exclusion was indicated. Table 6.4 presents examples of the assessment.

Table 6.4 Examples of the ontology items evaluation

Type	Name	Evaluation	Meaning
Class	*clauses*	Approval	class represents all the clauses of query, such as *select*, *from*, *limit*, among others
Subclass	*from*	Approval	subclass represents the *from* clause of the query
Object property	*is_similar_to*	Inclusion	property indicates that an *attribute* is semantically similar to another *attribute*

6.2.2 Experiment 2: Motivating Example

As a motivating example to illustrate *MIDAS-OWL*, we suppose a hospital system that consumes data distributed across multiple DaaS, such as *w7* and *vz* (Appendix A). In emergency situations, the medical team may need the patient's information from different data sources. In such cases, family contact can take a long time. Moreover, knowing the historical medical diseases from those patients is vital to prescribe new drugs or not. *MIDAS-OWL* enables the hospital to retrieve patient data stored in both DaaS.

We assume that medical team requires patient data (e.g., *first name*, *last name*, *age*, and *blood type*) published in DaaS *w7*. Since *w7* is a proprietary cloud, *w7* owner can update DaaS anytime. Therefore, a single label change in *w7* can suspend the interoperability between *w7* and hospital. Consequently, the lack of information can prevent an emergency prescription of some drugs to save lives.

To surpass such a problem, *MIDAS-OWL* triggers similar queries (with similar attributes) to the same data source. In this case, if *w7* owner changes the label from *last name* to *family name*, then two URLs are created: one with the *last name* attribute (incorrect) and another with the *family name* attribute (correct). We consider as correct the query that returns some valid result. We consider as valid the result without error, such as *invalid field*, *runtime error*, *syntax error*, and others.

Although our motivating example illustrates the query rewriting resulting from the change in the attribute name, *MIDAS-OWL* enables rewriting due to other changes, such as the DaaS name. In this case, a property to define semantically similar DaaS names would be required.

We assume that the query submitted by the hospital in SaaS is as follows:

```
SELECT firstname, lastname, age, blood
FROM w7
WHERE lastname = "Torres"
```

In the case that there is no change in label attributes, MIDAS performs the query without problems. The URL created to collect data in w7 DaaS can be exemplified as follows:

```
URL1 = http://w7.com/api/w/?rcw=firstname,lastname,age,blood&qw=
    lastname="Torres"
```

This URL (URL$_1$) returns data about a specific patient.

In the case of changing the label from *lastname* to *familyname*, *MIDAS-OWL* enables MIDAS to create similar queries automatically. The ontology allows the rewriting of queries based on semantically similar attributes. In this case, MIDAS creates two URLs instead of just one. In addition to the URL created in the previous case (URL$_1$), MIDAS creates another URL[3] since *lastname* and *familyname* are semantically similar. The second URL (URL$_2$) can be exemplified as follows:

```
URL2 = http://w7.com/api/w/?rcw=firstname,familyname,age,blood&qw=
    familyname="Torres"
```

MIDAS-OWL changed the old label (*lastname*) to the current one (*familyname*) in this URL. Table 6.5 summarizes the data returned by URL$_2$ after changing the label from *lastname* to *familyname*. In this scenario, URL$_1$ returned an error since the *lastname* attribute does not exist anymore.

Table 6.5 Data returned by URL$_2$ after changing the label

firstname	lastname/familyname	age	blood
Ayanna	Torres	39	B+

[3]Both URL$_1$ and URL$_2$ are fictitious.

MIDAS automatically translates queries written for the outdated version of DaaS (with *lastname*) into the updated version of DaaS (with *familyname*). This new query ensures backward compatibility of any SaaS that communicated with the DaaS transparently, even after changes in the DaaS structure.

We are aware that performing two queries instead of one could lead to performance problems in MIDAS. However, we emphasize that the benefits of our approach outperform the overhead caused.

6.2.3 Experiment 3: Query Running

In this third experiment, we evaluate the ontology correctness by running queries with and without data join based on DaaS in Appendix A. This experiment aims to evaluate the representation power of the ontology and its usefulness for representing, building, and reasoning about queries in MIDAS. For this reason, in this experiment, queries are purposefully simple. We present and discuss the queries in the following subsections.

6.2.3.1 Query without Data Join. We consider the following query without a join clause in this evaluation:

```
SELECT firstname, lastname, age
FROM w7
WHERE age > 10
ORDER BY lastname
LIMIT 5
```

Figure 6.3 represents the query without the join clause. The query without join is then broken down into individual clauses.

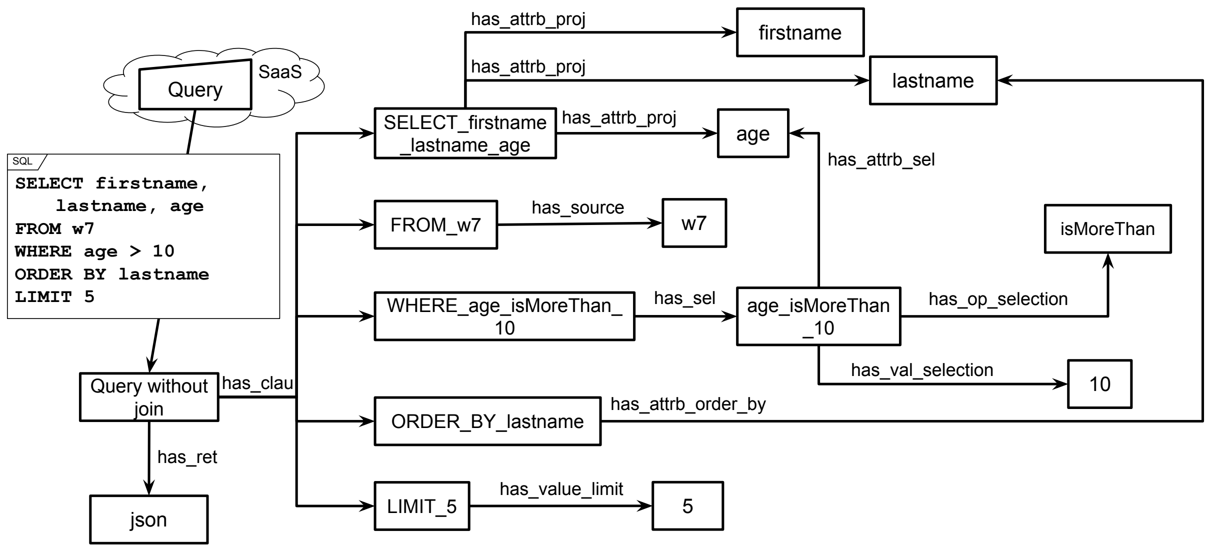


Figure 6.3 Detailed clauses of the query without data join in our ontology.

In this example, SaaS sends a SQL query without join by SaaS. *MIDAS-OWL* maps the query's clauses to obtain a representation of the elements from SaaS:

i) *SELECT_firstname_lastname_age* has three projection attributes (*firstname*, *lastname* and *age*);

ii) *FROM_w7* has one source (*w7*);

iii) *WHERE_age_isMoreThan_10* has the three elements necessary to perform the attributes selection: attribute (*age*), operator (*isMoreThan*), and value (*10*);

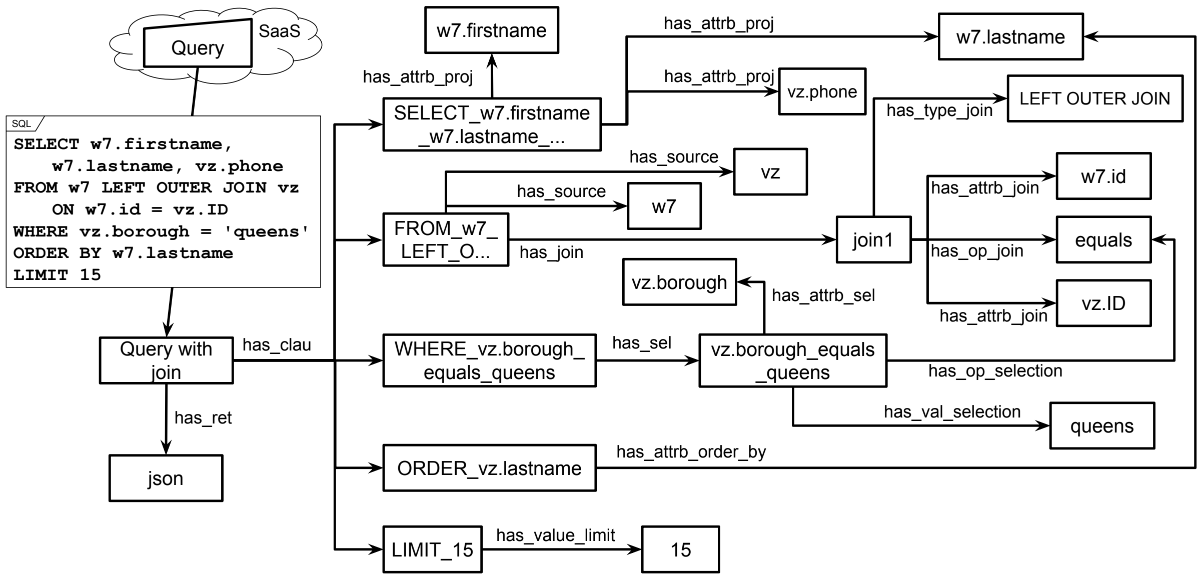
iv) *ORDER_BY_lastname* has order by attribute (*lastname*); and

v) *LIMIT_5* has limit value (*5*).

6.2.3.2 Query with Data Join. We consider the following query with a join clause in this evaluation:

```
SELECT w7.firstname, w7.lastname, vz.phone
FROM w7 LEFT OUTER JOIN vz ON w7.id = vz.ID
WHERE vz.borough = 'queens'
ORDER BY w7.lastname
LIMIT 15
```

Figure 6.4 represents the query with the join clause. This query is then broken down into individual clauses similar to previous query.



```
SELECT w7.firstname,
    w7.lastname, vz.phone
FROM w7 LEFT OUTER JOIN vz
    ON w7.id = vz.ID
WHERE vz.borough = 'queens'
ORDER BY w7.lastname
LIMIT 15
```

Figure 6.4 Detailed clauses of the query with data join in our ontology.

In this example, SaaS sends a SQL query with join by MIDAS SaaS. As in the previous example, *MIDAS-OWL* maps the query's clauses to obtain a representation of the elements from SaaS:

i) *SELECT_w7.fisrtname_w7.lastname_vz.phone* has three projection attributes (*w7.fisrtname*, *w7.lastname*, and *vz.phone*);

ii) *FROM_w7_LEFT_O...* has two sources (*w7* and *vz*), and join clauses: type join (*LEFT OUTER JOIN*), attributes join (*w7.id* and *vz.ID*), and operation join (*equals*);

iii) *WHERE_vz.borough_equals_queens* has the three elements necessary to perform the attributes selection: attribute (*vz.borough*), operator (*equals*), and value (*'queens'*);

iv) *ORDER_BY_vz.lastname* has order by attribute (*vz.lastname*); and

v) *LIMIT_15* has limit value (*15*).

Based on the representation effort, we evaluate that the ontology is appropriate in the context of query representation in MIDAS. This evaluation suggests that our ontology can be used by MIDAS to represent the DaaS structure. Additionally, our ontology can be used in the query building process, since it allows automation of (re)writing queries, which will enable semantic data integration.

6.3 DISCUSSION

MIDAS-OWL structure is independent of the label, value, and type of the attributes from a DaaS. Each attribute is an individual of the ontology. The ontology can consider parameters such as label (e.g., *firstname* and *age*), value (e.g., *Alana* and *39*), and type (e.g., *string* and *integer*) of attributes. Similar attributes do not imply attributes with the same content, as they can have different representations.

Another prominence point is *MIDAS-OWL* flexibility. Our model for semantic interoperability makes no assumptions about the structure and technology of data sources. *MIDAS-OWL* works transparently and does not require other reasoning services. Semantic similarity detection in DaaS (MANE et al., 2020) and methods based on ontology alignment (EUZENAT; SHVAIKO et al., 2007) are examples of strategies for populating our ontology.

Currently, MIDAS-OWL recognizes only SQL queries. Although all queries presented in our experiments were in SQL format, the model for semantic interoperability can provide queries in other formats, such as NoSQL and SPARQL Protocol and RDF Query Language (SPARQL). In such cases, MIDAS middleware needs to recognize the query format and the corresponding clauses into respective classes/subclasses of the ontology. For instance, `SELECT` clause (SQL queries) might be replaced by `find` clause (MongoDB queries). Additionally, relations must be reevaluated to represent the new format.

About the limitations, since our focus was on a direct attribute-attribute similarity, the absence of complex queries (e.g., queries with multiple join conditions) was not that important. More complex similarities (e.g., attribute-query type) are not part of the scope of this research. However, we emphasize that with this second type of similarity, a high computational overhead could incur with rewriting queries.

6.4 RELATED WORK

Despite the amount of research dedicated to ontology design, to our knowledge, data representation in the cloud environment has received far less attention in the literature.

Among the work on semantic interoperability, the use of ontologies and knowledge representation models to reason about data integration has been proposed by Vidal et al. (2009), which present an ontology for integrating geographic data. Their approach does not focus on cloud computing, and authors do not explicitly consider heterogeneous data sources. Differently than in their work, our ontology deals with data interoperability, not integration.

We also analyzed some works that propose using ontologies and semantic web technologies in the area of cloud computing. Rekik, Boukadi and Ben-Abdallah (2015) propose an ontology for describing cloud services. Their ontology takes into account the most common cloud service model: SaaS, PaaS, and IaaS. The authors tackle heterogeneous sources, but they solve the problem in service level without data.

Joshi, Yesha and Finin (2014) present an OWL ontology to describe the cloud service lifecycle. Their ontology provides models for requirement, discovery, negotiation, composition, and consumption phases of cloud services. Although they present the design of their ontology in cloud computing, their focus is on the relationship among the five phases of the model and how each phase relates to consumer and service cloud. Authors do not consider communication among different cloud services.

Moscato et al. (2011) propose mOSAIC, an ontology for negotiation (for instance, between customers and providers) and composition (for instance, by an administrator) of cloud services. Authors claim that ontology can improve interoperability among cloud solutions, platforms, and services. Unlike our approach, interoperability is addressed at a service level and not at a data level.

Zhang, Haller and Wang (2019) present CoCoOn, an ontology that defines concepts, attributes, and relationships to IaaS. CoCoOn recommends IaaS services based on price and features. Similar to previous work, this article considers neither service data nor interoperability among different cloud services.

In Ontology-Based Query Answering (OBQA), an ontology eases query rewrite based on a union of conjunctive queries (MUGNIER; THOMAZO, 2014). Contrary to our *MIDAS-OWL*, OBQA assumes that an ontology supports all databases and that, moreover, these ontologies are jointly consistent (MUGNIER; THOMAZO, 2014; ORTIZ, 2013). This assumption is not feasible in cases where the semantics of data is not formalized, such as DaaS service. In such a case, only access to the attributes is available with no schema information. Such a lack of knowledge happens because we deal with distributed and independent data sources. The different conceptualizations of the domain can be inconsistent with each other.

We emphasize that our approach is less expressive than OBQA. OBQA is a high-level solution to a well-defined problem. Notwithstanding, the OBQA results are not satisfactory when the solution is applied on large bases, such as Cloud Computing (AKERKAR, 2014). *MIDAS-OWL* deals with a more dynamic problem than OBQA. Although our solution is very similar to data integration and OBQA approaches, we perform *MIDAS-OWL* in a more general context.

6.5 CHAPTER SUMMARY

This chapter described *MIDAS-OWL*, an ontology to explain communication among cloud layers. *MIDAS-OWL* describes the MIDAS semantic interoperability. Our model for semantic interoperability assists SaaS applications to access data in a cloud transparently, even after updating attributes in DaaS. We perform some experiments to validate the model for semantic interoperability and show our approach's effectiveness.

MIDAS-OWL provides aspects that are not considered in the current MIDAS version, as such query rewriting. We emphasize that if two attributes are similar, then these attributes can perform a join without the SaaS having to map this relationship explicitly. In a syntactic query, SaaS needs to explicitly know the structure of a DaaS and each relationship of DaaS data. Our *MIDAS-OWL* moves towards making the interoperability provided by MIDAS more transparent.

The next chapter presents the CAPITAL, our model for pragmatic interoperability and the model evaluation.

Chapter

7

This chapter presents the model for pragmatic interoperability. We detail our pragmatic interoperability, and then we present the evaluate of this model based on four scenarios. Finally, we present some related work.

MODEL FOR PRAGMATIC INTEROPERABILITY

Contrary to syntactic and semantic interoperability, we employ a top-down strategy in our model for pragmatic interoperability for two reasons. First, we realize that there is no consensus on the pragmatic interoperability definition. Second, our intention is to provide a domain-independent model for pragmatic interoperability. Therefore, each scenario is coded depending on their requirements. For these reasons, before investigating the pragmatic interoperability of MIDAS (Chapter 8), we present our unified definition for pragmatic interoperability and our CAPITAL. Our CAPITAL framework comprises three artifacts: a canonical model, a textual definition, and schemas in Z notation.

We evaluate the CAPITAL framework into four distinct scenarios as modeling and implementation guidance: (i) service from laboratory tests, (ii) service that checks DNA sequence ancestry, (iii) service of the car's Bluetooth, and (iv) service related to the public security domain. These scenarios (i) represent current and future situations, and (ii) address effectiveness, completeness, and mandatory aspects of the CAPITAL.

Our results pointed out that our CAPITAL fulfills the mentioned issues since the framework identifies the mandatory elements to provide pragmatic interoperability based on different scenarios. We believe that pragmatic interoperability is achieved when the context and intention intersection increases. Our intuition is that a common and shared understanding of pragmatic interoperability guides towards full interoperability, even with different strategies and systems.

The next section details our model for pragmatic interoperability. Afterwards, we evaluate this model based on four scenarios. The artifacts are presented in detail. Finally, we briefly discuss our findings and related work.

7.1 MODEL

A conceptual framework aims to present a literature synthesis on a specific topic and show how related elements are associated with each other (JABAREEN, 2009). We develop our CAPITAL framework based on four activities: definition, investigation, specification, and evaluation (Figure 7.1). We start by unifying the definition. We perform a search for definitions on pragmatic interoperability, and then we list eight definitions. Afterwards, we analyze each definition and identify four recurring terms in the pragmatic interoperability definitions. We describe the CAPITAL framework and, finally, we evaluate our framework into four distinct scenarios.

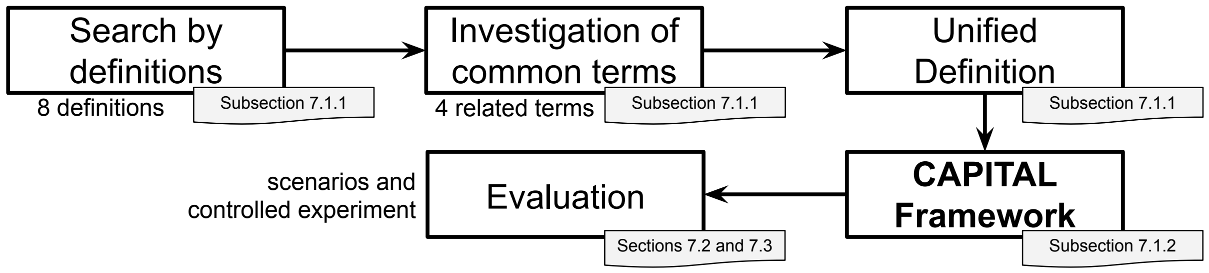


Figure 7.1 Activities for CAPITAL framework.

Our CAPITAL framework comprises a canonical model, a textual definition, and schemas in Z notation. A canonical model is a universal data model based on trees and key-values (SCHREINER; DUARTE; MELLO, 2015). Our canonical model details terminologies related to pragmatic interoperability towards a unified definition. The text notation provides a textual description of the elements involved in the pragmatic interoperability concept. Z notation is a formal language that describes computational systems with mathematical concepts, such as predicate logic and set theory (SPIVEY, 1989). While the canonical model presents a structural relationship among elements, the textual description enables a non-formal understanding of the framework. Z notation simplifies the coding of systems with pragmatic interoperability. Figure 7.2 summarizes the artifacts in CAPITAL framework.

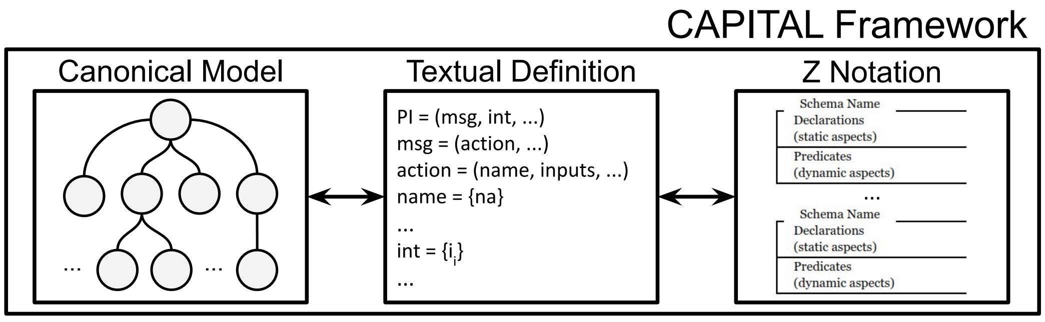


Figure 7.2 CAPITAL's Artifacts: Canonical model, textual definition, and schemes in Z notation.

The following subsections depict the literature's definitions, related terms, our unified definition of pragmatic interoperability, and CAPITAL framework: our model for pragmatic interoperability.

7.1.1 Pragmatic Interoperability Unified Definition

Pragmatic interoperability definitions from the literature are summarized in Table 7.1. The set of definitions provides evidence that there is no consensus of pragmatic interoperability definition.

Table 7.1 Pragmatic interoperability definitions in chronological order

Definition	Reference
Pragmatic interoperability ensures that the messages exchanged have the desired effect. Usually, this occurs by sending and receiving multiple messages in a specific order, defined by a protocol.	Pokraev et al. (2005)
Pragmatic interoperability is achieved when one system knows the methods and procedures of other systems.	Lee et al. (2007)
The pragmatic web is a set of pragmatic contexts about semantic resources.	Tamani and Evripidou (2007)
Pragmatic interoperability is the compatibility between the intended effect and the actual effect of message exchange.	Asuncion and Sinderen (2010)
Pragmatic interoperability is the compatibility between intended use and actual use of message within a relevant shared context.	Asuncion et al. (2011)
At the system level, pragmatic interoperability is a shared understanding between intended use and messages actual use within a context. At the business level, pragmatic interoperability considers the compatibility of business intentions, business rules, and organizational systems policies.	Asuncion et al. (2011)
Pragmatic interoperability is achieved when processes from different contexts are compounded to support a common intention. The integration emphasis is context-awareness.	Liu, Li and Liu (2014)
Pragmatic interoperability uses syntax and semantics as a tool to achieve goals.	Webster (2014)

Despite the lack of consensus, we identify in Table 7.1 the existence of repeated terms. For that reason, we list the related terms to make up a unified definition. The literature presents several definitions for related terms, such as:

- **use**: (i) how to realize the sender's intention; (ii) how the receiver interprets the intended information; and (iii) how systems use data;
- **intention**: (i) message objective; (ii) actions desired by the message; and (iii) possible desired state that a sender can achieve through collaboration;
- **context**: (i) circumstance in which the message is shared; (ii) contextual dimensions: why, how, when, who, where, and what; and (iii) set of contexts in semantic resources; and

- **effect**: (i) relationship between information, action, and context; (ii) (ii) it requires the receiver to understand the message intent deeply; and (iii) it can be accomplished by sending and receiving messages in a specific order.

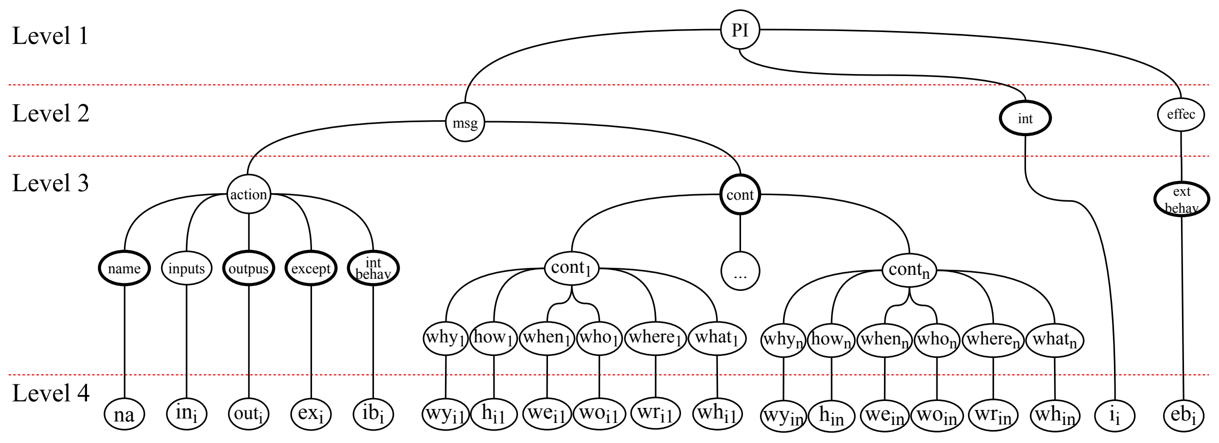
Based on these descriptions, we define the related terms as follows: (i) **use** is the message achievement and it depends on the intent and context to ensure achievement of the desired effect; (ii) **intention** is the collaboration objective and it is composed of a set of states (conditions); (iii) **context** is the information necessary to achieve the effect; and (iv) **effect** is the result or consequence.

Based on interpretation of the non-consensual definitions (Table 7.1) and related terms of pragmatic interoperability (use, intention, context, and effect), our pragmatic interoperability unified definition is as follows:

> *At the system level, pragmatic interoperability is achieved when there is a shared understanding of the* ***intention*** *and* ***context*** *necessary for communication, aiming to provide the correct* ***use*** *of the message and produce results within the expected* ***effects****.*

7.1.2 CAPITAL Framework

From the definitions of Table 7.1, the set of terms, and our definition, Figure 7.3 depicts the canonical model of the CAPITAL framework. Mandatory elements are bold. Although there are other approaches to model context, we apply the 5W1H[1] format (ISODA; KURAKAKE; IMAI, 2005). This format is frequently employed in context-sensitive systems.



Level 1
Level 2
Level 3
Level 4
PI
msg
int
effec
action
cont
ext behav
name
inputs
outpus
except
int behav
$cont_1$
...
$cont_n$
why_1 how_1 $when_1$ who_1 $where_1$ $what_1$
why_n how_n $when_n$ who_n $where_n$ $what_n$
na in_i out_i ex_i ib_i
wy_{i1} h_{i1} we_{i1} wo_{i1} wr_{i1} wh_{i1}
wy_{in} h_{in} we_{in} wo_{in} wr_{in} wh_{in}
i_i
eb_i

Figure 7.3 Canonical model of our CAPITAL framework. Mandatory elements are bold.

In the root (level 1) there is the term pragmatic interoperability (PI). PI is composed of three sets of information at level 2: *msg* (representing the message), *int* (representing the term **intention**), and *effec* (representing the term **effect**).

[1]Why, When, Who, Where, What, and How

Level 3 details the elements of level 2: (i) *msg* is composed of *action* and *cont* (context), and (ii) *effec* contains the behavior performed by the receiver (*ext behav*). *Action* represents the term **use**, and *cont* represents the term **context**. Still at level 3, *action* is composed of the service *name*, *inputs*, *outputs*, exceptions (*except*), and behavior (*int behav*). The *cont* element is composed of a set of contexts.

According to the key-value structure, level 4 represents the value(s) for each key present in level 3. Therefore, *cont* contains a set of sub-levels, where (i) each sub-level represents a specific context ($cont_n$), and (ii) each specific context stores the information in the 5W1H format to detail the context. We consider a set of contexts since each context represents a possible circumstance or situation.

The following set of tuples express the textual definition of our canonical model:

- $PI = (msg, int, effec)$, where *msg* represents a message, *int* represents intention, and *effec* represents effect;
- $msg = (action, cont)$, where *action* represents action and *cont* represents context;
- $action = (name, inputs, outputs, except, intbehav)$, where *name* represents the name, *inputs* represents the inputs, *outputs* represents the outputs, *except* represents the exceptions, and *int behav* represents the internal behavior;
- $name = \{na\}$, where *na* is the service name;
- $inputs = \{in_i\}$, $i = 1..j$, where in_i is the i-th input argument;
- $outputs = \{out_i\}$, $i = 1..j$, where out_i is the i-th output argument;
- $except = \{ex_i\}$, $i = 1..j$, where ex_i is the i-th exception;
- $intbehav = \{ib_i\}$, $i = 1..j$, where ib_i is the i-th service internal behavior;
- $cont = (cont_1, \ldots, cont_n)$, where $cont_n$ is the n-th context;
- $cont_n = (why_n, how_n, when_n, who_n, where_n, what_n)$;
- $why_n = \{wy_{in}\}$, $i = 1..k$, where wy_{in} is the i-th why feature of the n-th context;
- $how_n = \{h_{in}\}$, $i = 1..k$, where h_{in} is the i-th how feature of the n-th context;
- $when_n = \{we_{in}\}$, $i = 1..k$, where we_{in} is the i-th when feature of the n-th context;
- $who_n = \{wo_{in}\}$, $i = 1..k$, where wo_{in} is the i-th who feature of the n-th context;
- $where_n = \{wr_{in}\}$, $i = 1..k$, where wr_{in} is the i-th where feature of the n-th context;
- $what_n = \{wh_{in}\}$, $i = 1..k$, where wh_{in} is the i-th what feature of the n-th context;
- $int = \{i_i\}$, $i = 1..j$, where i_i is the i-th intention;
- $effec = (extbehav)$, where *extbehav* is the behavior performed by the receiver; and

- $extbehav = \{eb_i\}$, $i = 1..j$, where eb_i is the i-th external behavior that the receiver needs to perform to meet the intention.

Z notation is a formal language based on mathematical concepts for specifying computing systems. A Z schema has a name, a declaration part (i.e., static aspects), and a predicate part (i.e., dynamic aspects). We apply Z notation in the CAPITAL framework to facilitate the implementations of pragmatic interoperability. The next section presents the CAPITAL modeling in four different scenarios as a modeling and coding guide.

7.2 MODELING AND CODING GUIDE

The purpose of this exploratory study is to evaluate effectiveness, completeness, and mandatory elements of CAPITAL framework on scenarios with pragmatic interoperability.

