UNIVERSITE DE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis

Mention : INFORMATIQUE

Présentée et Soutenue par

Luca COSTABELLO

Context-Aware Access Control and Presentation of Linked Data

Thèse dirigée par Fabien GANDON

et co-encadrée par Ivan HERMAN

Préparée à l'Inria Sophia Antipolis, Equipe WIMMICS

Soutenue le 29 Novembre 2013

Jury :

Président:	Peter SANDER	-	Université de Nice-Sophia Antipolis
Rapporteurs:	Jérôme EUZENAT	-	Inria (EXMO)
	Stefan DECKER	-	National University of Ireland, Galway
Directeur:	Fabien GANDON	-	Inria (WIMMICS)
Co-encadrant:	Ivan HERMAN	-	CWI, W3C

A Sumi

Acknowledgments

First of all, I would like to express deep gratitude to my advisor Fabien Gandon that assisted and guided me during all the steps of my work. His advices have been invaluable along all my adventure, and go well beyond the pursuit of an academic title and the fulfilment of a research project. Thanks for the personal and professional touch, for the patience, and for teaching me that new perspectives come out of an open mind, and that assumptions can be challenged only by listening and respecting others' ideas first.

I am particularly grateful for the precious advices and suggestions of my co-advisor Ivan Herman, whose experience and support overcame the geographical distance and played a crucial role in the progress of the work.

I would also like to thank all the co-authors of the papers published during my PhD program. In particular, I would like to address special thanks to Serena Villata, whose priceless ideas and boundless enthusiasm encouraged me during these years working together.

I want to thank all the past and present members of the WIMMICS team for their friendly, positive, and helpful attitude. I am particular grateful to Olivier Corby and Alain Giboin, whose suggestions and support has been extremely important during my research activities. Thanks to my fellow PhD students colleagues Rakeb Hasan, Corentin Follenfant, Maxime Lefrançois, Nicolas Marie, Oumy Seye, Zide Meng, Papa Fary Diallo, and Pavel Arapov: working with you guys has been an honour, and I wish you all the best for a future full of achievements. I am grateful to all other members of the team, in particular Elena Cabrio, Julien Cojan