We modeled our CAPITAL framework in four distinct scenarios. CAPITAL models classic and hypothetical scenarios. The scenarios describe problems in different domains, as follows:

- scenario 1 concerns a situation where an ambulance, hospital, or patient requests emergency, urgency, or normal laboratory test (ASUNCION et al., 2011);
- scenario 2 concerns a service that verifies the ancestry between two DNA sequences (NEIVA et al., 2016);
- scenario 3 concerns a car's Bluetooth that may avoid speakerphone calls when a undesired person is inside the car with the driver; and
- scenario 4 concerns smart security cameras that recognizes and reports the occurrence of new crimes.

Contextual elements vary depending on the scenario. Scenario 1 contains one contextual variable (*importance*) that can assume three values: *urgency*, *emergency*, and *normal*. Scenario 2 contains two contextual variables (*sequence* and *method*) and each variable can assume two values: sequence can be *DNA* or *RNA*; and method can be *local* or *global*. Scenario 3 contains two contextual variables (*distance* and *relationship*) and each variable can assume three values: distance can be *far*, *moderate*, or *near*; and relationship can be *unknown*, *known*, or *familiar*. Scenario 4 contains one contextual variable (*crimes*) that can assume four values: *child pornography*, *graffiti*, *stolen car*, and *person with firearm*.

The representation for each context can consider distinct 5W1H elements. For instance, the first context ($cont_1$) in scenario 1 (importance: emergency) uses *why*, *how*, *when*, *who*, and *what*, while first context ($cont_1$) in scenario 2 (sequence: DNA) uses only *who*, *what*, and *who*.

The scenarios aim to verify whether the framework answers the following research questions:

RQ1 *Does the CAPITAL framework model scenarios with pragmatic interoperability?* This question focuses on the *effectiveness* of our framework: CAPITAL may represent scenarios with pragmatic interoperability?

RQ1.1 *Does the framework consider all the elements necessary to represent the pragmatic interoperability of the scenarios?* This question focuses on the *completeness* of our framework: CAPITAL considers all the elements necessary to provide pragmatic interoperability?

RQ1.2 *What elements are needed to provide pragmatic interoperability?* This question focuses on the *mandatory elements* of our framework: CAPITAL considers all mandatory elements when it models pragmatic interoperability among systems?

RQ1.3 *The strategy chosen to model context was enough?* This question investigates whether the 5W1H format is capable of representing varied contexts.

The scenarios are presented in the following sections.

7.2.1 Scenario 1: Laboratory Test

The first scenario considers a hospital that requests a laboratory test and expects the laboratory to send the result at an emergency level, according to Figure 7.4 (ASUNCION et al., 2011).

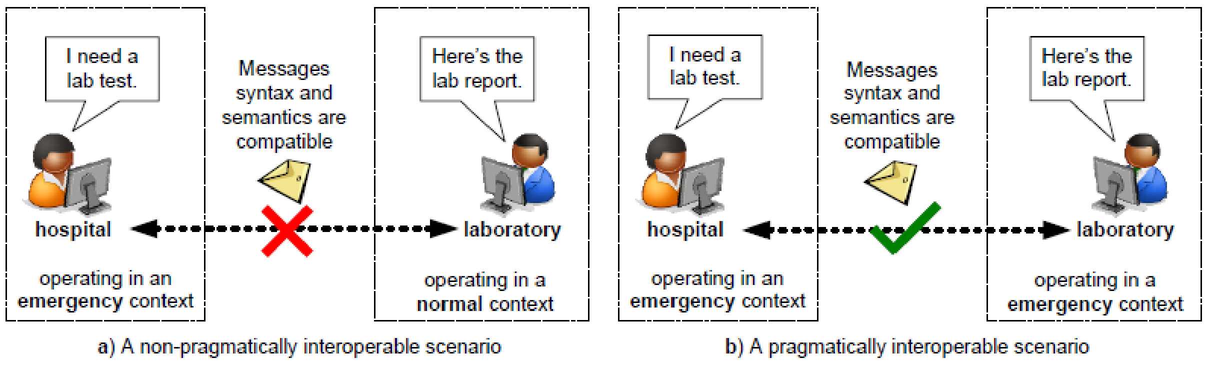

I need a lab test.
hospital
operating in an emergency context
Messages syntax and semantics are compatible
Here's the lab report.
laboratory
operating in a normal context
a) A non-pragmatically interoperable scenario
I need a lab test.
hospital
operating in an emergency context
Messages syntax and semantics are compatible
Here's the lab report.
laboratory
operating in a emergency context
b) A pragmatically interoperable scenario

Figure 7.4 Scenario of laboratory test. Figure 7.4(a) describes the scenario without pragmatic interoperability, while Figure 7.4(b) describes the pragmatic scenario. Adapted from (ASUNCION et al., 2011).

In Figure 7.4(a), the hospital intends to receive a test in an emergency context. However, the laboratory assumes the request as normal because the laboratory and hospital have a different understanding of the context. Consequently, the laboratory does not realize hospital intention. Despite performing syntactic and semantic interoperability, there is no pragmatic interoperability because the planned collaboration implicit in context were not performed.

In Figure 7.4(b), the laboratory accomplishes the hospital intention since both laboratory and hospital have the same understanding of the context. Pragmatic interoperability allows the laboratory to perform the request in the hospital context. In this case, syntactic, semantic, and pragmatic interoperability was achieved.

The initial step to represent this scenario in our canonical model is to structure the data necessary for the receiver to accomplish the sender's intention. This data (called **action**) may be represented as follows:

- **Name**: check test results
- **Internal behavior**: laboratory (i) receives sender's identifier, (ii) receives laboratory's employer identification number (EIN), (iii) receives client's identity document (ID), and (iv) returns available test
- **Inputs**: sender's identifier (literal), laboratory's EIN (numeric) and client's ID (numeric)
- **Outputs**: active client (boolean), available test (literal), and last updated (date)
- **Exceptions**: invalid sender, invalid laboratory, invalid client, inactive client, inactive service, and timeout.

We model the **context** based on the variables that can modify the sender's behavior. The laboratory test can be processed in *normal*, *urgency*, and *emergency* importance. We consider *importance* as **what** based on the answer to the following question: **what are the variable elements of the scenario?** $Cont_1$ models the conditions so that the importance is *emergency*; $Cont_2$ models *urgency*; and $Cont_3$ models *normal* importance. For instance, emergency ($Cont_1$) occurs when an ambulance (**who**) sends the request on the way to the hospital. In this case, the ambulance waits for a request test in two hours.

Figure 7.5 presents the canonical model for scenario 1. Contexts are presented separately in 7.5(a), 7.5(b), and 7.5(c) parts.

The effect expected by the sender (**int**) is to receive the test in emergency context. The receiver behavior (**ext behav**) is (i) get inputs, (ii) get contexts, and (iii) to return the laboratory test. Therefore, the context $Cont_1$ must be realized since this context meets the sender's intention: *laboratory test in the emergency context.*

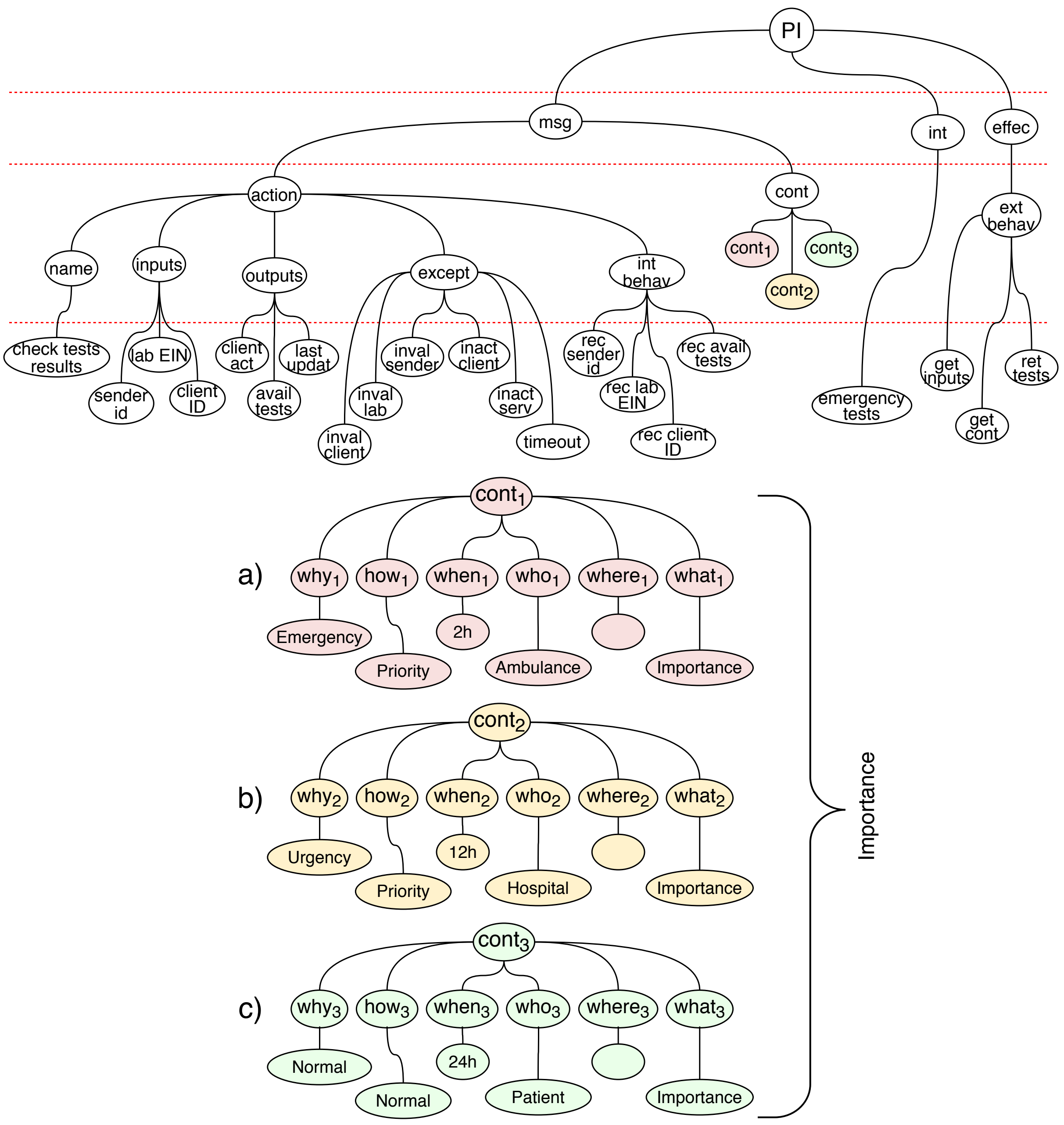


Figure 7.5 The canonical model for laboratory test scenario. Figure 7.5(a) represents emergency importance (in red), Figure 7.5(b) urgency (in yellow), and Figure 7.5(c) the normal importance (in green).

The following definitions list abstract data types in Z notation for our first scenario:

$[LabTest]$
$RetLogic ::= \text{‘}\texttt{yes}\text{’} \mid \text{‘}\texttt{no}\text{’}$
$why ::= \text{‘}\texttt{emergency}\text{’} \mid \text{‘}\texttt{urgency}\text{’} \mid \text{‘}\texttt{normal}\text{’}$
$how ::= \text{‘}\texttt{priority}\text{’} \mid \text{‘}\texttt{normal}\text{’}$
$when ::= \text{‘}\texttt{2h}\text{’} \mid \text{‘}\texttt{12h}\text{’} \mid \text{‘}\texttt{24h}\text{’}$
$who ::= \text{‘}\texttt{ambulance}\text{’} \mid \text{‘}\texttt{hospital}\text{’} \mid \text{‘}\texttt{patient}\text{’}$
$where ::=$
$what ::= \text{‘}\texttt{importance}\text{’}$

A sender can be an *ambulance*, *hospital*, or *patient*. Each sender has a *name*, an *id*, and data about *laboratory* and *client*. The *id* is mandatory and depends on the type of sender. The specification of a sender is given below.

```
Sender
  name? : who
  idSender? : seq₁ Char
  labEIN?, clientID? : ℕ
  ──────────
  if name = ‘ambulance’ then #idSender = 7
  if name = ‘hospital’ then #idSender = 9
  if name = ‘patient’ then #idSender = 11
  #labEIN = 9
  #clientID = 11
```

In this scenario, we have one contextual variable (*importance*) that can assume three values: *urgency*, *emergency*, and *normal*. Consequently, the context specification (*ImportanceContext*) contains three statements, a statement for each value that the variable can assume: *Emergency*, *Urgency*, and *Normal* importance.

```
ImportanceContext
  ΞEmergency
  ΞUrgency
  ΞNormal
  ──────────
```

Each context contains the following contextual elements: *why*, *how*, *who*, *where*, and *what*. The element *where* is not necessary. The value of each contextual element depends on the context. The following specifications show the context for *emergency*, *urgency*, and *normal* importance, respectively.

Emergency

whyEmerg : *why*
howEmerg : *how*
whenEmerg : *when*
whoEmerg : *who*
whatEmerg : *what*

whyEmerg = ‘`emergency`’
howEmerg = ‘`priority`’
whenEmerg = ‘`2h`’
whoEmerg = ‘`ambulance`’
whatEmerg = ‘`importance`’

Urgency

whyUrg : *why*
howUrg : *how*
whenUrg : *when*
whoUrg : *who*
whatUrg : *what*

whyUrg = ‘`urgency`’
howUrg = ‘`priority`’
whenUrg = ‘`12h`’
whoUrg = ‘`hospital`’
whatUrg = ‘`importance`’

Normal

whyNorm : *why*
howNorm : *how*
whenNorm : *when*
whoNorm : *who*
whatNorm : *what*

whyNorm = ‘`normal`’
howNorm = ‘`normal`’
whenNorm = ‘`24h`’
whoNorm = ‘`patient`’
whatNorm = ‘`importance`’

Given the context and sender, we can define service with pragmatic interoperability (*AttendanceWithPI*). The central idea is to capture the priority based on the sender's name (*Sender.name*). Afterwards, we perform the context for the sender type: ambulance, hospital, or patient.

```
AttendanceWithPI
    $\Xi Sender$
    $\Xi ImportanceContext$
    $priority : why$
    $activeClient! : RetLogic$
    $availableTest! : \mathbb{P}_1 \; LabTest$
    $lastUpdate! : Date$
    ----------------------------------------
    $\textbf{if } Sender.name? = ImportanceContext.Emergency.whoEmerg \textbf{ then}$
        $priority = ImportanceContext.Emergency.howEmerg$
    $\textbf{if } Sender.name? = ImportanceContext.Urgency.whoUrg \textbf{ then}$
        $priority = ImportanceContext.Urgency.howUrg$
    $\textbf{if } Sender.name? = ImportanceContext.Normal.whoNorm \textbf{ then}$
        $priority = ImportanceContext.Normal.howNorm$
```

Finally, exceptions can be modeled.

```
Exceptions
    $\Xi Sender$
    $\Xi AttendanceWithPI$
    $error! : RetLogic$
    ----------------------------------------
    $\textbf{if } Sender.name \notin who \textbf{ then } error! = \text{`yes'}$
    $\textbf{if } AttendanceWithPI.activeClient = \text{`no'} \textbf{ then } error! = \text{`yes'}$
```

The Z notation formalizes the canonical model of Figure 7.5. Although both approaches represent the same scenario, Z notation may facilitate coding since it is less abstract than the canonical model.

Regarding the RQ1, our results suggest that the CAPITAL framework represents the pragmatic interoperability between laboratory and hospital, as previously discussed.

About RQ1.1, our findings indicate that our framework considers the elements necessary to provide pragmatic interoperability in this laboratory test scenario.

In respect of RQ1.2, mandatory elements in this scenario are: one **name**, three **inputs**, three **outputs**, six exceptions (**except**), one internal behavior (**int behav**) composed of four steps, three contexts (**cont**), one intention (**int**), and one external behavior (**ext behav**) composed of three steps.

Concerning RQ1.3, we noticed that the 5W1H format modeled as expected three variations of sender behavior: *emergency*, *urgency*, and *normal* priority.

7.2.2 Scenario 2: DNA Ancestry

In this second scenario, we model a service that verifies DNA ancestry (NEIVA et al., 2016). Ancestry is defined by comparing any DNA sequence with a known DNA sequence, such as *Homo sapiens*. *Local* and *global* are examples of methods to verify DNA ancestry. The local method performs alignment on specific regions, and the global method performs alignment throughout the sequence. Consequently, the local method is useful for corrupted sequences or with different sizes, and the global method is useful for sequences with close sizes (COURONNE et al., 2003).

Syntactic interoperability occurs when, for instance, a system sends a string, and another system also expects a string. At this level, the string content is not evaluated. Semantic interoperability occurs when, for instance, a system sends a string with a DNA sequence, and another system also expects a string with a DNA sequence. At this level, systems only send and expect DNA sequences. Problems occur when a system expects to receive a DNA sequence and receives an RNA sequence. DNA sequence contains genetic instructions from an organism, while the RNA sequence is responsible for translating those instructions into proteins by regulating gene expression (CRICK, 1970).

Figure 7.6 describes how pragmatic interoperability is achieved in this scenario (adapted from (NEIVA et al., 2016)).

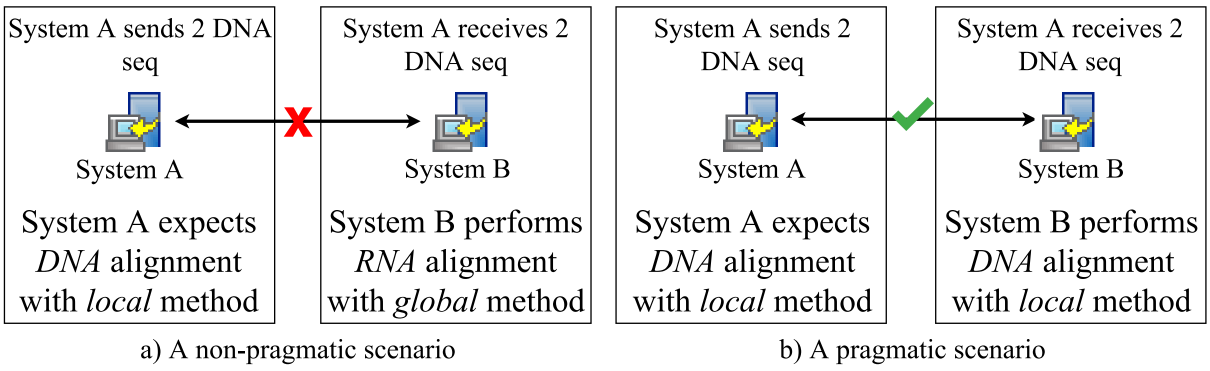

System A sends 2 DNA seq
System A
System A expects *DNA* alignment with *local* method
System A receives 2 DNA seq
System B
System B performs *RNA* alignment with *global* method
a) A non-pragmatic scenario
System A sends 2 DNA seq
System A
System A expects *DNA* alignment with *local* method
System A receives 2 DNA seq
System B
System B performs *DNA* alignment with *local* method
b) A pragmatic scenario

Figure 7.6 Scenario of DNA ancestry. Figure 7.6(a) describes the scenario without pragmatic interoperability and Figure 7.6(b) describes the pragmatic scenario. Adapted from (NEIVA et al., 2016).

In Figure 7.6(a), *system A* sends two *DNA sequences*. *System A* intends to receive the level of ancestry according to the *local method* because both sequences have different sizes. Nonetheless, *system B* assumes receiving *RNA sequences*. Consequently, *system B* performs an alignment with another method. *System B* does not realize *system A* intent since both systems have a different understanding of the context.

In Figure 7.6(b), *system B* performs the intent of *system A* since both systems have the same understanding of the context: *DNA alignment with the local method.*

In this scenario, we represent the **action** as follows:

- **Name**: verify ancestry
- **Internal behavior**: system B (i) receives two sequences (test and known), (ii) determines sequence types, (iii) determines method, and (iv) returns ancestry
- **Inputs**: test (literal) and known (literal) sequences
- **Outputs**: ancestry (boolean), confidence (numeric), and method (literal)

- **Exceptions**: invalid sequence, invalid type, inactive service, and timeout.

We model the **context** based on the number of variables that can modify the sender's behavior. The sequences can be *DNA* or *RNA*, and alignment method can be *local* or *global*. We consider *sequences* and *types* as **what** based on the answer to the following question: **what is the variable element of the scenario?** $Cont_1$ models the DNA sequence, $Cont_2$ models the RNA sequence, $Cont_3$ models the Local method, and $Cont_4$ models the Global method. Since this scenario has more than one contextual variable (*sequences* and *types*), the number of possible situations is the cartesian product of both contexts: (i) DNA alignment with local method ($Cont_1$ and $Cont_3$), (ii) DNA alignment with global method ($Cont_1$ and $Cont_4$), (iii) alignment of RNA with local method ($Cont_2$ and $Cont_3$), and (iv) alignment of RNA with global method ($Cont_2$ and $Cont_4$).

Figure 7.7 presents the canonical model for scenario 2. Contexts are presented separately in 7.7(a), 7.7(b), 7.7(c), and 7.7(d) parts.

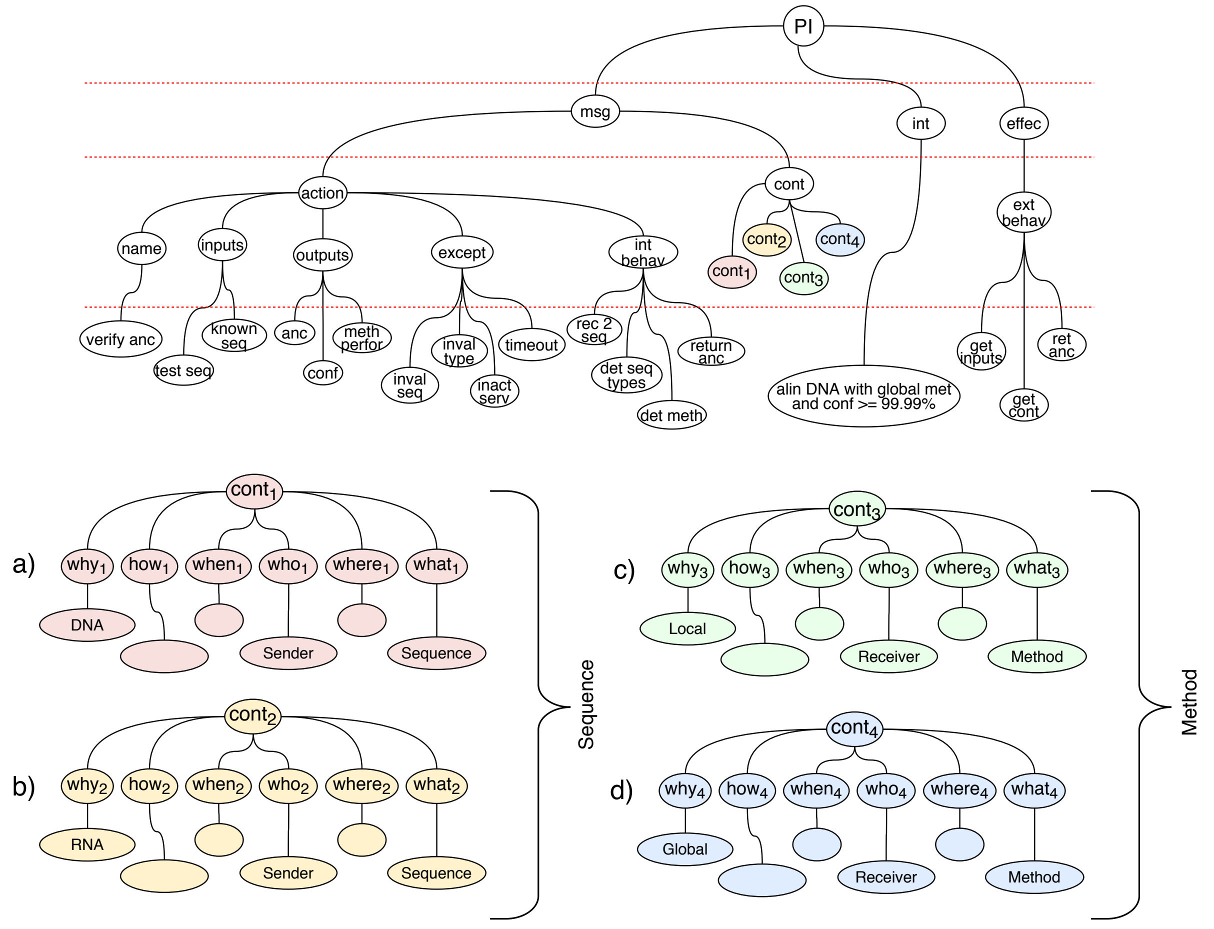


Figure 7.7 The canonical model for DNA ancestry scenario. Figure 7.7(a) represents the DNA sequence (in red) and Figure 7.7(b) represents the RNA sequence (in yellow); Figure 7.7(c) represents local method (in green) and Figure 7.7(d) represents global method (in blue).

$Cont_1$ models the conditions to sent a *DNA sequence*, and $Cont_2$ models the *RNA*.

$Cont_3$ models the requirements to use the *local method*, and $Cont_4$ models the *global method*. For instance, $Cont_1$ occurs when the sender (**who**) sends a *DNA sequence* (**why**), and $Cont_4$ occurs when the receiver (**who**) performs the *global method*.

The sender's intention (**int**) is to align DNA sequences with the *local* method with confidence greater than or equal to 99.99%. The receiver behavior (**ext behav**) is (i) get inputs, (ii) get contexts, and (iii) to return the ancestral level. The combination of $Cont_1$ and $Cont_4$ satisfies the sender's intention.

The following definitions list abstract data types in Z notation for our second scenario:

SequenceTypes ::= ‘`DNA`’ | ‘`RNA`’
MethodsTypes ::= ‘`Local`’ | ‘`Global`’
RetLogic ::= ‘`yes`’ | ‘`no`’
why ::= ‘`SequenceTypes`’ | ‘`MethodsTypes`’
how ::=
when ::=
who ::= ‘`Sender`’ | ‘`Receiver`’
where ::=
what ::= ‘`Sequence`’ | ‘`Method`’

Sender (*system A* in Figure 7.6) has two sequences (*test* and *known*), a *method*, and a *confidence level*. Both sequences are mandatory. The specification of a sender is given below.

Sender
$seqTest?, seqKnow? : \mathrm{seq}_1\ Char$
$method : methodsTypes$
$conf! : \mathbb{R}$

$\#seqTest? > 0$
$\#seqKnow? > 0$
$conf! \geq 99.99$ [test confidence level]

In this scenario, we have two contextual variables: *sequence* and *method*. The sequence can assume *DNA* and *RNA* values, and the method can assume *local* and *global* values. Each contextual variables (*SequenceContext* and *MethodContext*) contains two statements, a statement for each value that the variable can assume: *DNA* and *RNA* in *SequenceContext*, and *local* and *global* in *MethodContext*.

SequenceContext
$\Xi DNAseq$
$\Xi RNAseq$

MethodContext

$\Xi Local$
$\Xi Global$

Each context contains the following contextual elements: *why*, *who*, and *what*. The elements *how*, *when*, and *where* are not necessary in this scenario. The value of each contextual element depends on the context. The following specifications show the context for *DNA* and *RNA* sequence, respectively.

DNAseq

$whyDNA : why$
$whoDNA : who$
$whatDNA : what$

$whyDNA =$ ‘`DNA`’
$whoDNA =$ ‘`Sender`’
$whatDNA =$ ‘`Sequence`’

RNAseq

$whyRNA : why$
$whoRNA : who$
$whatRNA : what$

$whyRNA =$ ‘`RNA`’
$whoRNA =$ ‘`Sender`’
$whatRNA =$ ‘`Sequence`’

Similarly, the following specifications show the context for *local* and *global* methods, respectively.

Local

$whyLocal : why$
$whoLocal : who$
$whatLocal : what$

$whyLocal =$ ‘`Local`’
$whoLocal =$ ‘`Receiver`’
$whatLocal =$ ‘`Method`’

```
Global
  $whyGlobal : why$
  $whenGlobal : when$
  $whatGlobal : what$
  ----
  $whyGlobal = \text{‘}\texttt{Global}\text{’}$
  $whoGlobal = \text{‘}\texttt{Receiver}\text{’}$
  $whatGlobal = \text{‘}\texttt{Method}\text{’}$
```

Given the set of contexts and a sender, we can define one service with pragmatic interoperability (*ReceiverWithPI*). The idea is to (i) determine the type of sequences based on the nitrogen base of the sequence (DNA: A, T, C, and G; RNA: A, U, C, and G), (ii) determine the most appropriate method based on sequence sizes, and (iii) execute the context based on sequence and method.

```
ReceiverWithPI
  $\Xi SequenceContext$
  $\Xi MethodContext$
  $\Xi Sender$
  $typeSeqTest, typeSeqKnow : SequenceTypes$          [DNA or RNA]
  $method! : MethodTypes$
  $ancestry! : RetLogic$
  ----
  if $\text{‘}\texttt{T}\text{’} \in Sender.seqTest$ then
      $typeSeqTest = SequenceContext.DNAseq.whyDNA$
  else $typeSeqTest = SequenceContext.RNAseq.whyRNA$
  if $\text{‘}\texttt{T}\text{’} \in Sender.seqKnow$ then
      $typeSeqKnow = SequenceContext.DNAseq.whyDNA$
  else $typeSeqKnow = SequenceContext.RNAseq.whyRNA$
  if $lengthApprox(Sender.seqTest?, Sender.seqKnow?) = \text{‘}\texttt{yes}\text{’}$ then
      $method! = MethodContext.Local.whyLocal$
  else $method! = MethodContext.Global.whyGlobal$
```

The most appropriate method is defined as follows. Two sequences have close sizes when the difference between sizes is less than a constant integer value.

```
[X, Y]
$lengthApprox : X \times Y \to RetLogic$
$limit : \mathbb{N}$
------------------------
$limit : 1000$
$\forall x : X, y : Y \bullet$
    **if** $(\#x > \#y + limit) \lor (\#y > \#x + limit)$ **then**
        $lengthApprox(x, y) =$ ‘yes’
    **else**
        $lengthApprox(x, y) =$ ‘no’
```

Finally, exceptions can be formalized.

```
Exceptions
$\Xi ReceiverWithPI$
$error! : RetLogic$
------------------------
**if** $ReceiverWithPI.typeSeqTest \neq$ ‘DNA’ $\land$
    $ReceiverWithPI.typeSeqTest \neq$ ‘RNA’ **then** $error! =$ ‘yes’
**if** $ReceiverWithPI.typeSeqKnow \neq$ ‘DNA’ $\land$
    $ReceiverWithPI.typeSeqKnow \neq$ ‘RNA’ **then** $error! =$ ‘yes’
```

The Z notation formalizes the canonical model of Figure 7.7.

Regarding the RQ1, our results suggest that the CAPITAL framework represents the pragmatic interoperability between two different systems (*system A* and *system B*).

About RQ1.1, our findings indicate that our framework considers the elements necessary to provide pragmatic interoperability in a DNA ancestry scenario. For pragmatic interoperability to occur in this scenario, *system B* needs to (i) receive two sequences, (ii) confirm that both are DNA sequences, (iii) determine the most appropriate method for verifying ancestry, and (iv) return the degree of ancestry.

In respect of RQ1.2, mandatory elements in this scenario are: one **name**, two **inputs**, three **outputs**, four exceptions (**except**), one internal behavior (**int behav**) composed of four steps, four contexts (**cont**), one intention (**int**), and one external behavior (**ext behav**) composed of three steps.

Concerning RQ1.3, we noticed that the 5W1H format modeled as expected four variations of sender behavior: *DNA* and *RNA* sequences; and *local* and *global* methods.

7.2.3 Scenario 3: Bluetooth

Our third scenario considers a hypothetical situation: one car's Bluetooth should avoid speakerphone calls when a undesired person is inside the car with the driver.

We model two variables in this scenario: *different types of people* and *different distances from these people to the driver.* Figure 7.8 describes when pragmatic interoperability is achieved in this scenario.

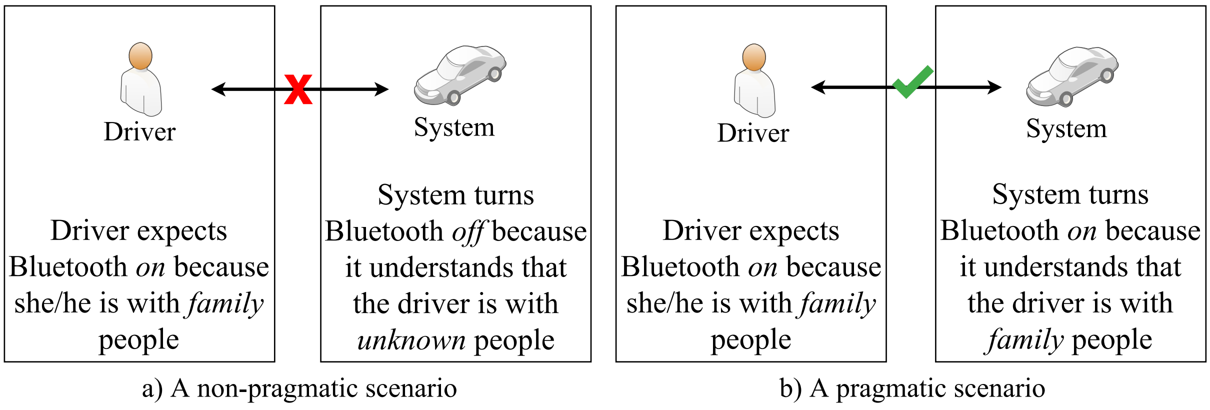

Driver
System
Driver expects Bluetooth *on* because she/he is with *family* people
System turns Bluetooth *off* because it understands that the driver is with *unknown* people
a) A non-pragmatic scenario
Driver
System
Driver expects Bluetooth *on* because she/he is with *family* people
System turns Bluetooth *on* because it understands that the driver is with *family* people
b) A pragmatic scenario

Figure 7.8 Scenario of the Bluetooth system. Figure 7.8(a) describes the scenario without pragmatic interoperability and Figure 7.8(b) describes the pragmatic scenario.

In this scenario, we represent the **action** as follows:

- **Name**: turn on Bluetooth
- **Internal behavior**: car's Bluetooth (i) determines the number of people, (ii) determines the relationship between the driver and the people, and (iii) decides Bluetooth status (on or off)
- **Output**: Bluetooth status (boolean)
- **Exceptions**: invalid relationship, inactive service, and timeout

We model the **context** based on the variables that can modify the sender's behavior. The model determines (i) the people near to the car and (ii) the relationship of these people with the driver. Since this scenario has more than one contextual variable (*distance* and *relationship*), the number of possible situations is the cartesian product of both contexts: (i) unknown far ($Cont_1$ and $Cont_4$), (ii) known far ($Cont_1$ and $Cont_5$), (iii) familiar far ($Cont_1$ and $Cont_6$), (iv) unknown at a moderate distance ($Cont_2$ and $Cont_4$), among others.

Figure 7.9 presents the canonical model for scenario 3. Contexts are presented separately in 7.9(a), 7.9(b), 7.9(c), 7.9(d), 7.9(e), and 7.9(f) parts.

$Cont_1$, $Cont_2$, and $Cont_3$ represent possible values for distance: *far*, *moderate*, and *near*. For instance, *far* (**why**) is a distance *greater than or equal to 100 meters* (**where**). Similarly, $Cont_4$, $Cont_5$, and $Cont_6$ represent possible values for *unknown*, *known*, and *familiar* relationships. For instance, the relationship *familiar* (**who**) represents someone in the *family* or a *friend*.

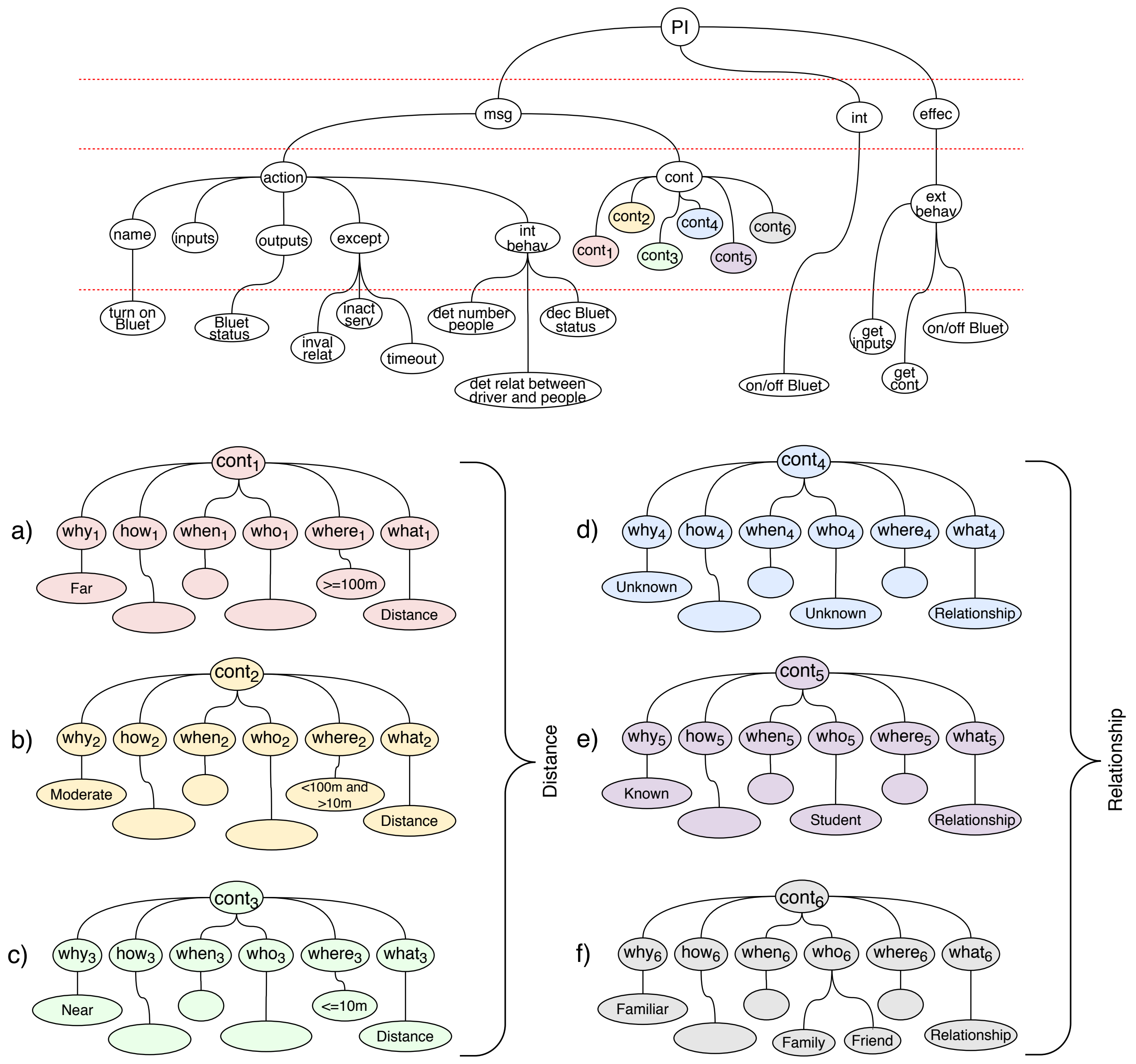


Figure 7.9 The canonical model for the Bluetooth scenario. Figure 7.9(a), Figure 7.9(b), and Figure 7.9(c) represent the far (in red), moderate (in yellow), and near (in green) distances, respectively. Figure 7.9(d), Figure 7.9(e), and Figure 7.9(f) represent the unknown (in blue), known (in purple), and familiar (in gray) relationships, respectively.

The following definitions list abstract data types in Z notation in this scenario:

BluetoothStatus ::= ‘`on`’ | ‘`off`’
DistanceTypes ::= ‘`Far`’ | ‘`Moderate`’ | ‘`Near`’
RelationshipTypes ::= ‘`Unknown`’ | ‘`Known`’ | ‘`Familiar`’
RetLogic ::= ‘`yes`’ | ‘`no`’
why ::= ‘`DistanceTypes`’ | ‘`RelationshipTypes`’
how ::=
when ::=
who ::= ‘`Unknown`’ | ‘`Student`’ | ‘`Friend`’ | ‘`Family`’
where ::= ‘`>=100m`’ | ‘`<100m and >10m`’ | ‘`<=10m`’
what ::= ‘`Distance`’ | ‘`Relationship`’

The Bluetooth system needs to establish the relationship between people and the driver to determine the Bluetooth status. The system determines the person’s distance to the car and the person’s relationship with the driver for each person.

Person
distCar : *DistanceTypes*
relationshipWithDriver : *who*

In this scenario, we have two contextual variables: *distance* and *relationship*. The *distance* can assume *far*, *moderate*, and *near* values, while the *relationship* can assume *unknown*, *known*, and *familiar* values.

DistanceContext
Ξ*Far*
Ξ*Moderate*
Ξ*Near*

RelationshipContext
Ξ*Unknown*
Ξ*Known*
Ξ*Familiar*

Context1, *Context2*, and *Context3* consider elements *why*, *where*, and *what*, while *Context4*, *Context5*, and *Context6* consider elements *why*, *who*, and *what*. The value of each contextual element depends on the context. The following specifications show the context for *far*, *moderate*, and *near* distances, respectively.

Far

whyFar : *why*
whereFar : *where*
whatFar : *what*

whyFar = ‘`Far`’
whereFar = ‘`>=100m`’
whatFar = ‘`Distance`’

Moderate

whyMod : *why*
whereMod : *where*
whatMod : *what*

whyMod = ‘`Moderate`’
whereMod = ‘`<100m and >10`’
whatMod = ‘`Distance`’

Near

whyNear : *why*
whereNear : *where*
whatNear : *what*

whyNear = ‘`Near`’
whereNear = ‘`<=10`’
whatNear = ‘`Distance`’

Similarly, the following specifications show the context for an *unknown*, *known*, and *familiar* relationship, respectively.

Unknown

whyUnkn : *why*
whoUnkn : *who*
whatUnkn : *what*

whyUnkn = ‘`Unknown`’
whoUnkn = ‘`Unknown`’
whatUnkn = ‘`Relationship`’

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Known}} \\ \hline
\mathit{whyKn} : \mathit{why} \\
\mathit{whoKn} : \mathit{who} \\
\mathit{whatKn} : \mathit{what} \\ \hline
\mathit{whyKn} = \text{‘}\texttt{Known}\text{’} \\
\mathit{whoKn} = \text{‘}\texttt{Student}\text{’} \\
\mathit{whatKn} = \text{‘}\texttt{Relationship}\text{’} \\ \hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Familiar}} \\ \hline
\mathit{whyFamil} : \mathit{why} \\
\mathit{whoFamil\_a}, \mathit{whoFamil\_b} : \mathit{who} \\
\mathit{whatFamil} : \mathit{what} \\ \hline
\mathit{whyFamil} = \text{‘}\texttt{Near}\text{’} \\
\mathit{whoFamil\_a} = \text{‘}\texttt{Familiar}\text{’} \\
\mathit{whoFamil\_b} = \text{‘}\texttt{Friend}\text{’} \\
\mathit{whatFamil} = \text{‘}\texttt{Relationship}\text{’} \\ \hline
\end{array}
$$

Given the set of contexts and a group of people, we can define a Bluetooth service with pragmatic interoperability (*BluetoothWithPI*). The main idea is to (i) determine the relationship between the driver and each person in the group, (ii) determine the ‘lowest’ relationship, and (iii) apply the contexts. The ‘lowest’ relationship is essential when there are people with different relationships: for instance, if an *unknown* and a *family* member accompany the driver, then Bluetooth must be turned off. The model performs this action because the *unknown* relationship is ‘lowest’ than the *family* relationship.

```
$BluetoothWithPI$
$\Xi Person$
$\Xi DistanceContext$
$\Xi RelationshipContext$
$totalPeople : \mathbb{N}$
$companionNear : \mathbb{P}_1\, Person$
$lowerRelationship : who$
$bluetStatus' : BluetoothStatus$
--------------------
$\forall people : People \bullet$
    $\textbf{if}\ people.distCar = DistanceContext.Near.whyNear\ \textbf{then}$
        $people \in CompanionNear$
$totalPeople = \#companionNear$
$\textbf{if}\ \exists\, p : companionNear \bullet$
    $p.relationshipWithDriver = Unknown.whoUnkn\ \textbf{then}$
        $lowerRelationship = RelationshipContext.Unknown.whyUnkn$
$\textbf{elseif}\ \exists\, p : companionNear \bullet$
    $p.relationshipWithDriver = Known.whoKn\ \textbf{then}$
        $lowerRelationship = RelationshipContext.Known.whyKn$
$\textbf{elseif}\ \exists\, p : companionNear \bullet$
    $p.relationshipWithDriver = Familiar.whoFamil\_a \lor$
    $p.relationshipWithDriver = Familiar.whoFamil\_b\ \textbf{then}$
        $lowerRelationship = RelationshipContext.Familiar.whyFamil$
$\textbf{if}\ lowerRelationship = RelationshipContext.Familiar.whyFamil\ \textbf{then}$
    $blueStatus' = \text{‘on’}$
$\textbf{else}$
    $blueStatus' = \text{‘off’}$
```

Finally, exceptions can be specified.

```
$Exceptions$
$\Xi Person$
$\Xi BluetoothWithPI$
$error! : RetLogic$
--------------------
$error! = \text{‘no’}$
$\textbf{if}\ Person.relationshipWithDriver \notin who\ \textbf{then}\ error! = \text{‘yes’}$
$\textbf{if}\ BluetoothWithPI.lowerRelationship \notin who\ \textbf{then}\ error! = \text{‘yes’}$
```

The Z notation formalizes the canonical model of Figure 7.9.

Regarding the RQ1, our results suggest that the CAPITAL framework represents the pragmatic interoperability between the driver and the car's Bluetooth system.

About RQ1.1, our finding evidence that CAPITAL framework considers the elements necessary to provide pragmatic interoperability in Bluetooth scenario. For pragmatic interoperability to occur in this scenario, the Bluetooth system needs to: (i) determine the number of people close to the car, (ii) determine the 'lowest' relationship between the people nearby and the driver, and (iii) decide the Bluetooth status.

In respect of RQ1.2, mandatory elements in this scenario are one **name**, one **output**, three exceptions (**except**), one internal behavior (**int behav**) composed of three steps, six contexts (**cont**), one intention (**int**), and one external behavior (**ext behav**) composed of three steps.

Concerning RQ1.3, 5W1H format represented as expected the six variations of sender behavior: *far*, *moderate*, and *near* distances; and *unknown*, *known*, and *familiar* relationships.

7.2.4 Scenario 4: Public Security Domain

The fourth scenario is related to the public security domain. We consider smart security cameras that recognize and report the occurrence of new crimes. After receiving a new crime, the system forwards the incident to the appropriate agency. Depending on the crime, the system sends the notification to the Federal Police of Brazil (PFB), Military Police of Bahia State (PMBA), or City Guard of Salvador city (GMSSA). Each agency acts according to its internal rules, i.e., outside the system scope.

Problems on pragmatic interoperability occur when, for instance, the security cameras report "person with a firearm" and "the person" is a police officer. We consider that there is no crime when a police officer carries a firearm. In this case, the system must understand the situation context to trigger the appropriate agency or not. *Uniform* and *location* are examples of elements that can assist in decision making.

Figure 7.10 describes how pragmatic interoperability is achieved in this scenario.

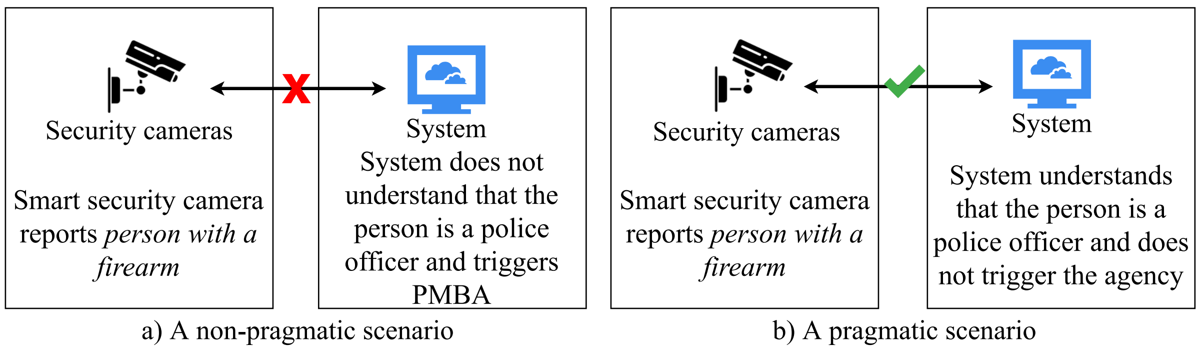

Security cameras
Smart security camera reports *person with a firearm*
System
System does not understand that the person is a police officer and triggers PMBA
a) A non-pragmatic scenario
Security cameras
Smart security camera reports *person with a firearm*
System
System understands that the person is a police officer and does not trigger the agency
b) A pragmatic scenario

Figure 7.10 Public security domain scenario. Figure 7.10(a) describes the scenario without pragmatic interoperability and Figure 7.10(b) describes the pragmatic scenario.

We represent the **action** as follows:

- **Name**: report a crime
- **Internal behavior**: system (i) receives the possible crime, (ii) determines if there is a crime, and (iii) triggers the appropriate agency

- **Input**: the crime (literal), date (date), time (time), and location (literal)
- **Output**: crime (boolean) and appropriate agency (literal)
- **Exceptions**: non-existent crime, inactive service, and timeout

We model the **context** based on four crimes and some contextual situations. We assume the system recognizes the following crimes: *child pornography*, illegal *graffiti*, *stolen car*, and *person with firearm*. These crimes are modeled according to contexts $Cont_1$, $Cont_2$, $Cont_3$, and $Cont_4$, respectively. Figure 7.11 presents the canonical model for scenario 4. Contexts are presented separately in 7.11(a), 7.11(b), 7.11(c), and 7.11(d) parts.

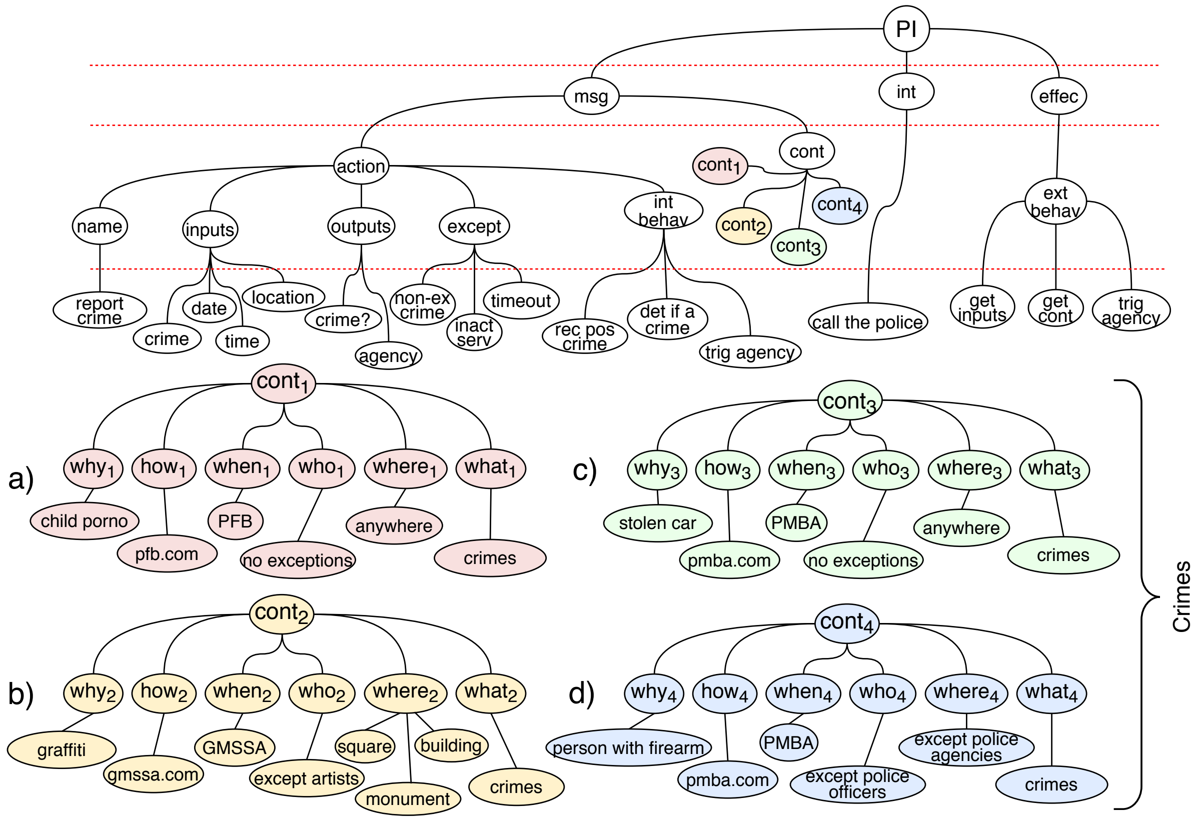


Figure 7.11 The canonical model for the public security domain scenario. Figure 7.11(a), Figure 7.11(b), Figure 7.11(c), and Figure 7.11(d) represent the crimes: child pornography (in red), graffiti (in yellow), stolen car (in green), and person with firearm (in blue), respectively.

Each context represents a crime. The model considers some contextual elements of each crime. For instance, *graffiti* (**why** in $Cont_2$) crime can occur in a *square*, *monument*, or *building* (**where**), except when *practiced by artists* (**who**). In this crime, the City Guard of Salvador city (**when**) must be called through the API *gmssa.com* (**how**). Similarly, *person with firearm* (**why** in $Cont_4$) crime can occur in anywhere, *except in police agencies* (**where**). We assume that this crime is not committed by police officers (**who**). In this crime, the Military Police of Bahia State (**when**) must be called through the API *pmba.com* (**how**).

We assume the Federal Police of Brazil (PFB) is responsible for the *child pornography* crime, the City Guard of Salvador city (GMSSA) is responsible for the *graffiti* crime, and the Military Police of Bahia State (PMBA) is responsible for *stolen car* and *person with firearm* crimes.

The following definitions list abstract data types in Z notation in this scenario:

isCrime ::= ‘`yes`’ | ‘`no`’
CrimesTypes ::= ‘`child porno`’ | ‘`graffiti`’ | ‘`stolen car`’ |
    ‘`person with firearm`’
RetLogic ::= ‘`yes`’ | ‘`no`’
why ::= ‘`CrimesTypes`’
how ::= ‘`pfb.com`’ | ‘`gmssa.com`’ | ‘`pmba.com`’
when ::= ‘`PFB`’ | ‘`GMSSA`’ | ‘`PMBA`’
who ::= ‘`no exceptions`’ | ‘`not artists`’ | ‘`not police officers`’
where ::= ‘`anywhere`’ | ‘`square`’ | ‘`monument`’ | ‘`building`’ |
    ‘`not police agencies`’
what ::= ‘`crimes`’

Smart security cameras sender one notification with *possible crime*, *date*, *time*, and *location*. The last three elements can be sent automatically depending on the device used in the notification. Other elements may be considered.

Notification
possCrime : *CrimesType*
date : *Date*
time : *Time*
location : seq *Char*

In this scenario, we have one contextual variable (*crimes*) that can assume four values: *child pornography*, *graffiti*, *stolen car*, and *person with firearm*.

CrimesContext
Ξ*Child_porn*
Ξ*Graffiti*
Ξ*Stolen_car*
Ξ*Person_w_fireman*

Each context contains the all contextual elements: *why*, *how*, *when*, *who*, *where*, and *what*. The value of each contextual element depends on the context. The following specifications show the context for *child pornography*, *graffiti*, *stolen car*, and *person with firearm* crimes, respectively.

Child_porn

whyChPorn : *why*
howChPorn : *how*
whenChPorn : *when*
whoChPorn : *who*
whereChPorn : *where*
whatChPorn : *what*

whyChPorn = ‘`child porno`’
howChPorn = ‘`pfb.com`’
whenChPorn = ‘`PFB`’
whoChPorn = ‘`no exceptions`’
whereChPorn = ‘`anywhere`’
whatChPorn = ‘`crimes`’

Graffiti

whyGraf : *why*
howGraf : *how*
whenGraf : *when*
whoGraf : *who*
whereGraf_a, *whereGraf_b*, *whereGraf_c* : *where*
whatGraf : *what*

whyGraf = ‘`graffiti`’
howGraf = ‘`gmssa.com`’
whenGraf = ‘`GMSSA`’
whoGraf = ‘`not artists`’
whereGraf_a = ‘`square`’
whereGraf_b = ‘`monument`’
whereGraf_c = ‘`building`’
whatGraf = ‘`crimes`’

Stolen_car

$whyStCar : why$
$howStCar : how$
$whenStCar : when$
$whoStCar : who$
$whereStCar : where$
$whatStCar : what$

$whyStCar =$ ‘`stolen car`’
$howStCar =$ ‘`pmba.com`’
$whenStCar =$ ‘`PMBA`’
$whoStCar =$ ‘`no exceptions`’
$whereStCar =$ ‘`anywhere`’
$whatStCar =$ ‘`crimes`’

Person_w_fireman

$whyPFire : why$
$howPFire : how$
$whenPFire : when$
$whoPFire : who$
$wherePFire : where$
$whatPFire : what$

$whyPFire =$ ‘`person with firearm`’
$howPFire =$ ‘`pmba.com`’
$whenPFire =$ ‘`PMBA`’
$whoPFire =$ ‘`not police officers`’
$wherePFire =$ ‘`not police agencies`’
$whatPFire =$ ‘`crimes`’

Given a set of contexts and one notification, we define public security service with pragmatic interoperability (*PublicSecurityWithPI*) as follows. The central idea is to capture the possible crime (*Notification.possCrime*) and check for divergence between the contextual elements of the notification (*data*, *time*, and *location*) and the context (*who* and *where*). If there are no divergences, the system triggers the most appropriate agency.

```
PublicSecurityWithPI
  $\Xi Notification$
  $\Xi CrimesContext$
  $isCrime? : isCrime$
  $apprAgency? : when$
  $crime : CrimesType$
  $local : where$
  ----------------
  $crime = Notification.possCrime$
  $local = Notification.location$
  $isCrime? =$ ‘yes’                                [Based on location only]
  if $crime = CrimesContext.Child\_porn.whyChPorn$ then
      if $local = CrimesContext.Child\_porn.whereChPorn$ then
          $apprAgency? =$ ‘PFB’
  elseif $crime = CrimesContext.Graffiti.whyGraf$ then
      if $local = CrimesContext.Graffiti.whereGraf\_a \lor$
         $local = CrimesContext.Graffiti.whereGraf\_b \lor$
         $local = CrimesContext.Graffiti.whereGraf\_c$ then
          $apprAgency? =$ ‘GMSSA’
  elseif $crime = CrimesContext.Stolen\_car.whyStCar$ then
      if $local = CrimesContext.Stolen\_car.whereStCar$ then
          $apprAgency? =$ ‘PMBA’
  elseif $crime = CrimesContext.Person\_w\_fireman.whyPFire$ then
      if $local = CrimesContext.Person\_w\_fireman.wherePFire$ then
          $apprAgency? =$ ‘PMBA’
  else
      $isCrime? =$ ‘no’
```

Finally, we specify the exceptions as follows.

```
Exceptions
  $\Xi Notification$
  $error! : RetLogic$
  ----------------
  if $Notification.possCrime \notin why$ then
      $error! =$ ‘yes’
  else
      $error! =$ ‘no’
```

The Z notation formalizes the canonical model of Figure 7.11. At the current stage, we evaluate crime based only on the reported location.

Regarding the RQ1, our findings suggest that our CAPITAL framework represents a very simple scenario of pragmatic interoperability in the public security domain. We are aware that we need to refine the contexts and consider other contextual elements in addition to location.

About RQ1.1, our results evidence that our framework considers the elements necessary to provide pragmatic interoperability. For pragmatic interoperability to occur in this scenario, the system needs to: (i) recognize the reported crime, (ii) determine the existence of the crime, and (iii) trigger the most appropriate agency.

In respect of RQ1.2, mandatory elements in this scenario are: one **name**, four **input**, two **output**, three exceptions (**except**), one internal behavior (**int behav**) composed of three steps, four contexts (**cont**), one intention (**int**), and one external behavior (**ext behav**) composed of three steps.

Concerning RQ1.3, 5W1H format represented as expected the four crimes.

7.2.5 Discussion

We evaluated the effectiveness, completeness, and mandatory elements of the CAPITAL framework. Our findings show that the CAPITAL framework represents pragmatic interoperability in heterogeneous scenarios. Our results suggest that our framework might be generalized to other scenarios. These results validate our RQ1.

Our framework represents the mandatory elements to provide pragmatic interoperability in each scenario. This result validates RQ1.1 and suggests that the elements in the CAPITAL framework might model other scenarios.

The heterogeneity of the scenarios emphasized the mandatory information. We suggest the framework mandatory elements based on the intersection of the results of each scenario. This assessment helps us to answer RQ1.2. Regarding the **action**, the **name** identifies the service, but not every system needs **inputs**, e.g., scenario 3. Since pragmatic interoperability provides communication among systems, at least one **output** is mandatory. *Timeout* and *inactive service* are examples of traditional **exceptions**. Some predefined sequences of steps (**int behavior**) illustrate what the receiver must do to send the **output** expected by the sender. The **context** represents all variables that can change the receiver behavior. Our framework assumes the existence of at least one variable and two contexts. In each context, CAPITAL considers mandatory only the **why** and **what**. *What* refers to the variable, and *why* are all possible variations of what. According to (MCCARTHY, 1993), the context cannot be described entirely because of its infinite dimension. The framework considers as mandatory the intention (**int**) and external behavior (**ext behav**). The external behavior is usually composed by (i) get inputs, (ii) get the context, and (iii) the action to be performed. Mandatory elements are bold in Figure 7.3.

Although our scenarios have validated RQ.1.3, other contextual representations can be employed in our framework. In this case, the novel representation must be shared between the receiver and the sender. Both canonical models and textual descriptions can guide conversion among context representation models.

The discussions in each scenario emphasize the relation between **context** and **intention**. Since the **context** describes possible variations of a scenario, we realize that pragmatic interoperability is achieved when the **context** encompasses a high number of situations present in the **intention**. In the Venn diagram, pragmatic interoperability is achieved when the intersection between **context** and **intention** increases. Figure 7.12 summarizes this relationship. In Figure 7.12(c), pragmatic interoperability occurs more frequently than in Figure 7.12(a).

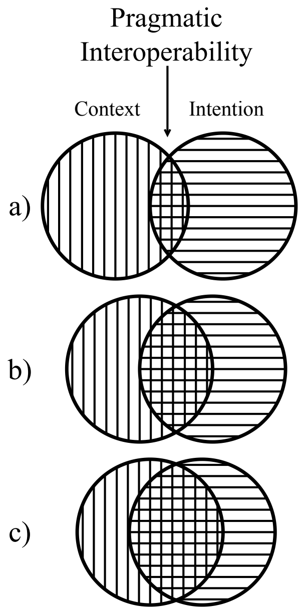


Figure 7.12 Venn diagram that relates context and intention. In Figure 7.12(c), pragmatic interoperability occurs more frequently than in Figure 7.12(a).

The sets are distinct because the context has infinite dimension (MCCARTHY, 1993), i.e., there is always an unmapped context contained only in the context set (not the intersection).

The next section presents the controlled experiment to evaluate CAPITAL framework understandability, completeness, consistency, conciseness, and performance.

7.3 CONTROLLED EXPERIMENT

This controlled experiment investigates the capacity of users to understand the definition and interpret scenarios with pragmatic interoperability with and without our framework. Based on the guidelines defined by Wohlin et al. (2012), this experiment is structured in four stages: *scope*, *planning*, *operation*, and *analysis and interpretation.*

7.3.1 Scope

The experiment scope defines the study goals and questions. In this stage, we define the goal in the Goal-Question-Metric (GQM) template (BASILI; ROMBACH, 1988), research question, and evaluation attributes of our experiment.

7.3.1.1 Goal. In this controlled experiment, we investigate the capacity of users to understand and interpret scenarios with pragmatic interoperability. According to GQM, the objective of this study is:

> Analyze <*the CAPITAL framework*> for the purpose of <*evaluation*> with respect to <*understandability, completeness, consistency, conciseness, and performance*> from the point of view of <*developers, professors, and students*> in the context of <*scenarios with pragmatic interoperability*>.

7.3.1.2 Research Question. Based on the goal, we defined the following research question:

> *Does the use of the CAPITAL framework influence the understanding and modeling of scenarios with pragmatic interoperability?*

Since there is no similar framework in the literature, we investigate whether there is a difference between (i) modeling scenarios with the CAPITAL framework and (ii) modeling scenarios only with the definition of pragmatic interoperability.

Our framework simplifies the pragmatic interoperability definitions from the literature since the CAPITAL framework considers our definition unified. We investigate whether there is a significant difference between modeling scenarios with and without our framework based on the attributes described in the next section.

7.3.1.3 Attributes. The comparison of models with and without CAPITAL framework is based on the following quality attributes:

- *understandability* (unders) evaluates the effort required to understand scenarios with pragmatic interoperability;
- *completeness* (compl) evaluates CAPITAL's ability to model scenarios with pragmatic interoperability;
- *consistency* (consist) evaluates the uniformity of the model generated by CAPITAL, i.e., whether CAPITAL's outcomes are uniform;
- *conciseness* (concis) evaluates the existence of unnecessary elements in the generated model. For instance, elements never used; and
- *performance* (perf) evaluates the time required to model scenarios with pragmatic interoperability.

We evaluate each attribute based on a questionnaire with nine questions available in Table 7.2. Each participant (specified in Subsection 7.3.2.3) assigned a score from 1 (very low) to 5 (very high) in each question. We measured each attribute by at least two questions, and then we consider the attribute evaluation as the average of the questions associated with them. The questionnaire is available in Appendix C.

Table 7.2 Questions used in the evaluation of attributes

	Attribute	Question
q_1	unders	What effort is required to understand the scenario?
q_2	compl	What effort is required to identify possible missing elements in the modeling?
q_3	consist	What effort is required to assess the uniformity of the elements in the modeling?
q_4	concis	What effort is required to identify unnecessary elements in modeling?
q_5	unders	What effort is required to understand and add a new context?
q_6	compl	What effort is required to assess the completeness of the modeling?
q_7	consist	What effort is required to identify conflicting or ambiguous elements in the modeling?
q_8	concis	What effort is required to identify redundant elements in modeling?
q_9	unders	What effort is required to list the intention, the message, and the effect of the scenario?

We perform the average (AVG) of understandability, completeness, consistency, and conciseness, respectively, as follow:

$$AVG\_unders = (q_1 + q_5 + q_9)/3 \tag{7.1}$$

$$AVG\_compl = (q_2 + q_6)/2 \tag{7.2}$$

$$AVG\_consist = (q_3 + q_7)/2 \tag{7.3}$$

$$AVG\_concis = (q_4 + q_8)/2 \tag{7.4}$$

We evaluate performance based on the time required to complete the task with and without the CAPITAL framework.

7.3.2 Planning

The planning stage describes the experiment conduct, i.e., how we will conduct the study. We present the design, materials, tasks, hypotheses, and participants.

7.3.2.1 Study Design. The design of our experiment comprises *one factor* with *two treatments*. The factor considered was *the modeling technique*, and the treatments were *CAPITAL framework* and *ad-hoc* (i.e., with and without CAPITAL framework). Therefore, we carried out this experiment based on two groups: *CAPITAL group* (CapG) and *Control group* (ConG). The *CAPITAL group* modeled pragmatic scenarios with the CAPITAL framework, and *Control group* modeled pragmatic scenarios only with pragmatic interoperability definitions from literature[2] (i.e., without the CAPITAL framework,

[2]The definitions are available in Table 7.1

ad-hoc). Each participant belongs only to one group.

The dependent variables are the attributes *understandability*, *completeness*, *consistency*, *conciseness*, and *performance*. Independent variable is the *modeling technique*. We vary the independent variable in two values: *CAPITAL framework* and *ad-hoc*.

7.3.2.2 Hypotheses. Based on the objective, research question, and attributes, we define five sets of null and alternative hypotheses. Each hypothesis corresponds to an attribute.

When true, a null hypothesis (H_0) indicates that there is no statistical difference between the two groups. When a null hypothesis is false, a true alternative hypothesis (Ha) indicates a statistical difference between them.

We divided each alternative hypothesis into two sub-hypotheses: the first (H_{a1}) indicates that the average for the *CAPITAL group* (CapG) is statistically more significant than the average for the *Control group* (ConG), and the other (H_{a2}) otherwise. We define the hypotheses as follows:

- ($H1$) Understandability
  - $H1_0$: the use of CAPITAL does not influence the understanding of scenario with pragmatic interoperability ($AVG\_unders_{ConG} = AVG\_unders_{CapG}$)
  - $H1_a$: the use of CAPITAL influences the understanding of scenario with pragmatic interoperability ($AVG\_unders_{ConG} \neq AVG\_unders_{CapG}$)
    - $H1_{a1}$: $AVG\_unders_{ConG} < AVG\_unders_{CapG}$
    - $H1_{a2}$: $AVG\_unders_{ConG} > AVG\_unders_{CapG}$
- ($H2$) Completeness
  - $H2_0$: the use of CAPITAL does not influence the modeling of a scenario with pragmatic interoperability ($AVG\_compl_{ConG} = AVG\_compl_{CapG}$)
  - $H2_a$: the use of CAPITAL influences the modeling of a scenario with pragmatic interoperability ($AVG\_compl_{ConG} \neq AVG\_compl_{CapG}$)
    - $H2_{a1}$: $AVG\_compl_{ConG} < AVG\_compl_{CapG}$
    - $H2_{a2}$: $AVG\_compl_{ConG} > AVG\_compl_{CapG}$
- ($H3$) Consistency
  - $H3_0$: the use of CAPITAL does not influence the standardization of scenarios with pragmatic interoperability ($AVG\_consist_{ConG} = AVG\_consist_{CapG}$)
  - $H3_a$: the use of CAPITAL influences the standardization of scenarios with pragmatic interoperability ($AVG\_consist_{ConG} \neq AVG\_consist_{CapG}$)
    - $H3_{a1}$: $AVG\_consist_{ConG} < AVG\_consist_{CapG}$
    - $H3_{a2}$: $AVG\_consist_{ConG} > AVG\_consist_{CapG}$

- ($H4$) Conciseness
  - $H4_0$: the use of CAPITAL does not influence the construction of complete scenarios with pragmatic interoperability ($AVG\_concis_{ConG} = AVG\_concis_{CapG}$)
  - $H4_a$: the use of CAPITAL influences the construction of complete scenarios with pragmatic interoperability ($AVG\_concis_{ConG} \neq AVG\_concis_{CapG}$)
    - $H4_{a1}$: $AVG\_concis_{ConG} < AVG\_concis_{CapG}$
    - $H4_{a2}$: $AVG\_concis_{ConG} > AVG\_concis_{CapG}$
- ($H5$) Performance
  - $H5_0$: the use of CAPITAL does not influence the time required to model scenarios with pragmatic interoperability ($AVG\_perf_{ConG} = AVG\_perf_{CapG}$)
  - $H5_a$: the use of CAPITAL influences the time required to model scenarios with pragmatic interoperability ($AVG\_perf_{ConG} \neq AVG\_perf_{CapG}$)
    - $H5_{a1}$: $AVG\_perf_{ConG} < AVG\_perf_{CapG}$
    - $H5_{a2}$: $AVG\_perf_{ConG} > AVG\_perf_{CapG}$

7.3.2.3 Participants. We sent a pre-questionnaire (Section C.1 in Appendix C) to potential participants aimed to collect their experience and status. Since our pre-questionnaire received 46 responses, we created two groups with 23 participants based on experience and status collected. We applied stratified random sampling (WOHLIN et al., 2012) to determine the group of each participants.

We recruited eight professors and 36 students related to different computer fields from two federal universities. Additionally, we recruited two professional developers with M.Sc. degree from two different companies. Among the 46 participants, four participants have a Ph.D. degree, six participants have a M.Sc. degree, and the others participants are undergraduate students.

We randomly allocated one developer, two professors with a Ph.D. degree, and two professors with an M.Sc. degree to each group. Regarding undergraduate students, we randomly allocated 18 students to each group. Since we invited students from two universities (U1 and U2), we randomly allocated 16 students from U1 and two students from U2 to each group. We also allocated the participants based on their experiences.

Although we do not require any prior knowledge, 82.6% of the participants do not know and have never used interoperability before, and 91.3% do not know and have never used pragmatic interoperability before. No participant was aware of our CAPITAL framework.

According to our design, we created two similar groups based on the participants' experience and status: *CAPITAL group* (CapG) and *Control group* (ConG). Section C.4 in Appendix C presents a participants overview involved in this controlled experiment. All participants agreed with the research objectives. We did not reward the participants and we guarantee the confidentiality of personal data.

7.3.3 Operation

The experiment operation phase describes how the treatments are applied to participants. We present the preparation, execution, and data validation steps.

7.3.3.1 Preparation. We invited students and professors from two different universities and two developers from two different companies in this phase. Along with the invitation, we present our research and the purpose of this controlled experiment.

After the pre-questionnaire answers, we group the participants into two groups. Each group received a specific questionnaire (Sections C.2 and C.3 in Appendix C). We make available and apply all questionnaires in electronic format.

Although we did not conduct a pilot study, two researchers with a Ph.D. degree validated our three questionnaires.

7.3.3.2 Execution. We provided the pre-questionnaire from 09/15/2020 to 09/22/2020 and the specific questionnaires for each group from 09/24/2020 to 09/30/2020.

The experiment was carried online and individually. We provided an e-mail with the questionnaires for eventual questions. We received two e-mails from the *Control group* (ConG) and one e-mail from the *CAPITAL group* (CapG).

Both groups consider two scenarios: Bluetooth and smartphone. The Bluetooth scenario is the same as that presented in Section 7.2.3. The smartphone scenario has the following description: the smartphone should prevent the device to change the screen setting to night mode during the day.

The experiment consisted of the following tasks:

- ConG: initially, each participant received definitions about pragmatic interoperability and the Bluetooth scenario described only textually. Then, they had to answer about the existence or not of pragmatic interoperability in the first scenario. After that, the group received the smartphone scenario and a set of questions about pragmatic interoperability in the scenario. Finally, the attributes were assessed. This questionnaire is available in Section C.2 (Appendix C).

- CapG: initially, each participant received our CAPITAL framework and the Bluetooth scenario described textually and modeled in CAPITAL. Then, they had to answer about the existence or not of pragmatic interoperability in the first scenario. After that, the group received the smartphone scenario and the blank CAPITAL to fill out the information. Finally, the attributes were assessed. This questionnaire is available in Section C.3 (Appendix C).

7.3.3.3 Data Validation. We did not detect any inconsistency in the answers. Therefore, we use all the answers in the data analysis.

7.3.4 Analysis and Interpretation

In this section, we present the analysis and interpretation of the results. This phase supports draw conclusions based on the collected data.

7.3.4.1 Results Overview. According to Appendix C, Part II of the questionnaires for each group verifies whether the participants identify pragmatic interoperability in the Bluetooth scenario with and without our CAPITAL framework. As a result, 60.9% of the *Control group* (14 participants) identify the scenario as pragmatic. Among these participants, two participants (14.3%) consider only the context to describe interoperability, three participants (21.4%) consider only the driver's intention, and the other participants (64.3%) describe pragmatic interoperability with other terms.

With our framework, 78.3% of the *CAPITAL group* (18 participants) identify pragmatic interoperability in the Bluetooth scenario. Among these participants, four participants (23.5%) consider only the context to describe interoperability, two participants (11.8%) consider only the driver's intention, four participants (23.5%) consider both context and intention, and the other participants (41.2%) describe pragmatic interoperability with other terms. This data suggests that our framework helps to understand pragmatic scenarios. Figure 7.13 details the answer by group.

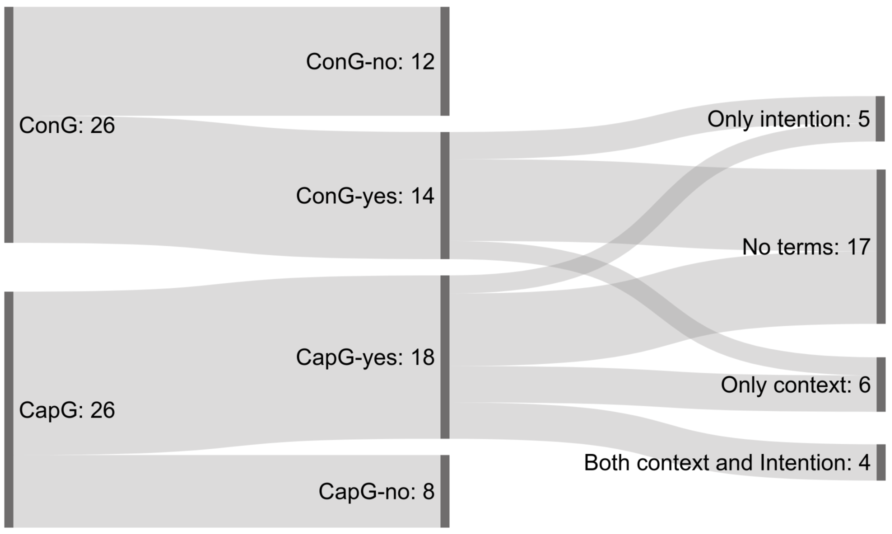


Figure 7.13 Summary of responses on identification and definition of pragmatic interoperability for the Bluetooth scenario.

The identification of pragmatic interoperability in the smartphone scenario (Part III of the questionnaires in Appendix C) follows the same pattern as in Part II. While 52.2% of the *Control group* (12 participants) identify the scenario as pragmatic, 73.9% of the *CAPITAL group* (17 participants) identify pragmatic interoperability in this scenario. The remaining questions in Part III aim to assist the questions in Part IV.

Table 7.3 presents all the data collected from Part IV of the questionnaires regarding the quantitative responses (Part IV of the questionnaires in Appendix C).

Table 7.3 Quantitative data collected by all 46 participants (23 participants by each group)

	Control group: ConG									*CAPITAL group*: CapG								
	unders			compl		consist		concis		unders			compl		consist		concis	
	q_1	q_5	q_9	q_2	q_6	q_3	q_7	q_4	q_8	q_1	q_5	q_9	q_2	q_6	q_3	q_7	q_4	q_8
p_1	4	2	4	5	3	5	2	3	5	2	3	4	3	3	1	1	2	2
p_2	4	3	4	5	4	4	3	3	3	2	5	2	2	1	3	2	3	3
p_3	3	3	3	3	2	4	3	3	3	1	2	2	2	2	2	2	2	2
p_4	4	5	4	4	5	3	5	5	5	1	1	2	3	2	2	3	4	2
p_5	3	3	4	3	3	4	4	3	3	3	2	2	2	4	2	1	3	5
p_6	4	2	4	5	4	2	5	1	5	1	1	2	1	1	2	4	2	5
p_7	4	5	5	5	4	3	3	5	5	2	2	2	2	2	4	2	3	5
p_8	4	3	4	4	4	3	3	5	4	3	1	2	3	1	1	1	2	3
p_9	4	5	5	4	5	2	5	5	5	3	3	1	3	2	1	1	2	2
p_{10}	3	3	3	3	3	2	3	3	3	1	4	2	4	4	4	2	4	4
p_{11}	4	4	4	4	5	2	2	4	4	1	5	2	3	5	4	2	5	5
p_{12}	1	4	4	3	5	4	2	5	5	2	3	2	3	3	2	3	1	5
p_{13}	2	4	4	4	4	4	2	3	5	2	1	2	1	1	2	1	1	1
p_{14}	1	3	5	3	3	2	2	1	2	4	3	3	2	4	4	3	4	3
p_{15}	5	2	3	3	3	1	2	3	3	5	5	5	3	5	4	2	5	5
p_{16}	2	4	3	2	4	3	4	3	2	2	2	4	3	2	1	2	2	2
p_{17}	4	5	3	5	5	2	2	5	3	2	3	2	3	3	4	3	3	3
p_{18}	4	5	4	5	4	5	5	5	3	1	5	3	2	3	1	1	3	2
p_{19}	2	4	3	3	4	3	2	1	1	3	3	2	3	4	5	3	4	3
p_{20}	1	2	5	3	4	3	5	1	2	2	3	4	4	3	1	2	4	3
p_{21}	1	3	3	3	4	4	4	3	3	2	5	4	3	4	2	2	2	2
p_{22}	1	5	5	4	5	5	4	4	3	4	4	4	3	5	1	3	4	2
p_{23}	4	5	5	4	4	4	4	4	4	3	3	1	3	2	1	2	4	1

The table is divided by participants ($p_1, p_2, \ldots, p_{23}$), by group (*Control group* and *CAPITAL group*), by attribute (**unders**tandability, **compl**eteness, **consist**ency, and **concis**eness), and by questions ($q_1, q_2, \ldots, q_9$). The questions associated with each attribute are defined according to Equations 7.1, 7.2, 7.3, and 7.4.

Table 7.4 presents the statistical measures of the collected data. We present the minimum value (min), lower quartile ($Q_{1/4}$), median (med), average (avg), upper quartile ($Q_{3/4}$), the maximum value (max), standard deviation (sd), and coefficient of variation (cv) for each attribute. The table is divided by attribute (**unders**tandability, **compl**eteness, **consist**ency, **concis**eness, and **perf**ormance) and by groups (*Control group*: ConG; and *CAPITAL group*: CapG). The values associated with the performance are given in minutes, except the coefficient of variation (cv).

Table 7.4 Statistical measures of the collected data

Attribute	Group	min	$Q_{1/4}$	med	avg	$Q_{3/4}$	max	sd	cv (%)
unders	*ConG*	2.33	3.00	3.33	3.54	4.00	4.67	0.66	18.80
	CapG	1.33	2.17	2.33	2.61	3.00	5.00	0.86	32.84
compl	*ConG*	2.50	3.25	4.00	3.87	4.50	5.00	0.69	17.95
	CapG	1.00	2.23	3.00	2.76	3.50	4.00	0.90	32.70
consist	*ConG*	1.50	2.75	3.50	3.26	4.00	5.00	0.88	26.91
	CapG	1.00	1.50	2.00	2.22	3.00	4.00	0.91	41.26
concis	*ConG*	1.00	3.00	3.50	3.46	4.00	5.00	1.14	32.90
	CapG	1.00	2.25	3.00	3.02	3.50	5.00	0.99	32.90
perf	*ConG*	6.00	9.00	13.00	13.43	17.00	23.00	4.91	36.52
	CapG	6.00	10.50	16.00	15.74	19.50	29.00	6.57	41.72

Regarding understandability (see Figure 7.14(a)), the average group suggests that understanding scenarios with the CAPITAL framework (avg = 2.61) is easier than understanding scenarios without the CAPITAL framework (avg = 3.54). However, the standard deviation (sd) and coefficient of variation (cv) of *CAPITAL group* (sd = 0.86 and cv = 32.85%) are greater than *Control group*'s values (sd = 0.66 and cv = 18.80%). These differences indicate that the *Control group*'s values had a smaller variation than the values of the *CAPITAL group*. This dispersion around the average may have influenced the increase in the *Control group*'s average value.

Regarding completeness (see Figure 7.14(b)), the average group suggests that modeling of scenarios is easier with the CAPITAL framework (avg = 2.76) than without the CAPITAL framework (avg = 3.87). However, the standard deviation (sd) and coefficient of variation (cv) of *CAPITAL group* (sd = 0.90 and cv = 32.70%) are also greater than *Control group*'s values (sd = 0.69 and cv = 17.95%). Similar to the previous attribute, this dispersion around the average may have influenced the increase in the *Control group*'s average value.

Regarding consistency (see Figure 7.14(c)), the average group suggests that pragmatic interoperability scenarios are more consistent with the CAPITAL framework (avg = 2.22) than without the CAPITAL framework (avg = 3.26). However, the standard deviation (sd) and coefficient of variation (cv) of *CAPITAL group* (sd = 0.91 and cv = 41.26%) are also greater than *Control group*'s values (sd = 0.88 and cv = 26.91%). Similar to the two previous attribute, this dispersion around the average may have influenced the increase in the *Control group*'s average value.

Regarding conciseness (see Figure 7.14(d)), the average group suggests that the construction of complete scenarios with pragmatic interoperability is slightly easier with the CAPITAL framework (avg = 3.02) than without the CAPITAL framework (avg = 3.46). The standard deviation (sd) and coefficient of variation (cv) of *CAPITAL group* (sd = 0.99 and cv = 32.90%) are very close to the *Control group* (sd = 1.14 and cv = 32.90%).

Finally, regarding performance (see Figure 7.14(e)), the groups' averages suggests a difference in the time required to model scenarios with pragmatic interoperability with the CAPITAL framework (avg = 15.74) and without the CAPITAL framework (avg = 13.43). The standard deviation (sd) and coefficient of variation (cv) of *CAPITAL group* (sd = 6.57 and cv = 41.72%) are greater than *Control group*'s values (sd = 4.91 and cv = 36.52%). These differences indicate that the time to model scenarios without the CAPITAL framework varied less than time to model scenarios with our framework.

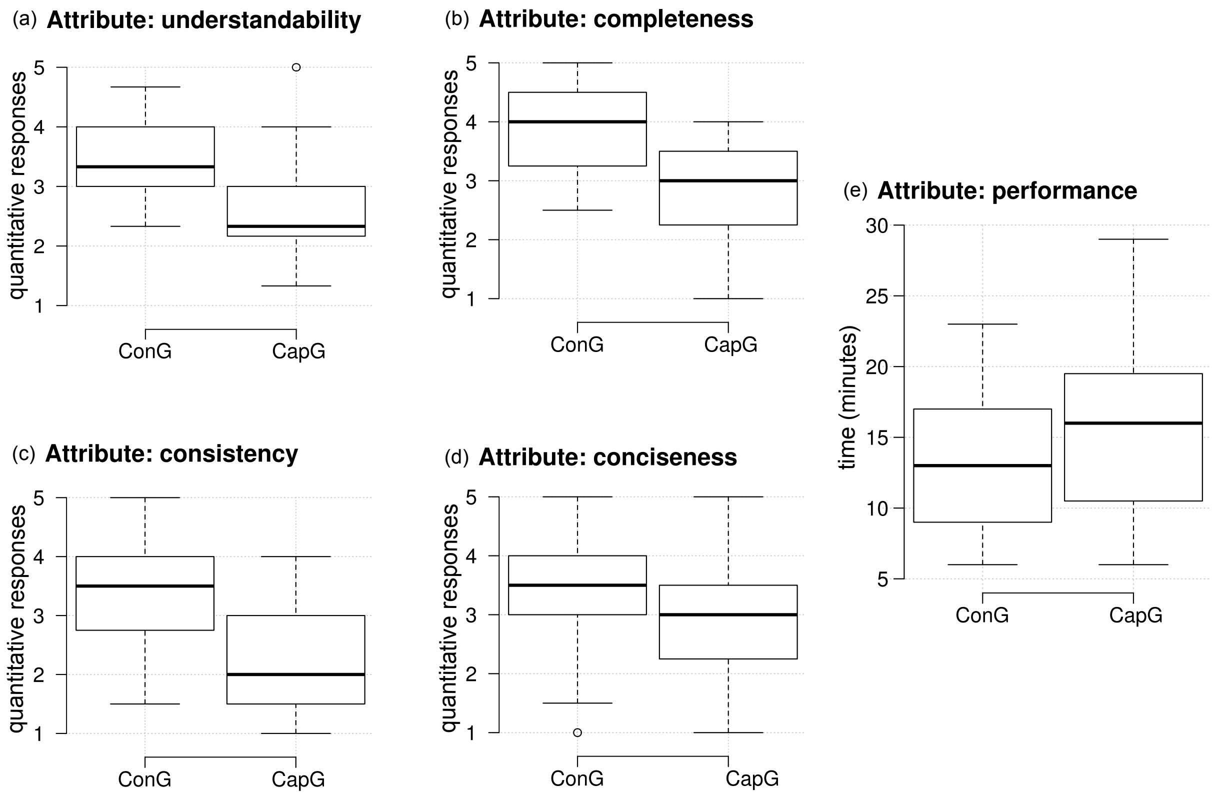


Figure 7.14 Comparison between the *Control group* (ConG) and *CAPITAL group* (CapG) for five attributes: (a) understandability, (b) completeness, (c) consistency, (d) conciseness, and (e) performance.

Generally, data suggest that the CAPITAL framework positively influenced the *understandability*, modeling (*completeness*), and standardization (*consistency*) of scenarios with pragmatic interoperability. By contrast, the CAPITAL framework negatively influenced the time required (*performance*) to model scenarios. We noted that our CAPITAL framework did not influence the construction of complete scenarios (*conciseness*). Next, we present and discuss a statistical analysis to confirm or refute the influence of our framework.

7.3.4.2 Hypothesis Testing. We perform the hypothesis test (WOHLIN et al., 2012) to investigate whether there is a significant difference between the data collected from the two groups based on attributes.

We apply the t-Student test (t-test) (HOTELLING et al., 1951) two-tailed for each attribute. T-test's alternative hypothesis is that there is a significant difference between the two sets of data. T-test has the following assumptions: (i) the samples are independent of each other, (ii) the samples follow normal distributions, and (iii) the samples have the same variance. Our samples are independent since they are associated with a specific group (control or CAPITAL) or attribute. We apply the Shapiro-Wilk test (RAZALI; WAH et al., 2011) to confirm that our samples are normal, and the Levene test (NORDSTOKKE; ZUMBO, 2010) to confirm that our samples have the same variance.

Table 7.5 presents the probability of the null hypothesis (p-value) for each evaluated attribute. We use the p-value to accept or reject each attribute's hypotheses with a significance level of $p = 0.05$ (i.e., 95% confidence level). Therefore, we may reject the null hypothesis (H_0) of attributes with $p < 0.05$.

Table 7.5 Probability for each evaluated attribute based on t-test (two-tailed)

Attribute	p-value
Understandability	0.000187
Completeness	0.000032
Consistency	0.000281
Conciseness	0.174552
Performance	0.185026

The test result showed that the null hypotheses of the attributes conciseness (p-value = 0.174552 > 0.05) and performance (p-value = 0.185026 > 0.05) cannot be rejected. These values show that there is no significant difference between the means of each group. As a consequence, we accept the following null hypotheses:

- $H4_0$ (about conciseness): the use of CAPITAL **does not influence** the construction of complete scenarios with pragmatic interoperability, i.e., $AVG\_concis_{ConG} = AVG\_concis_{CapG}$

- $H5_0$ (about performance): the use of CAPITAL **does not influence** the time required to model scenarios with pragmatic interoperability, i.e., $AVG\_perf_{ConG} = AVG\_perf_{CapG}$

This result shows no evidence that the CAPITAL framework influences the time required to model scenarios and the construction of complete scenarios.

However, the t-test showed that we can reject the null hypotheses of the attributes understandability (p-value = 0.000187 < 0.05), completeness (p-value = 0.000032 < 0.05), and consistency (p-value = 0.000281 < 0.05). These values show that there is a significant difference between the means of each group. As a consequence, we accept the following alternative hypotheses:

- $H1_a$ (about understandability): the use of CAPITAL **influences** the understanding of scenario with pragmatic interoperability, i.e., $AVG\_unders_{ConG} \neq AVG\_unders_{CapG}$

- H2$_a$ (about completeness): the use of CAPITAL **influences** the modeling of a scenario with pragmatic interoperability, i.e., $AVG\_compl_{ConG} \neq AVG\_compl_{CapG}$
- H3$_a$ (about consistency): the use of CAPITAL **influences** the standardization of scenarios with pragmatic interoperability, i.e., $AVG\_consist_{ConG} \neq AVG\_consist_{CapG}$

This result shows evidence that the CAPITAL framework influences the understanding, modeling, and standardization of scenarios with pragmatic interoperability.

7.3.4.3 Threats to Validity. This sections describes some threats to the validity of our controlled experiment. Additionally, we present our strategy to mitigate each threat.

Threats to **conclusion validity** concern the ability to relate treatment and experiment outcomes (WOHLIN et al., 2012). The *low statistical power* of a test may affect the decision to accept or reject a hypothesis. In our controlled experiment, we consider a robust and well-known test to assess our hypotheses. Moreover, we take care not to *violate any test assumptions*. We mitigate the threat of a *random heterogeneity of participants* based on the proper randomization of participants by treatments. We designate each participant to a group depending on their experience and status according to stratified random sampling.

Threats to **internal validity** regard the non-controlled factors that can affect the independent variable without the researcher's knowledge (WOHLIN et al., 2012). We argue that answering the online questionnaire without a predetermined time may mitigate the *maturation effect*. This strategy avoids the participants are affected negatively (e.g., tired or bored) over time. Regarding the *selection*, all participants answer the questionnaire voluntarily. We emphasized that there would be no rewards of any kind.

Threats to **construct validity** concern the generalization of the results (WOHLIN et al., 2012). *Evaluation apprehension* might have influenced the overall results. Some participants were probably afraid of being evaluated because we presented this experiment to students during an undergraduate course. To mitigate this threat, we informed that the controlled experiment was to evaluate only our framework. We also reported that there were no right or wrong answers. We mitigate the threat of *mono-method bias* by applying at least two questions for the same attribute. We argue that this strategy decreases the questionnaire's subjectivity.

Threats to **external validity** regard the ability to generalize the results (WOHLIN et al., 2012). The *interaction of selection and treatment* is our most massive threat. Although we applied our experiment with two developers, our participants are almost all students or professors. This sample may not be enough to generalize our results to a non-academic environment.

7.4 RELATED WORK

Tamani and Evripidou (2007) propose a pragmatic methodology for discovering web services in a specific domain. Authors summarize the problem of semantic web as the inability to select the most appropriate service from similar ones. This solution is based on human judgment to select the most suitable service automatically. As the semantic web is usually dynamic, these judgments can make the solution infeasible since the contexts change. Different from our approach, CAPITAL deals as many contexts as possible.

Neiva et al. (2016) introduce PRIME (Pragmatic Interoperability to Meaningful Collaboration), an architecture to support pragmatic interoperability in the collaborative development of scientific workflows. PRIME provides a mechanism for scientists to find services that meet their expectations. The architecture receives some search parameters, and it returns a list of services that satisfy the request. Although this search results in services ordered by relevance, contextual elements are not applied to compare two or more services.

Liu, Li and Liu (2014) propose a framework to interoperate data from a radiology department by a semantic and pragmatic perspective. Similar to other work, authors offer a solution for a specific domain, and they do not consider contextual elements when modeling the pragmatic model.

Lee et al. (2007) present a context-aware geospatial data and service integration framework based on the combination of syntactic, semantic, and pragmatic models. Although the authors use notions of context, pragmatic model considers only contextual aspects related to place and time. The absence of elements (e.g., why, who, and how) can limit the representation of other scenarios. Additionally, even if not explicit, authors consider that the pragmatic model has a single intention.

Most of these researches are domain-specific, i.e., each approach fulfills problems in a specific area. Similar to the definitions, there is no consensus on employing contextual nor intentional elements in solutions.

7.5 CHAPTER SUMMARY

This chapter described the CAPITAL framework, our model for pragmatic interoperability. Pragmatic interoperability enables systems to mutually affect each other's state and behavior so that the result produced by one participant matches the result expected by other participants.

Due to a lack of consensus, we performed a literature review on pragmatic interoperability definitions and related terms towards a unified definition. Based on this review, we presented our CAPITAL framework with canonical models, textual definition, and Z notation. We validated the CAPITAL framework in (i) four different scenarios and (ii) a controlled experiment. As discussed, our findings suggest that our CAPITAL framework positively influences the understanding, modeling, and standardization of scenarios with pragmatic interoperability.

The next chapter presents the pragmatic MIDAS architecture. We discuss the workings of pragmatic MIDAS, and we perform a proof of concept of our CAPITAL framework.

Chapter 8

This chapter presents and discusses MIDAS 3.0: middleware for pragmatic interoperability among cloud services. We evaluated MIDAS 3.0 based on our CAPITAL framework.

PRAGMATIC MIDAS ARCHITECTURE

As a proof of concept of our CAPITAL, we develop a prototype of the pragmatic MIDAS: MIDAS 3.0. Similar to the previous versions, MIDAS 3.0 intermediates communication between SaaS applications and heterogeneous data sources (e.g., DaaS and DBaaS) independently of API.

The next section describes and discusses the MIDAS 3.0 architecture.

8.1 MIDAS 3.0

MIDAS 3.0 is our first attempt to address pragmatic interoperability in cloud environments. MIDAS 3.0 recognizes (i) SQL and MongoDB (NoSQL) queries, (ii) data stored in a single or multiple DaaS/DBaaS providers, and (iii) queries with and without data join. Based on the CAPITAL framework, MIDAS 3.0 stores contextual information for each data source in the DIS. This information enables pragmatic MIDAS to understand the SaaS intent and access the appropriate source. MIDAS 3.0 is based on the previous versions since pragmatic interoperability requires syntactic and semantic levels (ASUNCION; SINDEREN, 2010).

We reuse functionalities of all modules and components of MIDAS 2.0 (MANE et al., 2020): *Request Module* (*Query Decomposer* and *Query Builder*), *Data Module* (*Data Mapping* and *Data Join*), *Semantic Module* (*Semantic Mapping* and *SMS*), *Result Module* (*Filtering* and *Formatter*), *DIS*, and *Crawler*. The reuse of the MIDAS 2.0 modules and components suggests that an appropriate syntax and semantic is necessary before implementing the pragmatic concept.

Additionally, we upgrade *Query Builder*, *Crawler*, and *DIS* components and we add in MIDAS 3.0 a new component: *Pragmatism Mapping*. Although SaaS is not within MIDAS scope, it requires adjustments since it must send contextual elements to pragmatic MIDAS. The encoding of this step depends on the device used in the query.

Figure 8.1 depicts MIDAS 3.0 architecture. The layered presentation illustrates the modules and components provided and reused by each level of interoperability.

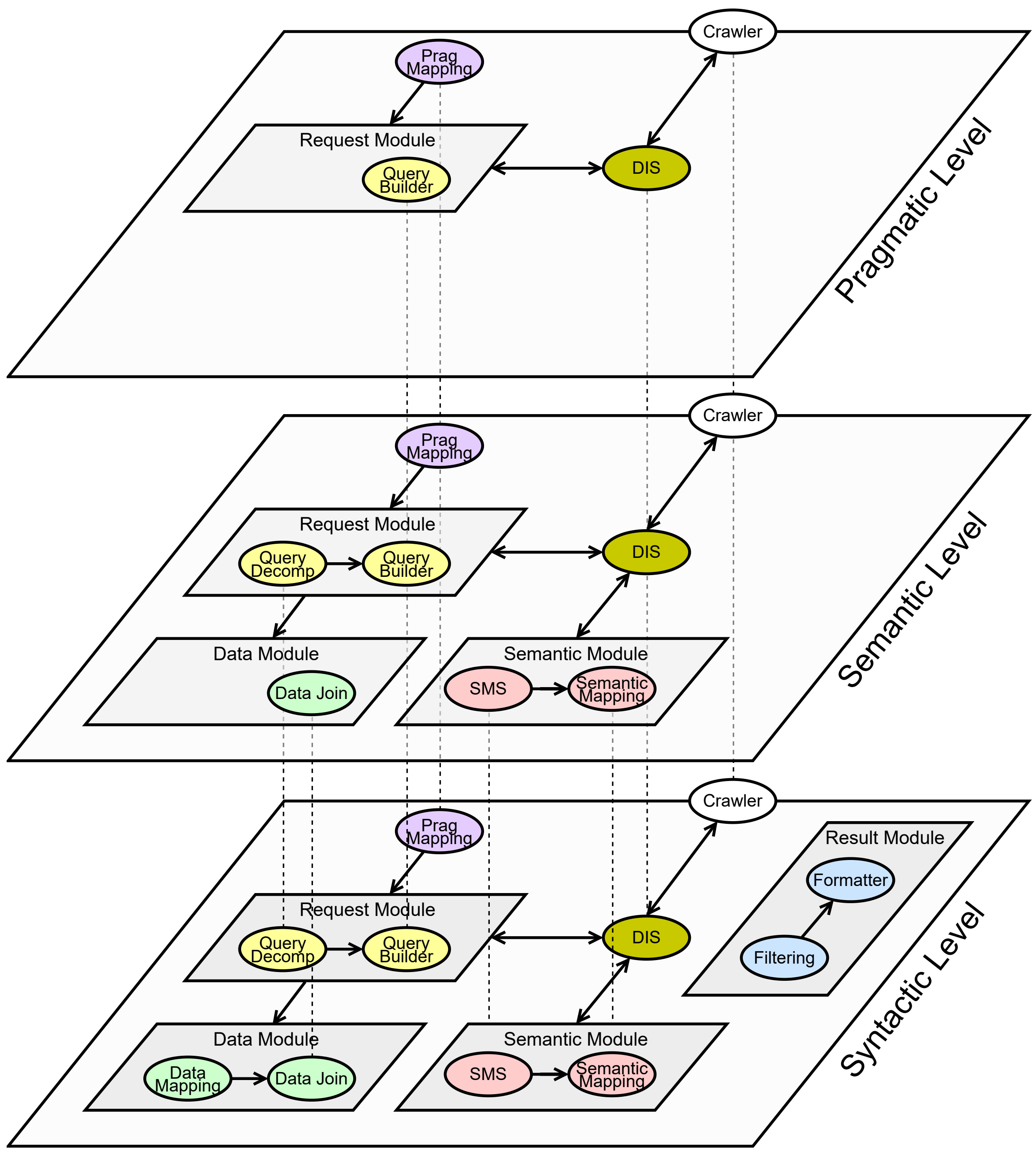


Figure 8.1 MIDAS 3.0 architecture.

Figure 8.2 provides a sequence diagram that describes the interaction between the user, SaaS, MIDAS 3.0, and providers. Initially, the user sends their query and return format. SaaS receives this data and gets the (i) user intention and (ii) contextual elements. We emphasize that the strategy or technology employed to capture these information is not part of this thesis's scope. Afterwards, SaaS sends to MIDAS: query, format, intention, and contextual elements. After receiving this data, MIDAS 3.0 performs the query based on intention and context. The query is performed, and then the result returns to MIDAS. If any source is a DBaaS, then MIDAS simulates a DBaaS as a DaaS, according to the *Data Mapping* component. If there is data join, then MIDAS will query more than one DaaS/DBaaS provider. Finally, MIDAS receives the data, processes it as expected, and then sends it to the SaaS provider.

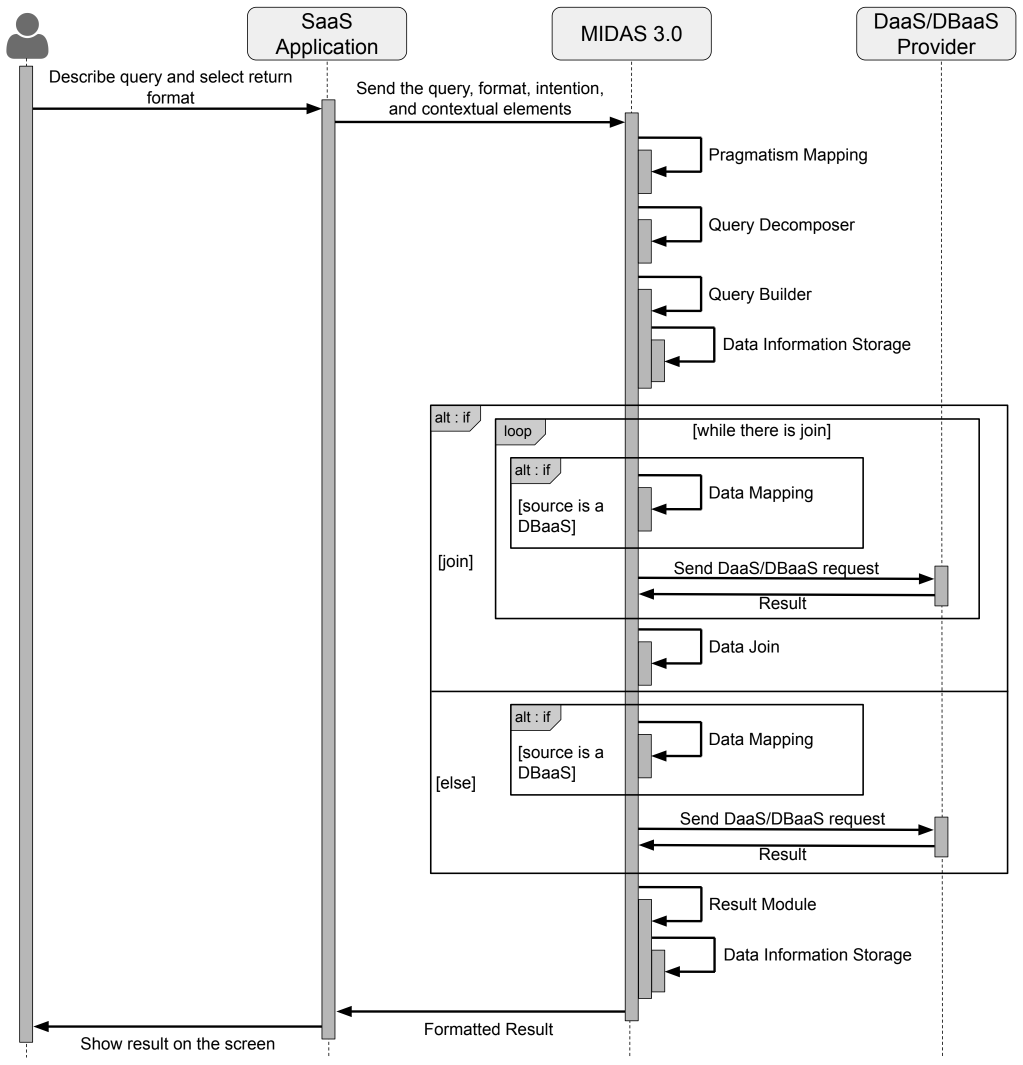


Figure 8.2 MIDAS 3.0 sequence diagram.

The next sections focus on the new and enhanced components of MIDAS 3.0.

8.1.1 Pragmatism Mapping

The *Pragmatism Mapping* component is responsible for (i) receiving the SaaS request, (ii) separating the query and format from pragmatic information, such as user intent and contextual elements, (iii) forwarding the query and format to *Query Decomposer*, and (iv) forwarding the pragmatic information to the *Query Builder*.

The component responsible for receiving the query in MIDAS 2.0 is the *Query Decomposer*. In previous version, SaaS sends only the query (in SQL or MongoDB) and the desired return format, such as XML, CSV, and JSON. Nonetheless, MIDAS 3.0 requires pragmatic information to facilitate the query in the source desired by the user, such as intention and contextual elements. Listing 8.1 illustrates a SaaS request. *Intention* and *contextual elements* are necessary information for MIDAS to provides pragmatic interoperability. MIDAS 2.0 does not consider these new pragmatic information.

Listing 8.1 Example of SaaS request

```
query: [SELECT blood FROM w7 WHERE id=10]
format: [JSON]
intention: [blood type of a patient within two hours]
contextual elements: [Date: 2020/12/09;
                      Time: 8:05 p.m.;
                      Requester: ambulance]
```

The *Query Decomposer* component decomposes queries and formats them as done by MIDAS 2.0. The *Query Builder* component builds a URL based on intention(s) and contextual elements. Other contextual elements can be captured or requested depending on the SaaS request, such as location.

8.1.2 Enhanced Components

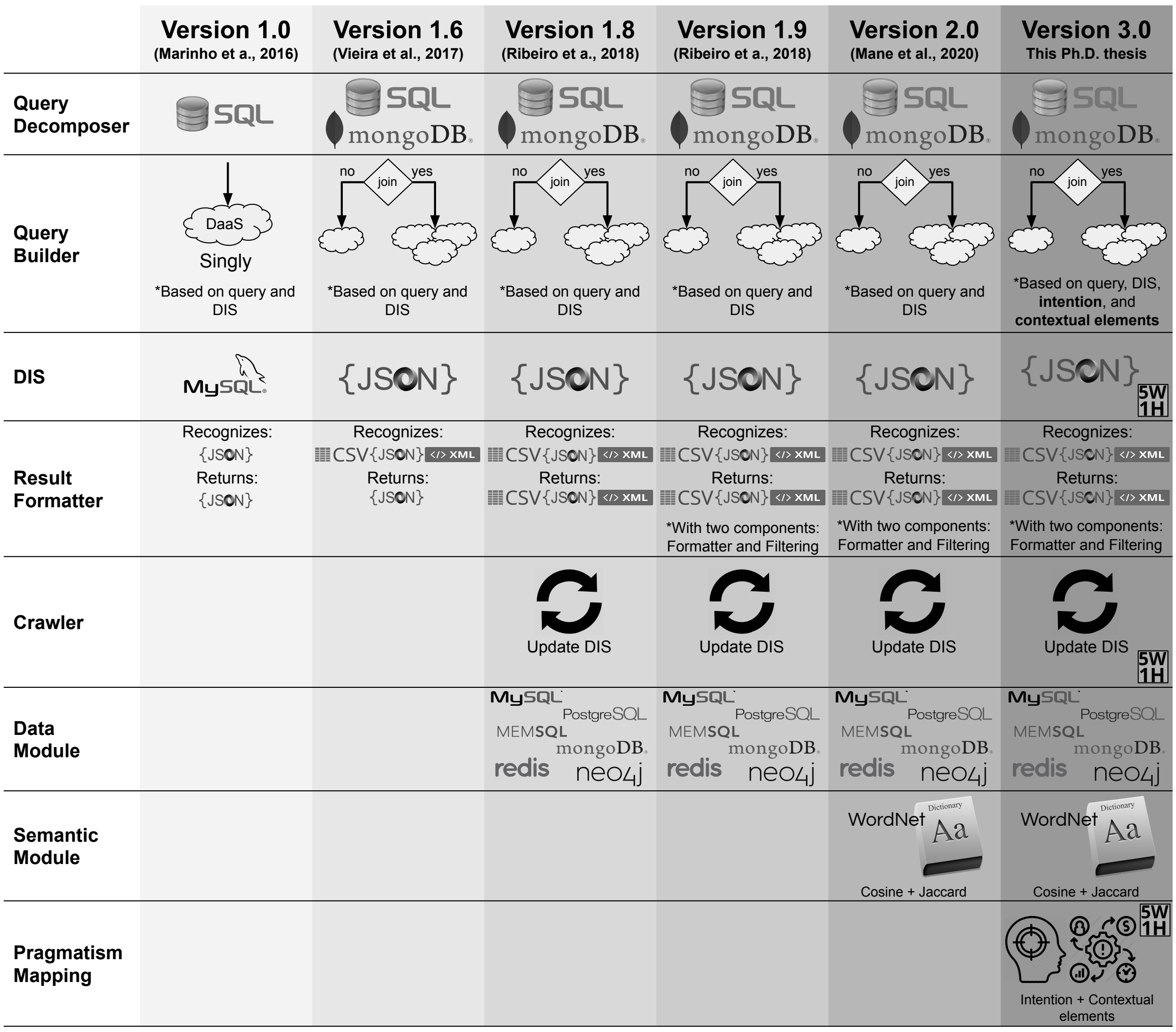
MIDAS 3.0 requires an update on three components: *Query Builder*, *Crawler*, and *DIS*. In MIDAS 3.0, the *Query Builder* assembles the URL to query DaaS based on (i) data received by the *Query Decomposer*, (ii) intention(s) and contextual elements received by the *Pragmatism Mapping*, and (iii) data about DaaS from the *DIS*. The *Query Builder* in MIDAS 2.0 considers only data received by the *Query Decomposer* and data about DaaS from the *DIS*. We notice that the *Query Builder* creates n URLs for a query with n intentions.

Additionally, the *Crawler* needs to search context information about each DaaS and to store this information into *DIS*. This new information enables the *Query Builder* to determine the correct data source based on user intention. We model the context about each DaaS with the 5W1H format. Based on the example in Section 7.2.1, the Listing 8.2 (in JSON syntax) presents fictitious data stored on *DIS* with two DaaS (*w7* and *vz*, lines 2–41) and three contexts (*ambulance*, *hospital*, and *patient*, lines 42–64). In this example, the key that identifies each context in the JSON file is the *who* element. Contexts are necessary for MIDAS to provide pragmatic interoperability.

Listing 8.2 Fictitious data stored on DIS

```
{
  "DaaS": {
    "w7": {
      "domain": "http://w7.com",
      "search_path": "/api/w/",
      "query_param": "qw",
      "filter_param": "flw",
      "sort_param": "sw",
      "limit_param": "lw",
      "dataset_param": "qsw",
      "records_param": "rcw",
      "fields_param": {
        "1": "id",
        "2": "age",
        "3": "firstname",
        "4": "lastname",
        "5": "blood"
      },
      "format_param": ".csv",
    },
    "vz": {
      "domain": "http://vz.com",
      "search_path": "/api/v/",
      "query_param": "qv",
      "filter_param": "flv",
      "sort_param": "sv",
      "limit_param": "lv",
      "dataset_param": "dsv",
      "records_param": "rcv",
      "fields_param": {
        "1": "ID",
        "2": "phone",
        "3": "borough",
      },
      "format_param": ".xml",
    }
  },
  "Context": {
    "ambulance": {
      "why": "emergency",
      "how": "priority",
      "when": 2,
      "where": "",
      "what": "importance",
    },
    "hospital": {
      "why": "urgency",
      "how": "priority",
      "when": 12,
      "where": "",
      "what": "importance",
    },
    "patient": {
      "why": "normal",
      "how": "normal",
      "when": 24,
      "where": "",
      "what": "importance",
    },
  }
}
```

Figure 8.3 depicts the main similarities and differences between MIDAS 3.0 and the previous versions.



	Version 1.0 (Marinho et a., 2016)	Version 1.6 (Vieira et al., 2017)	Version 1.8 (Ribeiro et a., 2018)	Version 1.9 (Ribeiro et a., 2018)	Version 2.0 (Mane et al., 2020)	Version 3.0 This Ph.D. thesis
Query Decomposer	SQL	SQL mongoDB	SQL mongoDB	SQL mongoDB	SQL mongoDB	SQL mongoDB
Query Builder	DaaS Singly *Based on query and DIS	no join yes *Based on query and DIS	no join yes *Based on query and DIS	no join yes *Based on query and DIS	no join yes *Based on query and DIS	no join yes *Based on query, DIS, **intention**, and **contextual elements**
DIS	MySQL	{JSON}	{JSON}	{JSON}	{JSON}	{JSON} 5W 1H
Result Formatter	Recognizes: {JSON} Returns: {JSON}	Recognizes: CSV {JSON} XML Returns: {JSON}	Recognizes: CSV {JSON} XML Returns: CSV {JSON} XML	Recognizes: CSV {JSON} XML Returns: CSV {JSON} XML *With two components: Formatter and Filtering	Recognizes: CSV {JSON} XML Returns: CSV {JSON} XML *With two components: Formatter and Filtering	Recognizes: CSV {JSON} XML Returns: CSV {JSON} XML *With two components: Formatter and Filtering
Crawler			Update DIS	Update DIS	Update DIS	Update DIS 5W 1H
Data Module			MySQL PostgreSQL MEMSQL mongoDB redis neo4j	MySQL PostgreSQL MEMSQL mongoDB redis neo4j	MySQL PostgreSQL MEMSQL mongoDB redis neo4j	MySQL PostgreSQL MEMSQL mongoDB redis neo4j
Semantic Module					WordNet Dictionary Aa Cosine + Jaccard	WordNet Dictionary Aa Cosine + Jaccard
Pragmatism Mapping						5W 1H Intention + Contextual elements

Figure 8.3 Similarities and differences between MIDAS 3.0 and the previous versions.

The next section presents the MIDAS 3.0 proof of concept. We implement the MI-

DAS 3.0 based on the example of Section 7.2.1. Our pragmatic MIDAS considers elements of CAPITAL framework aiming to provide pragmatic interoperability among cloud services.

8.2 PROOF OF CONCEPT

As a MIDAS 3.0 proof of concept, we implement the example from Section 7.2.1 into MIDAS middleware. This proof of concept is based on our CAPITAL to provide pragmatic interoperability between SaaS and DaaS/DBaaS. We assume that MIDAS recognizes two sources, i.e., two DaaS (*w7* and *vz*):

- DaaS *w7* concerns personal data about patients (attributes: *id*, *age*, *firstname*, *lastname*, and *blood*), and
- DaaS *vz* concerns phone and residence data of people (attributes: *ID*, *phone*, and *borough*).

Table A.1 and Table A.2 in Appendix A illustrate both DaaS. Data is fictitious.

In this proof of concept, SaaS requests data stored in a single DaaS (DaaS *w7*). Although the example (Section 7.2.1) is concerned with retrieving a patient's laboratory tests, we assume a query that requests a patient's blood type:

```
SELECT blood
FROM w7
WHERE id = 10
```

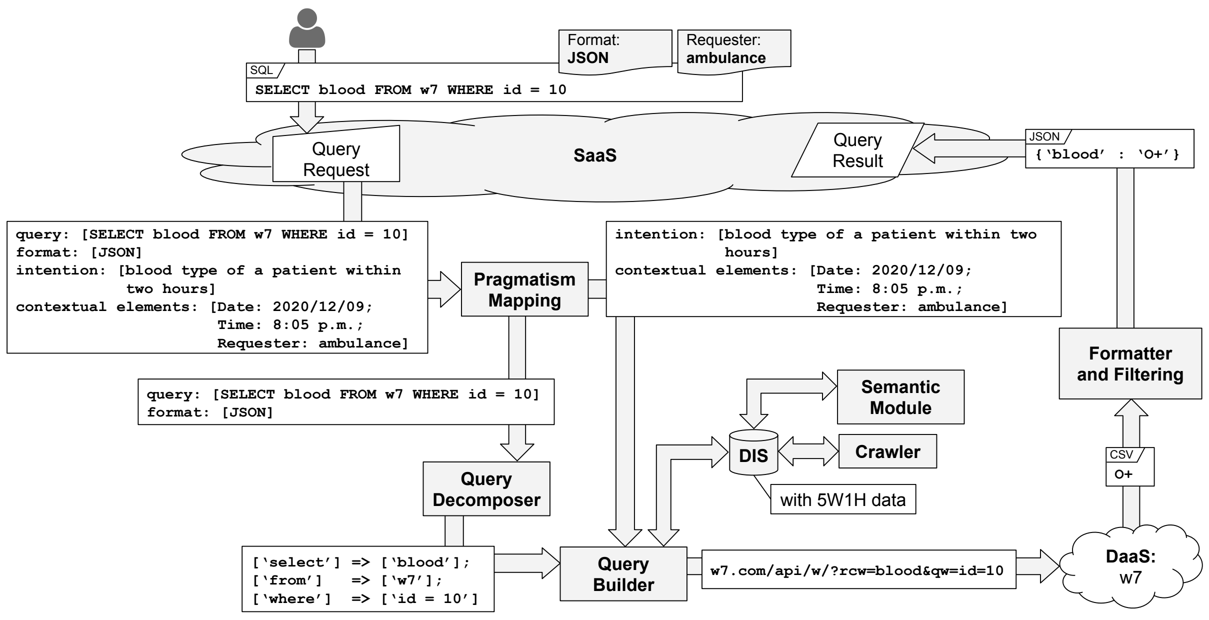
We assume that a query must be performed by an *ambulance*, a *hospital*, or a *patient*. Each requester has some specific characteristics, such as the time waiting for the return (*when* element). Listing 8.2 presents the *DIS* of this proof of concept. The *DIS* contains two DaaS and three contexts, one context for each requester. Each context is modeled based on the 5W1H template, and the key for each context is the *who* element.

Since the automatic detection of contextual elements is outside this thesis's scope, we set the contexts (Listing 8.2) and intentions, and we capture the requester (*ambulance*, *hospital*, or *patient*). The intention depends on the requester:

- the *ambulance*'s intention is to receive the result *within 2 hours*,
- the *hospital*'s intention is to receive the result *within 12 hours*, and
- the *patient*'s intention is to receive the result *within 24 hours*.

We have created one priority queue for each requester since the difference among the requester's intentions is the data return time. We take this approach only to perform a simple query and evaluate the MIDAS 3.0 architecture. However, we emphasize improving the capture of intentions and contexts is a relevant future work. Figure 8.4 depicts the MIDAS 3.0 execution sequence when an *ambulance* performs the above query.

In this execution, the user sends a SQL *query* by SaaS to MIDAS. Besides the *query* and *format*, the SaaS application sends the user *intention* and some *contextual elements*



```
SELECT blood FROM w7 WHERE id = 10
```

```
{'blood' : 'O+'}
```

```
query: [SELECT blood FROM w7 WHERE id = 10]
format: [JSON]
intention: [blood type of a patient within
            two hours]
contextual elements: [Date: 2020/12/09;
                      Time: 8:05 p.m.;
                      Requester: ambulance]
```

```
intention: [blood type of a patient within two
            hours]
contextual elements: [Date: 2020/12/09;
                      Time: 8:05 p.m.;
                      Requester: ambulance]
```

```
query: [SELECT blood FROM w7 WHERE id = 10]
format: [JSON]
```

```
['select'] => ['blood'];
['from']   => ['w7'];
['where']  => ['id = 10']
```

```
w7.com/api/w/?rcw=blood&qw=id=10
```

Figure 8.4 MIDAS 3.0 execution sequence.

to MIDAS, such as *date*, *time*, and *requester*. The *Pragmatism Mapping* (i) receives the SaaS request, (ii) forwards *query* and *format* to the *Query Decomposer*, and (iii) forwards the *intention* and *contextual elements* to the *Query Builder*. The *Query Decomposer* performs the query decomposition according to MIDAS internal structure and forwards the decomposed query to the *Query Builder*. Based on the decomposed query (sent by the *Query Decomposer*), *intention* and *contextual elements* (sent by the *Pragmatism Mapping*), and *DIS* (including contexts in 5W1H format), the *Query Builder* builds the request to DaaS *w7*. Finally, *Formatter* and *Filtering* format the return and forward the data to SaaS. Since there is no data join nor access to the DBaaS in this example, the *Data Module* is not triggered.

The next sections present the experiments and results of MIDAS 3.0.

8.2.1 Experiments

We provide a set of experiments to handle functional, execution time, overhead, and interoperability issues. These experiments investigate the communication among cloud services. We performed three experiments to evaluate our pragmatic MIDAS.

The first experiment (E_1) evaluates the overhead of our *Pragmatism Mapping* module in MIDAS middleware. For this, we submitted:

- 100 queries to one DaaS provider without our *Pragmatism Mapping* module, and
- 100 queries to one DaaS provider with our *Pragmatism Mapping* module.

In both tasks of E_1, we perform queries for a single DaaS, and we vary the number of records returned by 100, 1000, and 10000.

In the second experiment (E_2), we evaluate MIDAS middleware's correctness when receiving a query. For this, we submit 100 queries to MIDAS, randomly varying the requester: *ambulance*, *hospital*, or *patient*. After that, we check the correctness of the responses based on the context of each requester. In this experiment, we do not limit the number of records returned.

Finally, the third experiment (E_3) evaluates the effort to implement MIDAS 3.0 with dynamic pragmatic information, such as intention and context. In this experiment, we provide a function point estimate. The purpose of this experiment is to estimate the effort to receive pragmatic information from online sources.

Function Point Analysis is based on five metrics (or *information domain values*) (PRESSMAN, 2011):

- Number of internal logical files (ILFs): An ILF is a logical grouping of data within the boundaries of the software.
- Number of external interface files (EIFs): An EIF is a logical grouping of data useful to the system and kept outside the software boundary.
- Number of external inputs (EIs): An EI processes data or control information that comes from outside the software boundary. These entries are generally used to update ILFs.
- Number of external outputs (EOs): An EO sends data or control information outside the software boundary.
- Number of external inquiries (EQs): An EQ presents the user with data or information through simple retrieval.

We develop an application based on HTML, CSS, and JavaScript front-end, and PHP back-end to simulate a SaaS performing queries. This application is hosted on the Heroku[1] PaaS, and it can be accessed at `<http://pragmidas.herokuapp.com/test>`. Our cloud instance has one CPU core, 512 MB of RAM, and enough storage space for the experiments. We perform experiments E_1 and E_2 with PHP native functions. We do not use any external tool to avoid interference in the assessment.

The following DaaS was used to perform experiments E_1 and E_2: *Transportation Sites*[2]. This DaaS has 22,891 instances and 18 attributes.

8.2.2 Results and Discussion

This section presents and discusses our findings.

8.2.2.1 Experiment 1: Overhead. We classify the results of this experiment based on the value assigned to the limit clause. This value defines the number of records returned, and we restricted this clause to 100, 1000, and 10000 records returned. Internet

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

[2]https://data.cityofnewyork.us/Transportation/Transportation-Sites/hg3c-2jsy

measured 21.2 Mbps download and 8.2 Mbps upload before the experiment, and 25.7 Mbps download and 9.1 Mbps upload after the experiment.

Initially, we submitted the same query 100 times to return 100 data records. In this case, Figure 8.5 shows the average execution time:

- 0.1728 ± 0.0232 s for queries without our *Pragmatism Mapping* module, and
- 0.1939 ± 0.0196 s for queries with our *Pragmatism Mapping* module.

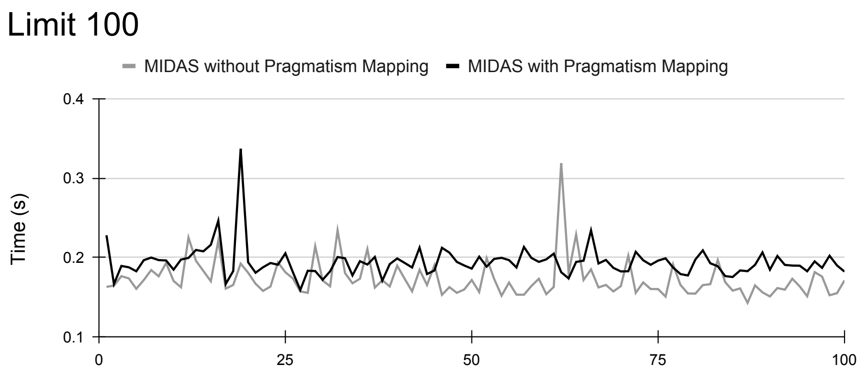


Figure 8.5 Return time (y-axis) for 100 queries (x-axis) with a limit of 100 records.

Afterward, we submitted the same query 100 times to return 1000 data records. In this case, Figure 8.6 shows the average execution time:

- 0.1975 ± 0.0415 s for queries without our *Pragmatism Mapping* module, and
- 0.2275 ± 0.0430 s for queries with our *Pragmatism Mapping* module.

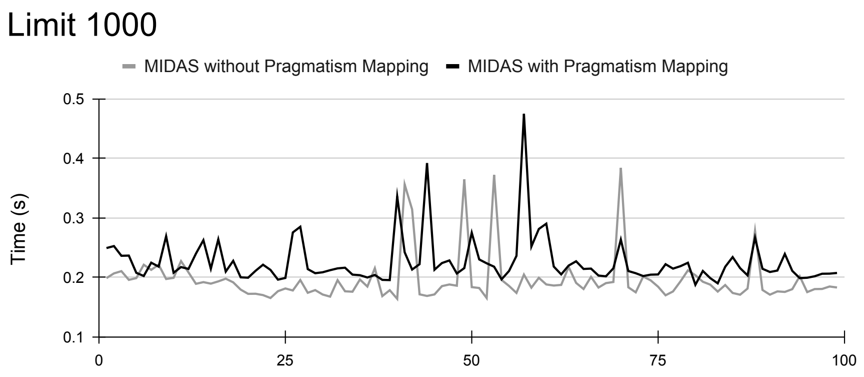


Figure 8.6 Return time (y-axis) for 100 queries (x-axis) with a limit of 1000 records.

Finally, we submitted the same query 100 times to return 10000 data records. In this case, Figure 8.7 shows the average execution time:

- 0.3435 ± 0.0597 s for queries without our *Pragmatism Mapping* module, and
- 0.4825 ± 0.0543 s for queries with our *Pragmatism Mapping* module.

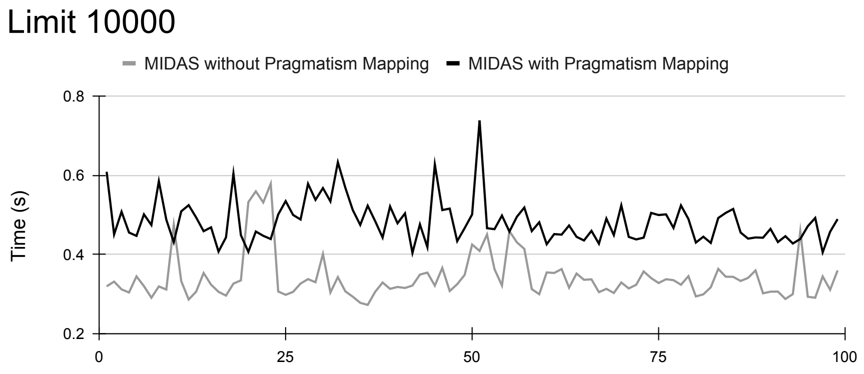


Figure 8.7 Return time (y-axis) for 100 queries (x-axis) with a limit of 10000 records.

Queries without our pragmatic module were on average faster than queries with our module: (i) 12.2% for queries with 100 returned data, (ii) 15.2% for queries with 1000 returned data, and (iii) 40.5% for queries with 10000 returned data.

The results show an overhead caused by the *Pragmatism Mapping* module of up to 40.5%. According to Ribeiro et al. (2018), the *Data Join* component is one of MIDAS middleware's most critical components because it aggregates results from distinct and heterogeneous providers. This component requires a certain time with or without our pragmatic module. The optimization of MIDAS algorithms is not part of this thesis's scope.

8.2.2.2 Experiment 2: Correctness. In this experiment, we assessed the pragmatic MIDAS's correctness. For this, we submit the same query to MIDAS 100 times, varying the requester in *ambulance*, *hospital*, or *patient*. In the end, we compared whether the context was selected correctly based on the context of the requesters.

Each requester had the following participation rate: 30% *ambulance*, 36% *hospital*, and 34% *patient*. As expected, MIDAS 3.0 correctly (i) selected 100% of contexts based on the requester and (ii) sent the results within the estimated deadline based on the requester's *when* element.

8.2.2.3 Experiment 3: Function Point. This experiment aims to estimate the effort required to design, implement, and test MIDAS 3.0 completely from online sources. Function Point Analysis is a standardized method to measure the functional size of the software. This metric derives from an empirical analysis based on the domain and complexity of the software. Although there are several methodologies (FREITAS JUNIOR; FANTINATO; SUN, 2015), we follow the guidelines presented by Pressman (2011). We conducted this experiment with the collaboration of two MIDAS developers.

This experiment considers the *type of count* as an enhancement project since we present MIDAS 3.0 based on previous versions.

The *counting scope* considers (i) the Pragmatism Mapping module and (ii) the three updated components in MIDAS 3.0: *Query Builder*, *Crawler*, and *DIS*. We consider that external users may (i) perform queries through a SaaS and (ii) receive data from DaaS. MIDAS is responsible for mediating the communication between SaaS and DaaS. Both SaaS and DaaS are external to MIDAS.

After defining the counting type and scope, we list MIDAS 3.0 ILFs, EIFs, EIs, EOs, and EQs:

- ILFs: (i) *DIS*, (ii) DBaaS configuration file, (iii) data return, and (iv) internal structure of MIDAS (count = 4);
- EIFs: none (count = 0);
- EIs: (i) type (DaaS or DBaaS), (ii) dataset, (iii) fields, (iv) where clause, (v) order by clause, (vi) limit clause, (vii) return format, (viii) intention, and (ix) contextual elements (count = 9);
- EOs: (i) send query result (count = 1); and
- EQs: (i) query a DaaS and (ii) query a DBaaS (count = 2).

After listing and classifying the domain information, we calculate the complexity of each counting group (Table 8.1). The value associated with each level of complexity is standardized, and it depends on subjective analysis (PRESSMAN, 2011).

Table 8.1 Calculation of function points

Information domain values	Count	Complexity factor				Total
		Simple	Average	Complex		
ILFs	4	$\times 7$	$\times 10$	$\times$**15**	=	60
EIFs	0	$\times 5$	$\times 7$	$\times 10$	=	0
EIs	9	$\times 3$	$\times 4$	$\times$**6**	=	54
EOs	1	$\times 4$	$\times$**5**	$\times 7$	=	5
EQs	2	$\times 3$	$\times$**4**	$\times 6$	=	8
				Total Count (TC)		127

The number of function points (FP) is given by the following equation:

$$FP = TC * [0.65 + 0.01 * \sum_{i=1}^{14} F_i] \tag{8.1}$$

where, FP is the total of function points, TC is the total count (Table 8.1), and F_i are the *value adjustment factors*. These factors are calculated based on 14 questions, where each question is measured with a factor ranging from $F_i = 0$ (i.e., question i is not important or it is not applicable) to $F_i = 5$ (i.e., question i is absolutely important). In this experiment, $\sum_{i=1}^{14} F_i = 46$. Individual measures by questions are available in Appendix D. The constant values are determined empirically (PRESSMAN, 2011). Therefore, according to Eq. 8.1, $FP = 127 * [0.65 + 0.01 * 46] = 140.97$.

MIDAS 3.0 with static pragmatic information (version used in experiments E_1 and E_2) has 9,155 non-comment lines of code. Considering the rate of 67 lines of code per function point in the PHP language[3], we estimate an increase of approximately 9,444.99 ($67 * 140.97$) lines of code so that MIDAS middleware performs pragmatic queries with online sources from intention and context. Therefore, we estimate MIDAS 3.0 with dynamic pragmatic information with 18,599 (9,155 + 9,444) lines of code.

Although this thesis is an advance in state of the art, we understand that this analysis illustrates the complexity of providing pragmatic interoperability among clouds.

8.3 CHAPTER SUMMARY

Our CAPITAL framework provides aspects that are not considered in the MIDAS 2.0, such as data that middleware must store and manage to provide pragmatic interoperability among cloud services. The absence of this information makes it difficult to provide pragmatic interoperability.

This chapter described the MIDAS 3.0, our first attempt to provide pragmatic interoperability among cloud services. We apply our CAPITAL framework as a coding guide to provide a version of MIDAS 3.0. Our findings suggest that (i) the overhead of the *Pragmatism Mapping* module is approximately 40%, (ii) pragmatic MIDAS correctly selects the requester's intent when the context and intentions are static, and (iii) providing pragmatic interoperability among clouds with dynamic context and intent is a complex task.

The next chapter presents the concluding remarks and future directions of this thesis.

[3]https://www.cs.helsinki.fi/u/taina/ohtu/fp.html

Chapter 9

This chapter presents our concluding remarks. We describe the contributions of this thesis and discuss future research directions.

CONCLUSION AND FUTURE WORK

Interoperability is the ability of heterogeneous systems to exchange and to use mutually exchanged information. Interoperability is generally described into syntactic, semantic, and pragmatic levels. Pragmatic interoperability enables systems to affect one another's state and behavior so that the produced result matches the expected result. In this investigation, our main efforts were mainly focused on investigating how to provide pragmatic interoperability among cloud services.

Our model for syntactic interoperability (Chapter 5) presents a lightweight formal description of MIDAS syntactic interoperability. We evaluated the correctness of this model based on syntactic interoperability between SaaS and DaaS.

Our model for semantic interoperability (Chapter 6) presents an ontology (*MIDAS-OWL*) to represent the MIDAS semantic interoperability. *MIDAS-OWL* provides a higher level of abstraction between service (SaaS) and data (DaaS) layers. We evaluated the consistency, acceptance, and correctness of MIDAS-OWL based on semantic interoperability among cloud services.

Finally, our model for pragmatic interoperability (Chapter 7) presents a CAPITAL. We investigated and introduced (i) the data needed to provide pragmatic interoperability among systems, (ii) a consensual definition for pragmatic interoperability, and (iii) a conceptual framework to represent pragmatic interoperability. We evaluated the CAPITAL framework in three ways: (i) running of four heterogeneous scenarios, (ii) controlled experiment, and (iii) proof of concept of MIDAS 3.0, version of pragmatic MIDAS.

Our investigation envisions the syntactic and semantic levels since these levels are required to provide pragmatic interoperability. To achieve all the objectives of this study, we present a model for each level of interoperability. This thesis's main objective was to investigate and provide a model for pragmatic interoperability among cloud services.

This thesis addresses an issue arising from an ongoing project: *Cloud Security Interoperable Society (CSIS)*, project number 8064/2015 funded by the Foundation for Research Support of the State of Bahia (FAPESB). CSIS aims to provide interoperability among different cloud services to mitigate the lack of interoperability in public security. This project intends to develop an infrastructure to provide syntactic, semantic, and pragmatic interoperability between SaaS and DaaS.

The following sections present the main contributions of this thesis, future research directions, and publications resulting from this work.

9.1 CONTRIBUTIONS

The main contributions of this thesis are listed as follows:

1. summary of the state of the art on interoperability and Cloud Computing (Chapters 2 and 3);
2. improvements in the MIDAS middleware to provides syntactic and semantic interoperability among cloud services (Chapter 4);
3. a detailed description of the operations performed by MIDAS syntactic interoperability (Chapter 5);
4. an ontology to represent the MIDAS semantic interoperability (Chapter 6);
5. a unified definition for pragmatic interoperability based on various definitions (Chapter 7);
6. mandatory elements to provide pragmatic interoperability (Chapter 7);
7. a conceptual framework capable of representing pragmatic interoperability among systems (Chapter 7); and
8. MIDAS architecture focused on pragmatic interoperability between SaaS and DaaS levels: MIDAS 3.0 (Chapter 8).

9.2 FUTURE RESEARCH DIRECTIONS

As future work, we plan to contribute to novel MIDAS improvements. These upgrades may suggest further improvements to the models previously presented. We list the main future research directions that arise from this thesis:

- *Adjust MIDAS-OWL to recognize queries in other languages.* We intend to adjust classes, properties of objects, and individuals of our ontology to recognize languages, e.g., NoSQL and SPARQL. We expect that no significant changes are necessary.
- *Include MIDAS-OWL into MIDAS.* We intend to add our ontology within MIDAS to evaluate the ontology overhead in real scenarios.
- *Apply CAPITAL framework in other scenarios.* We expect to model other scenarios to investigate the accuracy of our model in more detail.

- *Refine the public security scenario.* We intend to improve (i) our modeling based on a real environment and (ii) our contexts. Detail more the crimes may emphasize the need for contextual elements not considered.

- *Convert canonical models into other formats.* We intend to convert the canonical models of the model for pragmatic interoperability into formats such as XML or JSON to facilitate compression and automate possible changes to the context representation.

- *Improve the capture of intention and contexts.* We intend to improve the capture of intention and contexts aiming to facilitate pragmatic interoperability among cloud services since, in practice, the interaction is not explicit. This future work is related to improvements in the pragmatic MIDAS.

- *In-depth evaluation of pragmatic interoperability provided by MIDAS 3.0.* We intend to investigate the reasons that caused an approximately 40% overhead. This future work is also related to improvements in the pragmatic MIDAS.

9.3 PUBLISHED PAPERS

The following papers summarize the main contributions of this thesis:

- **RIBEIRO, E. L. F.**; VIEIRA, M. A.; CLARO, D. B.; SILVA, N. Transparent Interoperability Middleware between Data and Service Cloud Layers. In: *8th International Conference on Cloud Computing and Services Science (CLOSER)*. Funchal, Portugal: SCITEPRESS, 2018. p. 148–157. (RIBEIRO et al., 2018).

- **RIBEIRO, E. L. F.**; MONTEIRO, E. L.; CLARO, D. B.; MACIEL, R. S. P. A Conceptual Framework for Pragmatic Interoperability. In: *15th Brazilian Symposium on Information Systems (SBSI)*. Aracaju, Brazil: SBC, 2019. p. 1–8. (RIBEIRO et al., 2019).

- **RIBEIRO, E. L. F.**; VIEIRA, M. A.; CLARO, D. B.; SILVA, N. Interoperability between SaaS and Data Layers: Enhancing the MIDAS Middleware. In: MUÑOZ, V. M.; FERGUSON, D.; HELFERT, M.; PAHL, C. (Ed.). *Cloud Computing and Services Science (CCIS)*. Cham: Springer International Publishing, 2019. p. 102–125. (RIBEIRO et al., 2019).

- **RIBEIRO, E. L. F.**; SOUZA, M.; CLARO, D. B. *Towards an Ontology for Interoperability between Data and Service Cloud Layers.* Under review by International Conference on Cloud Computing and Services Science (CLOSER) 2021.

- **RIBEIRO, E. L. F.**; MONTEIRO, E. L.; CLARO, D. B.; MACIEL, R. S. P. *CAPITAL: A Conceptual Framework for Pragmatic Interoperability.* Under review by Information and Software Technology (IST) Journal.

- **RIBEIRO, E. L . F.**; CLARO, D. B.; MACIEL, R. S. P. *Comparison and Choice of Computational Architectures Based on Cost-Value Approach.* Under review by Brazilian Journal of Information Systems (iSys).
- **RIBEIRO, E. L . F.**; JESUS, L. E. N.; CLARO, D. B.; MOURA, N. S. *Towards a Pragmatic Interoperability on the MIDAS Middleware.* Under review by Brazilian Symposium on Information Systems (SBSI) 2021.
- **RIBEIRO, E. L . F.**; CLARO, D. B.; MACIEL, R. S. P. *Pragmatic Interoperability for Cloud Services: the CAPITAL Framework.* To be submitted to a journal.

Additionally, this thesis contributed less significantly in the publication of the following works in the FORMAS research group:

- VIEIRA, M. A.; **RIBEIRO, E. L. F.**; ROCHA, W. S.; MANE, B.; CLARO, D. B.; OLIVEIRA, J. S.; LIMA, E. Enhancing MIDAS Towards a Transparent Interoperability Between SaaS and DaaS. In: *13th Brazilian Symposium on Information Systems (SBSI).* Lavras, Brazil: SBC, 2017. p. 356–363. (VIEIRA et al., 2017).
- SILVA, N.; **RIBEIRO, E. L. F.**; CLARO, D. B. DaaS Repository Through MIDAS Web Crawler. In: *14th Brazilian Symposium on Information Systems (SBSI).* Caxias do Sul, Brazil: SBC, 2018. p. 246–253. (SILVA; RIBEIRO; CLARO, 2018).
- MANE, B.; ROCHA, W. S.; **RIBEIRO, E. L. F.**; JESUS, L. E. N.; MOTTA, I. C.; LIMA,E.; CLARO, D. B. Enhancing Semantic Interoperability on MIDAS with Similar DaaS Parameters. In: *16th Brazilian Symposium on Information Systems (SBSI).* São Bernardo do Campo, Brazil: SBC, 2020. p. 1–8. (MANE et al., 2020).
- VIEIRA, M. A.; **RIBEIRO, E. L. F.**; CLARO, D. B. *Integration Model between Heterogeneous Data Services in a Cloud.* Under review by Journal of Universal Computer Science (JUCS).

BIBLIOGRAPHY

ADEBESIN, F.; FOSTER, R.; KOTZÉ, P.; GREUNEN, D. V. A Review of Interoperability Standards in E-health and Imperatives for their Adoption in Africa. *South African Computer Journal*, v. 50, n. 1, p. 55–72, 2013.

AKERKAR, R. *Big Data Computing*. [S.l.]: CRC Press, 2014.

AN, Y. Z.; ZAABA, Z. F.; SAMSUDIN, N. F. Reviews on Security Issues and Challenges in Cloud Computing. *IOP Conference Series: Materials Science and Engineering*, v. 160, n. 012106, p. 1–9, 2016.

ANTONIOU, G.; HARMELEN, F. van. Web Ontology Language: OWL. In: _____. *Handbook on Ontologies*. [S.l.]: Springer, 2004. cap. 4, p. 62–92.

ARDAGNA, D.; NITTO, E.; MOHAGHEGHI, P.; MOSSER, S.; BALLAGNY, C.; D'ANDRIA, F.; CASALE, G.; MATTHEWS, P.; NECHIFOR, C.; PETCU, D.; GERICKE, A.; SHERIDAN, C. MODAClouds: A model-driven approach for the design and execution of applications on multiple Clouds. In: *4th International Workshop on Modeling in Software Engineering (MISE)*. Zurich, Switzerland: IEEE, 2012. p. 50–56.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A View of Cloud Computing. *Commun. ACM*, v. 53, n. 4, p. 50–58, 2010.

ARUNKUMAR, G.; VENKATARAMAN., N. A Novel Approach to Address Interoperability Concern in Cloud Computing. *Procedia Computer Science*, v. 50, n. 1, p. 554 – 559, 2015.

ASUNCION, C. H.; BOLDYREFF, C.; ISLAM, S.; LEONARD, M.; THALHEIM, B. Pragmatic interoperability in the enterprise - A research agenda. In: *23rd Conference on Advanced Information Systems Engineering (CAiSE)*. London, United Kingdom: Springer, 2011. p. 8.

ASUNCION, C. H.; IACOB, M.; SINDEREN, M. J. van. Towards a Flexible Service Integration through Separation of Business Rules. In: *14th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*. Vitoria, Brazil: IEEE, 2010. p. 184–193.

ASUNCION, C. H.; SINDEREN, M. J. van. Pragmatic Interoperability: A Systematic Review of Published Definitions. In: *5th International Conference Enterprise Architecture, Integration and Interoperability (EAI2N)*. Brisbane, Australia: Springer, 2010. p. 164–175.

BASILI, V. R.; ROMBACH, H. D. The TAME Project: Towards Improvement-oriented Software Environments. *IEEE Transactions on Software Engineering*, v. 14, n. 6, p. 758–773, 1988.

BENZADRI, Z.; BELALA, F.; BOUANAKA, C. Towards a Formal Model for Cloud Computing. In: LOMUSCIO, A. R.; NEPAL, S.; PATRIZI, F.; BENATALLAH, B.; BRANDIĆ, I. (Ed.). *11th International Conference on Service Oriented Computing (IC-SOC)*. Cham: Springer International Publishing, 2014. p. 381–393.

BRAVO, M.; ALVARADO, M. On the pragmatic similarity between agent communication protocols: Modeling and measuring. In: *8th Confederated International Conferences: On the Move to Meaningful Internet Systems (OTM)*. Berlin, Germany: Springer, 2008. p. 128–137.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. *Cloud Computing: Principles and Paradigms*. 1. ed. [S.l.]: John Wiley & Sons, 2011.

CHEN, D.; DOUMEINGTS, G.; VERNADAT, F. Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry*, v. 59, n. 7, p. 647 – 659, 2008.

COURONNE, O.; POLIAKOV, A.; BRAY, N.; ISHKHANOV, T.; RYABOY, D.; RUBINDWARD, E.; PACHTER, L.; DUBCHAK, I. Strategies and Tools for Whole-Genome Alignments. *Genome research*, v. 13, n. 1, p. 73–80, 2003.

CRICK, F. Central Dogma of Molecular Biology. *Nature*, v. 227, n. 1, p. 3, 1970.

DIKAIAKOS, M. D.; KATSAROS, D.; MEHRA, P.; PALLIS, G.; VAKALI, A. Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing*, v. 13, n. 5, p. 10–13, 2009.

DILLON, T.; WU, C.; CHANG, E. Cloud Computing: Issues and Challenges. In: *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Perth, Australia: IEEE, 2010. p. 27–33.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of Database Systems*. 6. ed. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2010.

EUZENAT, J.; SHVAIKO, P. et al. *Ontology matching*. 1. ed. [S.l.]: Springer, 2007.

FAHL, S.; HARBACH, M.; MUDERS, T.; SMITH, M. Confidentiality As a Service – Usable Security for the Cloud. In: *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. Liverpool, England: IEEE, 2012. p. 153–162.

FATIMA, H.; WASNIK, K. Comparison of SQL, NoSQL and NewSQL databases for internet of things. In: *2nd Bombay Section Symposium (IBSS)*. Baramati, India: IEEE, 2016. p. 1–6.

FERNANDES, D.; BERNARDINO, J. Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB (DATA). In: *7th International Conference on Data Science, Technology and Applications*. Porto, Portugal: SCITEPRESS, 2018. p. 373–380.

FERNÁNDEZ-LOPEZ, M.; GÓMEZ-PEREZ, A.; JURISTO, N. METHONTOLOGY: from Ontological Art towards Ontological Engineering. *AAAI-97 Spring Symposium Series*, v. 1, n. 1, p. 33–40, 1997.

FREITAS JUNIOR, M.; FANTINATO, M.; SUN, V. Improvements to the Function Point Analysis Method: A Systematic Literature Review. *IEEE Transactions on Engineering Management*, v. 62, n. 4, p. 495–506, 2015.

GRAVINA, R.; PALAU, C. E.; MANSO, M.; LIOTTA, A.; FORTINO, G. *Integration, Interconnection, and Interoperability of IoT Systems*. 1. ed. [S.l.]: Springer International Publishing, 2017.

GROLINGER, K.; HIGASHINO, W.; TIWARI, A.; CAPRETZ, M. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Application*, v. 2, n. 22, p. 24, 2013.

GRÜNINGER, M.; FOX, M. S. Methodology for the Design and Evaluation of Ontologies. In: *14th International Joint Conferences on Artificial Intelligence Organization (IJCAI)*. Montreal, Canada: IJCAI, 1995. p. 1–10.

HACIGUMUS, H.; IYER, B.; MEHROTRA, S. Providing database as a service. In: *18th International Conference on Data Engineering (ICDE)*. San Jose, USA: IEEE, 2002. p. 29–38.

HAN, J.; HAIHONG, E.; LE, G.; DU, J. Survey on NoSQL database. In: *6th International Conference on Pervasive Computing and Applications (ICPCA)*. Port Elizabeth, South Africa: IEEE, 2011. p. 363–366.

HOGAN, M.; LIU, F.; SOKOL, A.; TONG, J. *NIST Cloud Computing Standards Roadmap (SP 500-291)*. Gaithersburg, USA, 2011.

HOTELLING, H. et al. A generalized T test and measure of multivariate dispersion. In: *2nd Berkeley Symposium on Mathematical Statistics and Probability*. California, USA: University of California Press, 1951. p. 23–41.

ISODA, Y.; KURAKAKE, S.; IMAI, K. Context-Aware Computing System for Heterogeneous Applications. In: *1st International Workshop on Personalized Context Modeling and Management for UbiComp Applications (ubiPCMM)*. Tokyo, Japan: Springer, 2005. p. 17–25.

JABAREEN, Y. Building a Conceptual Framework: Philosophy, Definitions, and Procedure. *International Journal of Qualitative Methods*, v. 8, n. 4, p. 49–62, 2009.

JOSHI, K. P.; YESHA, Y.; FININ, T. Automating Cloud Services Life Cycle through Semantic Technologies . *IEEE Trans. Serv. Comput.*, v. 7, n. 1, p. 109–122, 2014.

KHORSHED, M. T.; ALI, A. B. M. S.; WASIMI, S. A. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. *Future Generation Computer Systems*, v. 28, n. 6, p. 833 – 851, 2012.

KOLB, D. A. *Experiential learning: experience as the source of learning and development.* 1. ed. [S.l.]: Prentice Hall, 1984.

KUBICEK, H.; CIMANDER, R.; SCHOLL, H. J. Layers of Interoperability. In: _____. *Organizational Interoperability in E-Government.* [S.l.]: Springer Berlin Heidelberg, 2011. cap. 7, p. 85–96.

LEE, J.; LEE, Y.; SHAH, S.; GELLER, J. HIS-KCWater: Context-aware Geospatial Data and Service Integration. In: *21st ACM Symposium on Applied Computing (SAC).* Seoul, Korea: ACM, 2007. p. 24–29.

LIU, F.; TONG, J.; MAO, J.; BOHN, R.; MESSINA, J.; BADGER, L.; LEAF, D. *NIST Cloud Computing: Reference Architecture (SP 500-292).* Gaithersburg, USA, 2011.

LIU, S.; LI, W.; LIU, K. Pragmatic Oriented Data Interoperability for Smart Healthcare Information Systems. In: *14th International Symposium on Cluster, Cloud and Grid Computing (CCGrid).* Chicago, USA: IEEE, 2014. p. 811–818.

LOUTAS, N.; KAMATERI, E.; BOSI, F.; TARABANIS, K. Cloud Computing Interoperability: The State of Play. In: *3rd IEEE International Conference on Cloud Computing Technology and Science (CLOUDCOM).* Athens, Greece: IEEE, 2011. p. 752–757.

LOVRENCIC, S.; CUBRILO, M. Ontology Evaluation - Comprising Verification and Validation. In: *19th Central European Conference on Information and Intelligent Systems (CECIIS).* Varazdin, Croatia: CECIIS, 2008. p. 1–7.

MA, H.; SCHEWE, K.; THALHEIM, B.; WANG, Q. A formal model for the interoperability of service clouds. *Serv. Oriented Comput. Appl.*, v. 6, n. 3, p. 189–205, 2012.

MACIEL, R. S. P.; DAVID, J. M. N.; CLARO, D. B.; BRAGA, R. Full Interoperability: Challenges and Opportunities for Future Information Systems. In: _____. *I Grand Research Challenges in IS in Brazil (GranDSI-BR) - 2016-2026.* [S.l.]: SBC, 2017. cap. 9, p. 107–118.

MALHOTRA, S.; DOJA, M. N.; ALAM, B.; ALAM, M. Bigdata analysis and comparison of bigdata analytic approaches. In: *3rd International Conference on Computing, Communication and Automation (ICCCA).* Greater Noida, India: IEEE, 2017. p. 309–314.

MALLIGA, P. Database Services for Cloud Computing – An Overview. *International Journal of Computers & Technology*, v. 2, n. 3, p. 67–70, 2012.

MANE, B.; ROCHA, W. S.; RIBEIRO, E. L. F.; JESUS, L. E. N.; MOTTA, I. C.; LIMA, E.; CLARO, D. B. Enhancing Semantic Interoperability on MIDAS with Similar DaaS Parameters. In: *16th Brazilian Symposium on Information Systems (SBSI)*. São Bernardo do Campo, Brazil: SBC, 2020. p. 1–8.

MARINHO, T.; CIDREIRA, V.; CLARO, D. B.; MANE, B. MIDAS: A Middleware to Provide Interoperability Between SaaS and DaaS. In: *12th Brazilian Symposium on Information Systems (SBSI)*. Florianópolis, Brazil: SBC, 2016. p. 401–408.

MCCARTHY, J. Notes on Formalizing Contexts. In: *13th International Joint Conference on Artificial Intelligence (IJCAI)*. Chambéry, France: IJCAI, 1993. p. 555–560.

MELL, P. M.; GRANCE, T. *The NIST Definition of Cloud Computing (SP 800-145)*. Gaithersburg, USA, 2011.

MIGUEL, P. A. C.; MORABITO, R.; PUREZA, V. *Metodologia de pesquisa em engenharia de produção e gestão de operações*. 1. ed. [S.l.]: Elsevier, 2010.

MONIRUZZAMAN, A. B. M.; HOSSAIN, S. A. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *CoRR*, v. 6, n. 4, p. 1–14, 2013.

MOSCATO, F.; AVERSA, R.; MARTINO, B. D.; FORTIş, T.; MUNTEANU, V. An analysis of mOSAIC ontology for Cloud resources annotation . In: *19th Federated Conference on Computer Science and Information Systems (FedCSIS)*. Szczecin, Poland: IEEE, 2011. p. 973–980.

MUGNIER, M.-L.; THOMAZO, M. An introduction to ontology-based query answering with existential rules. In: _____. *Reasoning Web: Reasoning on the Web in the Big Data Era*. Cham, Switzerland: Springer, 2014. cap. 6, p. 245–278.

NARANG, A.; GUPTA, D. A Review on Different Security Issues and Challenges in Cloud Computing. In: *1st International Conference on Computing, Power and Communication Technologies (GUCON)*. Greater Noida, India: IEEE, 2018. p. 121–125.

NEIVA, F. W.; DAVID, J. M. N.; BRAGA, R.; CAMPOS, F. Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature. *Information and Software Technology*, v. 72, n. 1, p. 137–150, 2016.

NORDSTOKKE, D. W.; ZUMBO, B. D. A new nonparametric Levene test for equal variances. *Psicologica*, v. 31, n. 2, p. 401–430, 2010.

OPARA-MARTINS, J.; SAHANDI, R.; TIAN, F. Critical review of vendor lock-in and its impact on adoption of cloud computing. In: *1st International Conference on Information Society (i-Society)*. London, UK: IEEE, 2014. p. 92–97.

OPARA-MARTINS, J.; SAHANDI, R.; TIAN, F. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, v. 5, n. 1, p. 4, 2016.

ORTIZ, M. Ontology Based Query Answering: The Story So Far. In: *7th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW)*. Puebla, Mexico: CEUR-WS.org, 2013. p. 1–14.

POKRAEV, S. *Model-Driven Semantic Integration of Service-Oriented Applications*. Tese (Doutorado) — The Netherlands, 2009.

POKRAEV, S.; REICHERT, M. U.; STEEN, M.; WIERINGA, R. J. Semantic and Pragmatic Interoperability: A Model for Understanding. In: *17th Conference on Advanced Information Systems Engineering (CAiSE)*. Porto, Portugal: FEUP, 2005. p. 377–382.

PRESSMAN, R. S. *Engenharia de Software: Uma Abordagem Profissional*. 7. ed. [S.l.]: AMGH Editora, 2011.

RANA, M. E.; DAUREN, J.; KUMARAN, S. An improved Requirements Engineering framework for cloud based application development. In: *13th IEEE Student Conference on Research and Development (SCOReD)*. Kuala Lumpur, Malaysia: IEEE, 2015. p. 702–709.

RANABAHU, A.; SHETH, A. Semantics Centric Solutions for Application and Data Portability in Cloud Computing. In: *2nd International Conference on Cloud Computing Technology and Science (CloudCom)*. Indianapolis, USA: IEEE, 2010. p. 234–241.

RAZALI, N. M.; WAH, Y. B. et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, v. 2, n. 1, p. 21–33, 2011.

REINSEL, D.; GANTZ, J.; RYDNING, J. *The Digitization of the World from Edge to Core*. Framingham, USA, 2018.

Rekik, M.; Boukadi, K.; Ben-Abdallah, H. Cloud description ontology for service discovery and selection . In: *10th International Joint Conference on Software Technologies (ICSOFT)*. Colmar, France: Springer, 2015. p. 1–11.

RIBEIRO, E. L. F.; MONTEIRO, E. L.; CLARO, D. B.; MACIEL, R. S. P. A Conceptual Framework for Pragmatic Interoperability. In: *15th Brazilian Symposium on Information Systems (SBSI)*. Aracaju, Brazil: SBC, 2019. p. 1–8.

RIBEIRO, E. L. F.; VIEIRA, M. A.; CLARO, D. B.; SILVA, N. Transparent Interoperability Middleware between Data and Service Cloud Layers. In: *8th International Conference on Cloud Computing and Services Science (CLOSER)*. Funchal, Portugal: SCITEPRESS, 2018. p. 148–157.

RIBEIRO, E. L. F.; VIEIRA, M. A.; CLARO, D. B.; SILVA, N. Interoperability Between SaaS and Data Layers: Enhancing the MIDAS Middleware. In: _____. *Communications in Computer and Information Science*. [S.l.]: Springer International Publishing, 2019. cap. 6, p. 102–125.

RIMAL, B. P.; CHOI, E.; LUMB, I. A Taxonomy and Survey of Cloud Computing Systems. In: *5th International Joint Conference on INC, IMS and IDC (NCM)*. Seoul, South Korea: IEEE, 2009. p. 44–51.

RIMAL, B. P.; JUKAN, A.; KATSAROS, D.; GOELEVEN, Y. Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach. *Journal of Grid Computing*, v. 9, n. 1, p. 3–26, 2011.

RINGS, T.; GRABOWSKI, J. Pragmatic Integration of Cloud and Grid Computing Infrastructures. In: *5th IEEE International Conference on Cloud Computing (CLOUD)*. Hawaii, USA: IEEE, 2012. p. 710–717.

RODERO-MERINO, L.; VAQUERO, L. M.; GIL, V.; GALÁN, F.; FONTÁN, J.; MONTERO, R. S.; LLORENTE, I. M. From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, v. 26, n. 8, p. 1226–1240, 2010.

SAHANDI, R.; ALKHALIL, A.; OPARA-MARTINS, J. Cloud Computing from SMEs Perspective: A Survey-based Investigation. *Journal of Information Technology Management (JITM)*, v. 24, n. 1, p. 1–12, 2013.

SCHAFFER, H. E. X as a Service, Cloud Computing, and the Need for Good Judgment. *IT Professional*, v. 11, n. 5, p. 4–5, 2009.

SCHREINER, G. A.; DUARTE, D.; MELLO, R. dos S. SQLtoKeyNoSQL: a layer for relational to key-based NoSQL database mapping. In: *17th International Conference on Information Integration and Web-based Applications & Services (iiWAS2019)*. Brussels, Belgium: ACM, 2015. p. 74.

SCHUBERT, L.; JEFFERY, K. New Software Engineering Requirements in Clouds and Large-Scale Systems. *IEEE Cloud Computing*, v. 2, n. 1, p. 48–58, 2015.

SCOTT, H. H. Interactive on-line manufacturing and repair system (io-mars). In: *10th International Automatic Testing Conference (AUTOTESTCON)*. Dayton, USA: IEEE, 1996. p. 40–45.

SEIBOLD, M.; KEMPER, A. Database as a Service. *Datenbank-Spektrum*, v. 12, n. 1, p. 59–62, 2012.

SHETH, A. P. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In: _____. *Interoperating Geographic Information Systems*. [S.l.]: Springer US, 1999. cap. 2, p. 5–29.

SILVA, G. C.; ROSE, L. M.; CALINESCU, R. A systematic review of cloud lock-in solutions. In: *5th International Conference on Cloud Computing Technology and Science (CloudCom)*. Bristol, UK: IEEE, 2013. p. 363–368.

SILVA, N.; RIBEIRO, E. L. F.; CLARO, D. B. DaaS Repository Through MIDAS Web Crawler. In: *14th Brazilian Symposium on Information Systems (SBSI)*. Caxias do Sul, Brazil: SBC, 2018. p. 246–253.

SIMMON, E. *Evaluation of Cloud Computing Services Based on NIST (SP 800-145)*. Gaithersburg, USA, 2018.

SIRIN, E.; PARSIA, B.; GRAU, B. C.; KALYANPUR, A.; KATZ, Y. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, v. 5, n. 2, p. 51 – 53, 2007.

SPIVEY, J. M. *The Z Notation: A Reference Manual*. 1. ed. [S.l.]: Prentice-Hall, Inc., 1989.

STAAB, S.; STUDER, R.; SCHNURR, H.; SURE, Y. Knowledge processes and ontologies. *IEEE Intelligent Systems*, v. 16, n. 1, p. 26–34, 2001.

STOREY, V. C.; SONG, I.-Y. Big data technologies and Management: What conceptual modeling can do. *Data & Knowledge Engineering*, v. 108, n. 1, p. 50 – 67, 2017.

STRASSNER, J.; DIAB, W. W. A semantic interoperability architecture for Internet of Things data sharing and computing. In: *3rd IEEE World Forum on Internet of Things (WF-IoT)*. Reston, USA: IEEE, 2016. p. 609–614.

TAMANI, E.; EVRIPIDOU, P. A Pragmatic Methodology to Web Service Discovery. In: *4th IEEE International Conference on Web Services (ICSW)*. Salt Lake City, USA: IEEE, 2007. p. 1168–1171.

TARIQ, A.; KHAN, S. A.; IFTIKHAR, S. Requirements Engineering process for Software-as-a-Service (SaaS) cloud environment. In: *10th International Conference on Emerging Technologies (ICET)*. Islamabad, Pakistan: IEEE, 2014. p. 13–18.

TERZO, O.; RUIU, P.; BUCCI, E.; XHAFA, F. Data as a Service (DaaS) for Sharing and Processing of Large Data Collections in the Cloud. In: *7th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*. Taichung, Taiwan: IEEE, 2013. p. 475–480.

TRUONG, H. L.; DUSTDAR, S. On Analyzing and Specifying Concerns for Data as a Service. In: *4th IEEE Asia-Pacific Services Computing Conference (APSCC)*. Biopolis, Singapore: IEEE, 2009. p. 87–94.

TUNG, Y.; LIN, C.; SHAN, H. Test as a Service: A Framework for Web Security TaaS Service in Cloud Environment. In: *8th IEEE International Symposium on Service Oriented System Engineering (SOSE)*. Oxford, England: IEEE, 2014. p. 212–217.

USCHOLD, M.; KING, M. Towards a Methodology for Building Ontologies. In: *14th International Joint Conferences on Artificial Intelligence Organization (IJCAI)*. Montreal, Canada: IJCAI, 1995. p. 1–15.

VALLE, P. H. D.; GARCÉS, L.; NAKAGAWA, E. Y. A Typology of Architectural Strategies for Interoperability. In: *8th Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS)*. Salvador, Brazil: Association for Computing Machinery, 2019. p. 3–12.

van der VEER, H.; WILES, A. *Achieving Technical Interoperability - The ETSI Approach.* Sophia Antipolis Cedex, France, 2008.

VECCHIOLA, C.; CHU, X.; BUYYA, R. Aneka: a Software Platform for .NET based Cloud Computing. *CoRR*, v. 9, n. 7, p. 267–295, 2009.

VIDAL, V. M. P.; SACRAMENTO, E. R.; MACÊDO, J. A. F.; CASANOVA, M. A. An Ontology-Based Framework for Geographic Data Integration . In: *28th International Conference on Conceptual Modeling (ER).* Berlin, Heidelberg: Springer, 2009. p. 337–346.

VIEIRA, M. A.; RIBEIRO, E. L. F.; ROCHA, W. S.; MANE, B.; CLARO, D. B.; OLIVEIRA, J. S.; LIMA, E. Enhancing MIDAS Towards a Transparent Interoperability Between SaaS and DaaS. In: *13th Brazilian Symposium on Information Systems (SBSI).* Lavras, Brazil: SBC, 2017. p. 356–363.

WANG, G.; TANG, J. The NoSQL Principles and Basic Application of Cassandra Model. In: *2nd International Conference on Computer Science and Service System (CSSS).* Nanjing, China: IEEE, 2012. p. 1332–1335.

WANG, W.; TOLK, A.; WANG, W. The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S. In: *3rd Spring Simulation Conference (SpringSim).* San Diego, USA: ACM, 2009. p. 1–9.

WAZLAWICK, R. *Metodologia de Pesquisa para Ciência da Computação.* 2. ed. [S.l.]: Elsevier Editora Ltda., 2017.

WEBSTER, C. *From Syntactic & Semantic To Pragmatic Interoperability In Healthcare.* 2014.

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. *Experimentation in Software Engineering.* 1. ed. [S.l.]: Springer, 2012.

ZARKO, I. P.; MUELLER, S.; PLOCIENNIK, M.; RAJTAR, T.; JACOBY, M.; PARDI, M.; INSOLVIBILE, G.; GLYKANTZIS, V.; ANTONIĆ, A.; KUSEK, M.; SOURSOS, S. The symbIoTe Solution for Semantic and Syntactic Interoperability of Cloud-based IoT Platforms. In: *3rd Global IoT Summit (GIoTS).* Aarhus, Denmark: IEEE, 2019. p. 1–6.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, v. 1, n. 1, p. 7–18, 2010.

ZHANG, Q.; HALLER, A.; WANG, Q. CoCoOn: Cloud Computing Ontology for IaaS Price and Performance Comparison. In: *18th International Semantic Web Conference (ISWC).* Auckland, New Zealand: Springer, 2019. p. 325–341.

ZHANG, Z.; WU, C.; CHEUNG, D. W. A survey on cloud interoperability: taxonomies, standards, and practice. *ACM SIGMETRICS Performance Evaluation Review*, v. 40, n. 4, p. 13–22, 2013.

ZHENG, Z.; ZHU, J.; LYU, M. R. Service-Generated Big Data and Big Data-as-a-Service: An Overview. In: *2nd International Congress on Big Data (BigData Congress)*. Santa Clara, USA: IEEE, 2013. p. 403–410.

Appendix

A

EXAMPLE OF DATA SOURCES

This Appendix presents examples of data in DaaS *w7* and *vz*. We consider these data to validate the models for syntactic and semantic interoperability.

Table A.1 Generic examples of data in DaaS *w7*

id	age	firstname	lastname	blood
1	21	Alana	Pena	A–
2	9	Oleg	John	AB+
5	39	Ayanna	Torres	B+
7	38	Howard	Whit	O–
10	19	Dakota	Garner	O+
13	8	Eliana	James	A+
15	86	Emerson	Hary	AB-
20	10	Alf	Wheeler	A–

Table A.2 Generic examples of data in DaaS *vz*

ID	phone	borough
1	1157-1882	queens
2	2964-9218	bronx
7	8710-6657	manhattan
10	8724-3917	queens
11	6117-9480	brooklyn
12	5396-1807	queens
14	1245-6236	manhattan
20	2134-4205	queens

The data is fictitious and both DaaS are purposefully simple. The complete and original data are available on the DaaS homepages:

- $w7$: `https://data.cityofnewyork.us/Health/NYC-Health-Hospitals-patient-care-locations-2011/f7b6-v6v3`
- vz: `https://data.cityofnewyork.us/Education/Borough-Enrollment-Offices/vz8c-29aj`

Appendix B

MIDAS-OWL

This Appendix presents *MIDAS-OWL* in RDF/XML syntax.

Listing B.1 *MIDAS-OWL*

```
<?xml version="5.3"?>
<rdf:RDF xmlns="http://ontmidas.000webhostapp.com/"
     xml:base="http://ontmidas.000webhostapp.com/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:obda="https://w3id.org/obda/vocabulary#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:ontmidas="http://ontmidas.000webhostapp.com/#"
     xmlns:untitled-ontology-5="http://www.semanticweb.org/elivaldo/ontologies/2019/9/untitled-ontology-5#">
    <owl:Ontology rdf:about="http://ontmidas.000webhostapp.com/">
        <owl:versionIRI rdf:resource="http://ontmidas.000webhostapp.com/"/>
    </owl:Ontology>

    <!--
    ///////////////////////////////////////
    // Object Properties
    ///////////////////////////////////////
    -->

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#assoc_from">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#associates"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#dataset_info"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#from"/>
        <rdfs:comment>This object property associates "dataset info" (DaaS name) and "from" (query source).</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#assoc_limit">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#associates"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#limit_info"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#limit"/>
        <rdfs:comment>This object property associates "limit info" (information about "limit" clause of each DaaS) and
            "limit" clause of the query.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#assoc_order_by">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#associates"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#sort_info"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#order_by"/>
        <rdfs:comment>This object property associates "sort info" (information about "order by" clause of each DaaS)
            and "order by" clause of the query.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#assoc_select">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#associates"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#records_info"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#select"/>
        <rdfs:comment>This object property associates "records info" (information about "select" clause of each DaaS)
            and "select" clause of the query.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#assoc_where">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#associates"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#query_info"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#where"/>
        <rdfs:comment>This object property associates "query info" (information about "where" clause of each DaaS) and
              "where" clause of the query.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#associates">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:comment>This property object associates elements of mDIS (for each DaaS) with elements in the query.</
             rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#composed_of">
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This object property indicates that an attribute can be composed of another attribute.</rdfs:
             comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#generates">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#join"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#source"/>
        <rdfs:comment>This object property indicates that a join operation generates a data source.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_DaaS">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#mDIS"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#DaaS"/>
        <rdfs:comment>This object property indicates that the mDIS contains a set of DaaS.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:comment>This object property indicates that a clause contains attribute(s).</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb_group_by">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_attrb"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#group_by"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb_group"/>
        <rdfs:comment>This object property indicates that a "group by" clause contains an attribute: "attrb_group".</
             rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb_having">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_attrb"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#having"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb_hav"/>
        <rdfs:comment>This object property indicates that a "having" clause contains an attribute: "attrb_hav".</rdfs:
             comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb_join">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_attrb"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#join"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb1_join"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb2_join"/>
        <rdfs:comment>This object property indicates that a join opoeration contains two attributes: "attrb1_join" and
              "attrb2_join".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb_order_by">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_attrb"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#order_by"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb_ord"/>
        <rdfs:comment>This object property indicates that a "order by" clause contains an attribute: "attrb_ord".</
             rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb_proj">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_attrb"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#select"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb_proj"/>
        <rdfs:comment>This object property indicates that a "select" clause contains an attribute: "attrb_proj".</rdfs
             :comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_attrb_sel">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_attrb"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#selection"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attrb_sel"/>
        <rdfs:comment>This object property indicates that a "where" clause contains an attribute: "attrb_sel".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_clau">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#query"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This object property indicates that a query contains a set of clauses, e.g., select, from, where, limit.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_ext">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#return"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#extension"/>
        <rdfs:comment>This object property indicates that each return contains an extension. For instance, JSON files contains ".json" extension.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#DaaS"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This object property indicates that a DaaS contains a set of information.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_dom">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about domain: "domain_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_ds">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about dataset name: "dataset_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_fie">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about fields: "fields_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_form">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about formats: "format_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_lim">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about "limit" clause: "limit_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_quer">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about "where" clause: "query_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_rec">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about "select" clause: "records_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_sort">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about "order by" clause: "sort_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_info_sp">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_info"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:comment>This object property indicates that a DaaS contains a information about search path: "search_path_info".</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_join">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#join"/>
        <rdfs:comment>This object property indicates that a clause can contain a join operation.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_op">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:comment>This object property indicates that some clauses can contain some operations.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_op_having">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_op"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#having"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#op_hav"/>
        <rdfs:comment>This object property indicates that the "having" clause contains an operations: "op_hav".</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_op_join">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_op"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#join"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#op_join"/>
        <rdfs:comment>This object property indicates that the join operation contains an operations: "op_join".</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_op_selection">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_op"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#selection"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#op_sel"/>
        <rdfs:comment>This object property indicates that the "where" clause contains an operations: "op_sel".</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_ret">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#query"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#return"/>
        <rdfs:comment>This object property indicates that a query contains a return.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_sel">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#where"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#selection"/>
        <rdfs:comment>This object property indicates that the "where" clause contains a selection operation.</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_source">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#from"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#source"/>
        <rdfs:comment>This object property indicates that the "from" clause contains source.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_type_join">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#join"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#type_join"/>
        <rdfs:comment>This object property indicates that the join operation contains a type join.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_val">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:comment>This object property indicates that some clauses can contain some values.</rdfs:comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_val_having">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_val"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#having"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#value_hav"/>
        <rdfs:comment>This object property indicates that the "having" clause contains an value: "value_hav".</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_val_limit">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_val"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#limit"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#value_limit"/>
        <rdfs:comment>This object property indicates that the "limit" clause contains an value: "value_limit".</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#has_val_selection">
        <rdfs:subPropertyOf rdf:resource="http://ontmidas.000webhostapp.com/#has_val"/>
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#selection"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#value_sel"/>
        <rdfs:comment>This object property indicates that the "where" clause contains an value: "value_sel".</rdfs:
            comment>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://ontmidas.000webhostapp.com/#is_similar_to">
        <rdfs:domain rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:range rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This object property indicates that an attribute can be semantically similar to another
            attribute.</rdfs:comment>
    </owl:ObjectProperty>

    <!--
    ///////////////////////////////////
    // Classes
    ///////////////////////////////////
    -->

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#DaaS"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb1_join">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#parameter"/>
        <rdfs:comment>This class models the first attribute of the join operation. For instance: given "FROM w7 LEFT
            OUTER JOIN vz ON w7.id = vz.id", attrb1_join = "w7.id".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb2_join">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#parameter"/>
        <rdfs:comment>This class models the second attribute of the join operation. For instance: given "FROM w7 LEFT
            OUTER JOIN vz ON w7.id = vz.id", attrb2_join = "vz.id".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb_group">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This class models the attribute(s) of the "group by" clause. For instance: given "GROUP BY w7.
            age" clause, attrb_group = "w7.age".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb_hav">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This class models the attribute(s) of the "having" clause. For instance: given "HAVING vz.b1 = "
            queens"" clause, attrb_hav = "vz.b1".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb_ord">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This class models the attribute(s) of the "order by" clause. For instance: given "ORDER BY w7.
            lastname" clause, attrb_ord = "w7.lastname".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb_proj">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This class models the attribute(s) of the "select" clause. For instance: given "SELECT w7.
            firstname, w7.lastname, w7.age" clause, attrb_proj has the values "w7.firstname", "w7.lastname", and "w7.
            age".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attrb_sel">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>This class models the attribute(s) of the "where" clause. For instance: given "WHERE vz.b1 = "
            queens"" clause, attrb_sel = "vz.b1".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#attribute"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#clauses"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#dataset_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the name of each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#domain_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the domain of each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>
```

```xml

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#extension"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#fields_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class represents all fields (attributes) of each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#format_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#return"/>
        <rdfs:comment>This class represents the formats recognized by each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#from">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "from" clause. For instance: given "FROM w7" clause, from
             = "FROM _w7".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#group_by">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "group by" clause. For instance: given "GROUP BY w7.age"
             clause, group_by = "GROUP_BY_w7.age".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#having">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "having" clause. For instance: given "HAVING vz.b1 = "
             queens"" clause, having = "HAVING_vz.b1_equals_queens".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#infoDaaS"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#join"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#limit">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "limit" clause. For instance: given "LIMIT 15" clause,
             limit = "LIMIT_15".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#limit_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the information about "limit" clause of each DaaS present in mDIS.</rdfs:
             comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#mDIS"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#op_hav">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>This class models the operation of the "having" clause. For instance: given "HAVING vz.b1 = "
             queens"" clause, op_hav = "equals".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#op_join">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#parameter"/>
        <rdfs:comment>This class models the operation of the join operation. For instance: given "FROM w7 LEFT OUTER
             JOIN vz ON w7.id = vz.id", op_join = "equals".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#op_sel">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>This class models the operation of the "where" clause. For instance: given "WHERE vz.b1 = "
             queens"" clause, op_sel = "equals".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#operator"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#order_by">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "order by" clause. For instance: given "ORDER BY w7.
             lastname" clause, order_by = "ORDER_BY_w7.lastname".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#parameter"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#query"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#query_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the information about "where" clause of each DaaS present in mDIS.</rdfs:
             comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#records_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the information about "select" clause of each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#return"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#search_path_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the search path of each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#select">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "select" clause. For instance: given "SELECT w7.firstname, w7.lastname, w7.age" clause, select = "SELECT_w7.firstname_w7.lastname_w7.age".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#selection"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#sort_info">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#infoDaaS"/>
        <rdfs:comment>This class models the information about "order by" clause of each DaaS present in mDIS.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#source"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#type_join">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#parameter"/>
        <rdfs:comment>This class models the type join of the join operation. For instance: given "FROM w7 LEFT OUTER JOIN vz ON w7.id = vz.id", type_join = "LEFT_OUTER_JOIN".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#value"/>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#value_hav">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#value"/>
        <rdfs:comment>This class models the value(s) of the "having" clause. For instance: given "HAVING vz.b1 = "queens"" clause, value_hav = "queens".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#value_limit">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#value"/>
        <rdfs:comment>This class models the value of the "limit" clause. For instance: given "LIMIT 15" clause, value_limit = "15".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#value_sel">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#value"/>
        <rdfs:comment>This class models the value(s) of the "where" clause. For instance: given "WHERE vz.b1 = "queens"" clause, value_sel = "queens".</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:about="http://ontmidas.000webhostapp.com/#where">
        <rdfs:subClassOf rdf:resource="http://ontmidas.000webhostapp.com/#clauses"/>
        <rdfs:comment>This class models all elements of the "where" clause. For instance: given "WHERE vz.b1 = "queens"" clause, where = "WHERE_vz.b1_equals_queens".</rdfs:comment>
    </owl:Class>

    <!--
    ///////////////////////////////////////
    // Individuals
    ///////////////////////////////////////
    -->

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#DaaS_vz">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#DaaS"/>
        <rdfs:comment>Example of DaaS name.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#DaaS_w7">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#DaaS"/>
        <rdfs:comment>Example of DaaS name.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#FROM_w7">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#from"/>
        <ontmidas:has_source rdf:resource="http://ontmidas.000webhostapp.com/#w7"/>
        <rdfs:comment>Example of "from" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#FROM_w7_LEFT_OUTER_JOIN_vz_ON_w7.id_equals_vz.id">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#from"/>
        <ontmidas:has_join rdf:resource="http://ontmidas.000webhostapp.com/#join1"/>
        <ontmidas:has_source rdf:resource="http://ontmidas.000webhostapp.com/#vz"/>
        <ontmidas:has_source rdf:resource="http://ontmidas.000webhostapp.com/#w7"/>
        <rdfs:comment>Example of "from" clause.</rdfs:comment>
    </owl:NamedIndividual>
    <owl:Axiom>
```

```
        <owl:annotatedSource rdf:resource="http://ontmidas.000webhostapp.com/#FROM_w7_LEFT_OUTER_JOIN_vz_ON_w7.
             id_equals_vz.id"/>
        <owl:annotatedProperty rdf:resource="http://ontmidas.000webhostapp.com/#has_join"/>
        <owl:annotatedTarget rdf:resource="http://ontmidas.000webhostapp.com/#join1"/>
        <rdfs:comment>teste</rdfs:comment>
    </owl:Axiom>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#FULL_OUTER_JOIN">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#type_join"/>
        <rdfs:comment>Example of type join.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#INNER_JOIN">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#type_join"/>
        <rdfs:comment>Example of type join.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#LEFT_OUTER_JOIN">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#type_join"/>
        <rdfs:comment>Example of type join.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#LIMIT_15">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#limit"/>
        <ontmidas:has_val_limit rdf:resource="http://ontmidas.000webhostapp.com/#15"/>
        <rdfs:comment>Example of "limit" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#LIMIT_5">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#limit"/>
        <ontmidas:has_val_limit rdf:resource="http://ontmidas.000webhostapp.com/#5"/>
        <rdfs:comment>Example of "limit" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#ORDER_BY_lastname">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#order_by"/>
        <ontmidas:has_attrb_order_by rdf:resource="http://ontmidas.000webhostapp.com/#w7.lastname"/>
        <rdfs:comment>Example of "order by" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#ORDER_BY_w7.lastname">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#order_by"/>
        <ontmidas:has_attrb_order_by rdf:resource="http://ontmidas.000webhostapp.com/#w7.lastname"/>
        <rdfs:comment>Example of "order by" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#RIGHT_OUTER_JOIN">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#type_join"/>
        <rdfs:comment>Example of type join.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#SELECT_firstname_lastname_age">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#select"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#w7.age"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#w7.firstname"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#w7.lastname"/>
        <rdfs:comment>Example of "select" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#SELECT_w7.firstname_w7.lastname_w7.age_vz.
         phone">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#select"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#vz.phone"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#w7.age"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#w7.firstname"/>
        <ontmidas:has_attrb_proj rdf:resource="http://ontmidas.000webhostapp.com/#w7.lastname"/>
        <rdfs:comment>Example of "select" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#WHERE_age_isMoreThan_10">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#where"/>
        <ontmidas:has_sel rdf:resource="http://ontmidas.000webhostapp.com/#age_isMoreThan_10"/>
        <rdfs:comment>Example of "where" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#WHERE_vz.b1_equals_queens">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#where"/>
        <ontmidas:has_sel rdf:resource="http://ontmidas.000webhostapp.com/#vz.b1_equals_queens"/>
        <rdfs:comment>Example of "where" clause.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#age_isMoreThan_10">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#selection"/>
        <ontmidas:has_attrb_sel rdf:resource="http://ontmidas.000webhostapp.com/#w7.age"/>
        <ontmidas:has_op_selection rdf:resource="http://ontmidas.000webhostapp.com/#isMoreThan"/>
        <ontmidas:has_val_selection rdf:resource="http://ontmidas.000webhostapp.com/#10"/>
        <rdfs:comment>Example of selection ("where" clause).</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#csv">
```

```
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#format_info"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#return"/>
        <ontmidas:has_ext rdf:resource="http://ontmidas.000webhostapp.com/#.csv"/>
        <rdfs:comment>Example of value for format_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#equals">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_hav"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_join"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_sel"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>Example of operation.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#isLessEqualThan">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_hav"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_sel"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>Example of operation.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#isLessThan">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_hav"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_sel"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>Example of operation.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#isMoreEqualThan">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_hav"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_sel"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>Example of operation.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#isMoreThan">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_hav"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#op_sel"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#operator"/>
        <rdfs:comment>Example of operation.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#join1">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#join"/>
        <ontmidas:generates rdf:resource="http://ontmidas.000webhostapp.com/#vz"/>
        <ontmidas:generates rdf:resource="http://ontmidas.000webhostapp.com/#w7"/>
        <ontmidas:has_attrb_join rdf:resource="http://ontmidas.000webhostapp.com/#vz.id"/>
        <ontmidas:has_attrb_join rdf:resource="http://ontmidas.000webhostapp.com/#w7.id"/>
        <ontmidas:has_op_join rdf:resource="http://ontmidas.000webhostapp.com/#equals"/>
        <ontmidas:has_type_join rdf:resource="http://ontmidas.000webhostapp.com/#LEFT_OUTER_JOIN"/>
        <rdfs:comment>Example of join operation.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#json">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#format_info"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#return"/>
        <ontmidas:has_ext rdf:resource="http://ontmidas.000webhostapp.com/#.json"/>
        <rdfs:comment>Example of value for format_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#lim">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#limit_info"/>
        <rdfs:comment>Example of value for limit_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#ord">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#sort_info"/>
        <rdfs:comment>Example of value for sort_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#queens">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#value_sel"/>
        <rdfs:comment>Example of value for value_sel.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#queryWithJoin">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#query"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#FROM_w7_LEFT_OUTER_JOIN_vz_ON_w7.
             id_equals_vz.id"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#LIMIT_15"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#ORDER_BY_w7.lastname"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#SELECT_w7.firstname_w7.lastname_w7.age_vz
             .phone"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#WHERE_vz.b1_equals_queens"/>
        <ontmidas:has_ret rdf:resource="http://ontmidas.000webhostapp.com/#json"/>
        <rdfs:comment>Example of query.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#queryWithoutJoin">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#query"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#FROM_w7"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#LIMIT_5"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#ORDER_BY_lastname"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#SELECT_firstname_lastname_age"/>
        <ontmidas:has_clau rdf:resource="http://ontmidas.000webhostapp.com/#WHERE_age_isMoreThan_10"/>
        <ontmidas:has_ret rdf:resource="http://ontmidas.000webhostapp.com/#json"/>
        <rdfs:comment>Example of query.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#rec">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#records_info"/>
        <rdfs:comment>Example of value for records_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#recds">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#records_info"/>
        <rdfs:comment>Example of value for records_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#setDaaS">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#mDIS"/>
        <rdfs:comment>Example of mDIS name.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#vz">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#dataset_info"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#source"/>
        <rdfs:comment>Example of value for dataset_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#vz.b1">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb_sel"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#fields_info"/>
        <rdfs:comment>Attribute. Example of value for attrb_sel and fields_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#vz.b1_equals_queens">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#selection"/>
        <ontmidas:has_attrb_sel rdf:resource="http://ontmidas.000webhostapp.com/#vz.b1"/>
        <ontmidas:has_op_selection rdf:resource="http://ontmidas.000webhostapp.com/#equals"/>
        <ontmidas:has_val_selection rdf:resource="http://ontmidas.000webhostapp.com/#queens"/>
        <rdfs:comment>Example of selection.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#vz.id">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb2_join"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#fields_info"/>
        <rdfs:comment>Attribute. Example of value for attrb2_join and fields_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#vz.phone">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb_proj"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#fields_info"/>
        <rdfs:comment>Attribute. Example of value for attrb_proj and fields_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#dataset_info"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#source"/>
        <rdfs:comment>Example of value for dataset_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.age">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb_proj"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#fields_info"/>
        <rdfs:comment>Attribute. Example of value for attrb_proj and fields_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.blood">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>Example of attribute.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.familyname">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <ontmidas:is_similar_to rdf:resource="http://ontmidas.000webhostapp.com/#w7.lastname"/>
        <rdfs:comment>Example of attribute.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.firstname">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>Example of attribute.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.id">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb1_join"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#fields_info"/>
        <rdfs:comment>Attribute. Example of value for attrb1_join and fields_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.identifier">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdfs:comment>Example of attribute.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#w7.lastname">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb_ord"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attrb_proj"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#attribute"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#fields_info"/>
        <rdfs:comment>Attribute. Example of value for attrb_ord, attrb_proj, and fields_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#xml">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#format_info"/>
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#return"/>
        <ontmidas:has_ext rdf:resource="http://ontmidas.000webhostapp.com/#.xml"/>
        <rdfs:comment>Example of value for format_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#$limit">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#limit_info"/>
        <rdfs:comment>Example of value for limit_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#$order">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#sort_info"/>
        <rdfs:comment>Example of value for sort_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#$where_v">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#query_info"/>
        <rdfs:comment>Example of value for query_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#$where_w">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#query_info"/>
        <rdfs:comment>Example of value for query_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#.csv">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#extension"/>
        <rdfs:comment>Example of extension.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#.json">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#extension"/>
        <rdfs:comment>Example of extension.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#.xml">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#extension"/>
        <rdfs:comment>Example of extension.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#/r/">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#search_path_info"/>
        <rdfs:comment>Example of value for search_path_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#/v/">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#search_path_info"/>
        <rdfs:comment>Example of value for search_path_info.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#10">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#value_sel"/>
        <rdfs:comment>Example of value for value_sel.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#15">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#value_limit"/>
        <rdfs:comment>Example of value for value_limit.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#5">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#value_limit"/>
        <rdfs:comment>Example of value for value_limit.</rdfs:comment>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#d.com/w7">
        <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#domain_info"/>
        <rdfs:comment>Example of value for domain_info.</rdfs:comment>
    </owl:NamedIndividual>

```

```
<owl:NamedIndividual rdf:about="http://ontmidas.000webhostapp.com/#qw.net/vz">
    <rdf:type rdf:resource="http://ontmidas.000webhostapp.com/#domain_info"/>
    <rdfs:comment>Example of value for domain_info.</rdfs:comment>
</owl:NamedIndividual>

<!-- Generated by the OWL API (version 4.5.9.2019-02-01T07:24:44Z) https://github.com/owlcs/owlapi -->
```

Appendix

C

CONTROLLED EXPERIMENT: SUPPORT MATERIAL

This Appendix presents the questionnaires (in Portuguese) and participants of our controlled experiment (Chapter 7). This appendix is organized as follows:

1. Section C.1 presents the pre-questionnaire applied to all participants.
2. Section C.2 presents the questionnaire applied to Control group (ConG).
3. Section C.3 presents the questionnaire applied to CAPITAL group (CapG).
4. Section C.4 presents a participants overview.

C.1 PRE-QUESTIONNAIRE

A. *Nome*: ______________________

B. E-mail: ______________________

C. *Escolaridade*:

◯ *Graduação incompleta*
◯ *Graduação completa*
◯ *Especialização*
◯ *Mestrado*
◯ *Doutorado*

Sobre interoperabilidade (1/2)

D. *Qual sua experiência com interoperabilidade?*

◯ *Não conhece e nunca utilizou*

◯ *Conhece o conceito mas nunca utilizou*

◯ *Conhece o conceito e utiliza (ou já utilizou)*

E. *Caso já tenha utilizado interoperabilidade antes, como/onde utilizou?*

☐ *Projetos de pesquisa*

☐ *Disciplinas ou cursos*

☐ *Empresas*

☐ *Outros projetos*

F. *Caso já tenha utilizado interoperabilidade anteriormente, utilizou por quanto tempo?*

◯ *Menos de 1 ano*

◯ *Entre 1 e 5 anos*

◯ *Entre 5 e 10 anos*

◯ *Mais de 10 anos*

G. *Qual sua experiência com interoperabilidade pragmática?*

◯ *Não conhece e nunca utilizou*

◯ *Conhece o conceito mas nunca utilizou*

◯ *Conhece o conceito e utiliza (ou já utilizou)*

H. *Caso já tenha utilizado interoperabilidade pragmática antes, como/onde utilizou?*

☐ *Projetos de pesquisa*

☐ *Disciplinas ou cursos*

☐ *Empresas*

☐ *Outros projetos*

I. *Caso já tenha utilizado interoperabilidade pragmática antes, utilizou por quanto tempo?*

◯ *Menos de 1 ano*

◯ *Entre 1 e 5 anos*

◯ *Entre 5 e 10 anos*

◯ *Mais de 10 anos*

Sobre interoperabilidade (2/2)

J. *Na sua opinião, qual dos conceitos abaixo melhor define interoperabilidade?*

◯ *Capacidade de um sistema combinar informações, artefatos e serviços entre si*

◯ *Capacidade de um sistema se comunicar de forma transparente com outro sistema*

◯ *Capacidade de extrair informações do cliente sobre o que ele deseja que seja desenvolvido*

◯ *Capacidade de componentes serem facilmente movidos e reutilizados, independentemente do formato e do provedor*

◯ *Não sei*

K. *Na sua opinião, qual dos conceitos abaixo melhor define interoperabilidade pragmática?*

◯ *Capacidade de entidades distintas compreenderem o significado da mensagem trocada entre os participantes*

◯ *Capacidade de entidades distintas compreenderem a intenção da mensagem trocada, de modo que o resultado produzido esteja dentro das expectativas comuns*

◯ *Capacidade de entidades distintas atingirem todos os níveis de interoperabilidade desejados, do mais básico ao mais específico*

◯ *Capacidade de entidades distintas trocarem dados entre si a partir de uma codificação comum*

◯ *Não sei*

C.2 CONTROL GROUP QUESTIONNAIRE

A. *Nome*: ____________________

B. *Hora inicial*: ____________________

Parte I

A literatura apresenta várias definições para "interoperabilidade pragmática". A seguir, listamos algumas dessas definições:

- A interoperabilidade pragmática garante que as mensagens trocadas causem o efeito pretendido. Muitas vezes, isso ocorre enviando e recebendo múltiplas mensagens em ordem específica, definidas em um protocolo de interação [1].
- A interoperabilidade pragmática é atingida quando os sistemas interoperantes são conscientes dos métodos e procedimentos uns dos outros. Ou seja, o contexto da aplicação é entendido pelos sistemas participantes [2].
- A web pragmática pode ajudar a superar o problema de descoberta na web porque é um conjunto de contextos pragmáticos sobre os recursos semânticos [3].
- A interoperabilidade pragmática é a compatibilidade entre o efeito pretendido e o efeito real da troca de mensagens [4].
- Compatibilidade entre uso pretendido e uso real da mensagem recebida dentro de um contexto compartilhado relevante [5].

- A interoperabilidade é alcançada neste nível quando os processos que atendem diferentes finalidades em diferentes contextos por diferentes sistemas de informação, podem ser compostos para apoiar conjuntamente uma intenção comum [6].
- A pragmática é sobre como usamos sintaxe e semântica como uma ferramenta para atingir metas. A interoperabilidade pragmática faz algo útil com os resultados da sintática e semântica. O significado não literal deve ser inferido a partir do conhecimento compartilhado (tarefas, fluxos de trabalho, planos e objetivos) [7].

[1] S. Pokraev, et al. 2005. Semantic and Pragmatic Interoperability: A Model for Understanding. In Proceedings of the CAiSE'05 Workshops, J. Castro and E. Teniente (Eds.). 377–382.
[2] J. Lee, et al. 2007. HIS-KCWater: Context-aware Geospatial Data and Service Integration. In Proceedings of the 2007 ACM Symposium on Applied Computing. ACM, New York, NY, USA, 24–29.
[3] E. Tamani and P. Evripidou. 2007. A Pragmatic Methodology to Web Service Discovery. In IEEE International Conference on Web Services. 1168–1171.
[4] C. H. Asuncion and M. M. J. v. Sinderen. 2011. Pragmatic interoperability: A systematic review of published definitions. Enterprise Architecture, Integration and Interoperability. 731 (2011), 164–175.
[5] C. H. Asuncion, C. Boldyreff, S. Islam, M. Leonard, and B. Thalheim. 2011. Pragmatic interoperability in the enterprise - A research agenda. 731.
[6] S. Liu, W. Li, and K. Liu. 2014. Pragmatic Oriented Data Interoperability for Smart Healthcare Information Systems. In 14th International Symposium on Cluster, Cloud and Grid Computing. 811–818.
[7] C. Webster. 2014. From Syntactic & Semantic To Pragmatic Interoperability In Healthcare. https://bit.ly/2PWCiTY

C. *Com base nas definições acima, como você definiria interoperabilidade pragmática?*
____

Parte II

Considere o seguinte cenário hipotético: o motorista deseja que o Bluetooth do seu automóvel seja habilitado (e esteja disponível para chamadas em viva-voz) apenas quando familiares estiverem dentro do automóvel. Não é desejado a ativação do Bluetooth em outras situações (por exemplo, quando amigos do motoristas estiverem dentro do automóvel).

D. *Na sua opinião, há interoperabilidade pragmática no cenário acima?*

◯ *Sim*

◯ *Não*

E. *Caso tenha sinalizado que "sim, há interoperabilidade pragmática no cenário acima", descreva essa interoperabilidade com base na sua definição:* ____________________

Parte III

Considere o seguinte cenário hipotético: um usuário deseja que seu smartphone altere a configuração de tela e mude a iluminação para o modo noturno (com "cores quentes") apenas durante a noite. Não é desejado o modo noturno durante o dia.

F. *Na sua opinião, há interoperabilidade pragmática no cenário acima?*

◯ *Sim*

◯ *Não*

G. *Caso tenha sinalizado que "sim, há interoperabilidade pragmática no cenário acima", descreva essa interoperabilidade com base na sua definição:* ____________________

H. *Caso tenha sinalizado que "sim, há interoperabilidade pragmática no cenário acima", descreva uma situação (i.e., um contexto) que pode ocorrer nesse cenário pragmático:* ____________________

Parte IV

Observações:

1. Responda as questões abaixo considerando o cenário do "smartphone no modo noturno".
2. Entenda "modelagem" como sua descrição textual (definição e contexto) para o cenário do smartphone.
3. Considere (1) como "muito baixo" e (5) como "muito alto".

I. *Pergunta:*

	1	2	3	4	5
q1: Qual o esforço requerido para entender o cenário?	◯	◯	◯	◯	◯
q2: Qual o esforço requerido para identificar possíveis elementos ausentes na modelagem?	◯	◯	◯	◯	◯
q3: Qual o esforço requerido para avaliar a uniformidade dos elementos na sua modelagem?	◯	◯	◯	◯	◯
q4: Qual o esforço requerido para identificar elementos desnecessários na sua modelagem?	◯	◯	◯	◯	◯
q5: Qual o esforço requerido para entender e adicionar um novo contexto?	◯	◯	◯	◯	◯
q6: Qual o esforço requerido para avaliar a completude de sua modelagem?	◯	◯	◯	◯	◯
q7: Qual o esforço requerido para identificar elementos conflitantes e/ou ambiguos na sua modelagem?	◯	◯	◯	◯	◯
q8: Qual o esforço requerido para identificar elementos redundantes na sua modelagem?	◯	◯	◯	◯	◯
q9: Qual o esforço requerido para listar a intenção, a mensagem e o efeito do cenário?	◯	◯	◯	◯	◯

J. *Hora final*: ____________________

C.3 CAPITAL GROUP QUESTIONNAIRE

A. *Nome*: ______________________

B. *Hora inicial*: ______________________

Parte I

Após uma extensa revisão da literatura, nosso grupo de pesquisa definiu interoperabilidade pragmática como segue: "Interoperabilidade pragmática, em nível de sistemas, é alcançada quando ocorre uma compreensão compartilhada do conjunto de condições (intenção) e dos contextos necessários para a comunicação, a fim de proporcionar o uso correto da mensagem e, como consequência, ter um resultado produzido dentro das expectativas comuns (efeito)."

Nossa definição aborda quatro elementos que consideramos fundamentais para que a interoperabilidade pragmática ocorra:

- efeito: aquilo que é produzido, consequência ou resultado;
- contexto: informações necessárias para realizar o efeito (representado por informações contextuais: why, how, when, who, where e what);
- uso: como dados (ou a mensagem) são usados; e
- intenção: o que o remetente espera que o efeito da mensagem seja.

De posse da definição, apresentamos o framework conceitual para interoperabilidade pragmática (chamado de CAPITAL) abaixo.
A interoperabilidade pragmática (PI) é composta por três conjuntos de informações no nível 2:

- "msg", representando a mensagem
- "int", representando a *intenção*
- "effec", representando o *efeito*

O nível 3 detalha os elementos do nível 2:

- "msg" é composta por uma ação (action) e um conjunto de contextos (cont)
- "effec" contém o comportamento externo executado pelo receptor (ext behav)

Ainda no nível 3, a ação (action) é composta pelo nome do serviço (name), entradas (inputs), saídas (outputs), exceções (except) e comportamento (int behav), e cont é um conjunto de contextos similares. Cada contexto específico é representado por "cont$_i$"
O elementos "action" representa o *uso* e o "cont" representa o *contexto*.

Framework CAPITAL:

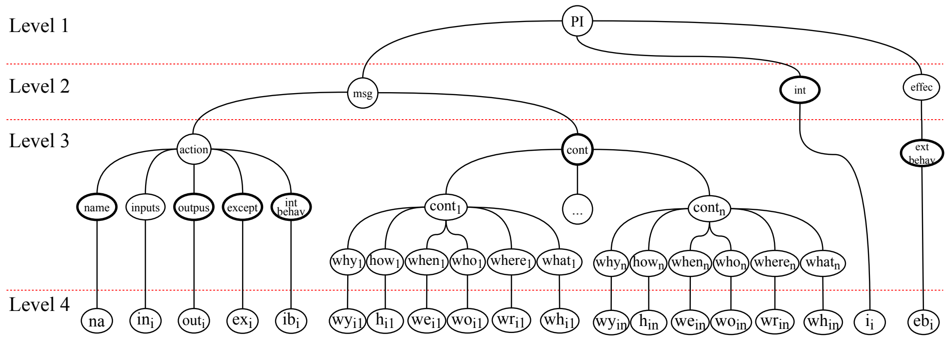


Imagem em tamanho maior:
https://github.com/elivaldolozer/imgExpCAPITAL/blob/master/fig1-CAPITAL.pdf

Parte II

Considere o seguinte cenário hipotético: o motorista deseja que o Bluetooth do seu automóvel seja habilitado (e esteja disponível para chamadas em viva-voz) apenas quando familiares estiverem dentro do automóvel. Não é desejado a ativação do Bluetooth em outras situações (por exemplo, quando amigos do motoristas estiverem dentro do automóvel).
Esse cenário pode ser representado no framework CAPITAL conforme figura abaixo.

Cenário do Bluetooth no framework CAPITAL:

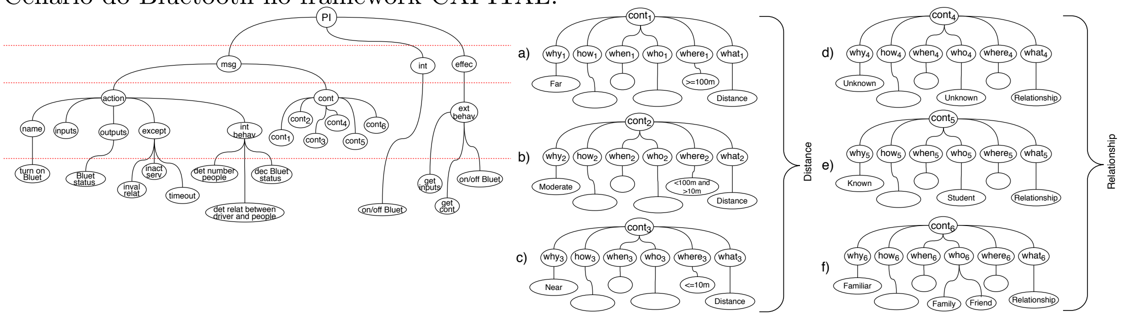


Imagem em tamanho maior:
https://github.com/elivaldolozer/imgExpCAPITAL/blob/master/fig2-cen3CAPITAL.pdf

C. *Na sua opinião, há interoperabilidade pragmática no cenário acima?*

◯ *Sim*

◯ *Não*

D. *Caso tenha sinalizado que "sim, há interoperabilidade pragmática no cenário acima", descreva essa interoperabilidade:* ____________________

Parte III

Considere o seguinte cenário hipotético: um usuário deseja que seu smartphone altere a configuração de tela e mude a iluminação para o modo noturno (com "cores quentes") apenas durante a noite. Não é desejado o modo noturno durante o dia.

E. *Na sua opinião, há interoperabilidade pragmática no cenário acima?*

◯ *Sim*

◯ *Não*

F. *Caso tenha sinalizado que "sim, há interoperabilidade pragmática no cenário acima", modele o cenário acima com o framework CAPITAL:* ____________________

Framework CAPITAL “em branco”

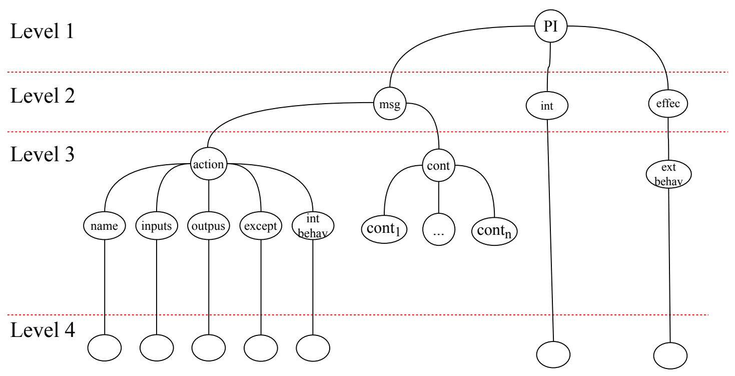


Imagem em tamanho maior:
https://github.com/elivaldolozer/imgExpCAPITAL/blob/master/fig3-CAPITALbranco.pdf

G. *Qual nome (name) você julga apropriado para a ação que esse cenário hipotético pode/deve realizar?* ____________________

H. *Qual(is) entrada(s) (inputs) esse cenário hipotético pode/deve ter para realizar a ação acima?* ____________________

I. *Qual(is) saídas(s) (outputs) esse cenário hipotético pode/deve ter para realizar a ação acima?* ____________________

J. *Qual(s) exceção(ões) (except) esse cenário hipotético pode/deve ter para realizar a ação acima?* ____________________

K. *Que comportamento (int behav) o smartphone pode/deve ter para realizar a ação nesse cenário hipotético?* ____________________

L. *Qual(is) contexto(s) (cont) você julga necessário(s) nesse cenário hipotético?* ____________________

M. *Qual é a intenção (int) do usuário do smartphone?* ____________________

N. *O que o smartphone de fato faz (effec)? Ou seja, qual o resultado final do cenário?* ____________________

Parte IV

Observações:

1. Responda as questões abaixo considerando o cenário do “smartphone no modo noturno”.
2. Entenda “modelagem” como sua respostas para o cenário do smartphone.
3. Considere (1) como “muito baixo” e (5) como “muito alto”.

O. *Pergunta:*

	1	2	3	4	5
q1: Qual o esforço requerido para entender o cenário?	◯	◯	◯	◯	◯
q2: Qual o esforço requerido para identificar possíveis elementos ausentes na modelagem?	◯	◯	◯	◯	◯
q3: Qual o esforço requerido para avaliar a uniformidade dos elementos na sua modelagem?	◯	◯	◯	◯	◯
q4: Qual o esforço requerido para identificar elementos desnecessários na sua modelagem?	◯	◯	◯	◯	◯
q5: Qual o esforço requerido para entender e adicionar um novo contexto?	◯	◯	◯	◯	◯
q6: Qual o esforço requerido para avaliar a completude de sua modelagem?	◯	◯	◯	◯	◯
q7: Qual o esforço requerido para identificar elementos conflitantes e/ou ambiguos na sua modelagem?	◯	◯	◯	◯	◯
q8: Qual o esforço requerido para identificar elementos redundantes na sua modelagem?	◯	◯	◯	◯	◯
q9: Qual o esforço requerido para listar a intenção, a mensagem e o efeito do cenário?	◯	◯	◯	◯	◯

P. *Hora final*: ______________________

C.4 PARTICIPANTS OVERVIEW

Table C.1 depicts the participants involved in this controlled experiment: 36 undergraduate students (Un-st), two developers (Dev), and eight professors (Prof). The students and professors are from two different federal universities (U1 and U2) and the developers work in two different companies (C1 and C2).

We classified the participant with respect to prior knowledge about interoperability (Interop.) and pragmatic interoperability (Prag. Interop.): participant knows the concept and has experience with interoperability or pragmatic interoperability (✓), participant knows the concept but has no experience with interoperability or pragmatic interoperability ($\odot$), participant does not know the concept and has no experience with interoperability or pragmatic interoperability ($\times$).

We grouped the participants into two groups: *Control group* (ConG) and *CAPITAL group* (CapG).

Table C.1: Participants overview

Participant	Institution	Status	Degree	Group	Prior knowledge	
					Interop.	Prag. Interop.
1	U2	Prof	Ph.D.	ConG	⊙	×
2	U2	Prof	M.Sc.	CapG	×	×
3	U2	Prof	Ph.D.	CapG	⊙	×
4	U2	Prof	Ph.D.	ConG	×	×
5	C1	Dev	M.Sc.	CapG	✓	⊙
6	C2	Dev	M.Sc.	ConG	✓	⊙
7	U2	Un-st	–	ConG	×	×
8	U2	Un-st	–	CapG	×	×
9	U2	Un-st	–	CapG	×	×
10	U2	Un-st	–	ConG	×	×
11	U2	Prof	M.Sc.	ConG	×	×
12	U2	Prof	M.Sc.	ConG	×	×
13	U1	Prof	Ph.D.	CapG	×	×
14	U1	Prof	M.Sc.	CapG	×	×
15	U1	Un-st	–	ConG	×	×
16	U1	Un-st	–	ConG	×	×
17	U1	Un-st	–	CapG	×	×
18	U1	Un-st	–	CapG	⊙	×
19	U1	Un-st	–	ConG	×	×
20	U1	Un-st	–	ConG	×	×
21	U1	Un-st	–	CapG	×	×
22	U1	Un-st	–	CapG	×	×
23	U1	Un-st	–	ConG	×	×
24	U1	Un-st	–	ConG	×	×
25	U1	Un-st	–	CapG	×	×
26	U1	Un-st	–	CapG	×	×
27	U1	Un-st	–	CapG	×	×
28	U1	Un-st	–	ConG	×	×
29	U1	Un-st	–	CapG	×	×
30	U1	Un-st	–	CapG	×	×
31	U1	Un-st	–	ConG	×	×
32	U1	Un-st	–	ConG	×	×
33	U1	Un-st	–	ConG	⊙	⊙
34	U1	Un-st	–	CapG	✓	⊙
35	U1	Un-st	–	ConG	⊙	×
36	U1	Un-st	–	ConG	×	×
37	U1	Un-st	–	ConG	×	×
38	U1	Un-st	–	ContG	×	×
39	U1	Un-st	–	CapG	×	×
40	U1	Un-st	–	CapG	×	×

41	U1	Un-st	–	CapG	×	×
42	U1	Un-st	–	ConG	×	×
43	U1	Un-st	–	CapG	×	×
44	U1	Un-st	–	CapG	×	×
45	U1	Un-st	–	CapG	×	×
46	U1	Un-st	–	ConG	×	×

Appendix D

FUNCTION POINT ANALYSIS ADJUSTMENT FACTOR

This Appendix presents Function Point Analysis value adjustment factors. In this part, we present the 14 questions used in the calculation. Each question must be measured with a factor F: $F_i = 0$ indicates that the question i is not important/applicable, and $F_i = 5$ indicates that the question i is absolutely important.

Table D.1 Value adjustment factors

i	Question	Factor F_i					
		0	1	2	3	4	5
1	Does the system require reliable backup and recovery?						$\times$
2	Are data communications required?						$\times$
3	Are there distributed processing functions?		$\times$				
4	Is performance critical?						$\times$
5	Will the system run in an existing, heavily utilized operational environment?					$\times$	
6	Does the system require on-line data entry?						$\times$
7	Does the on-line data entry require the input transaction to be built over multiple?			$\times$			
8	Are the master files updated on-line?					$\times$	
9	Are the inputs, outputs, files, or inquiries complex?					$\times$	
10	Is the internal processing complex?			$\times$			
11	Is the code designed to be reusable?				$\times$		
12	Are conversion and installation included in the design?		$\times$				
13	Is the system designed for different organizations?		$\times$				
14	Is the application designed to facilitate change and use by the user?					$\times$	
				$\sum_{i=1}^{14} F_i$		$=$	**46**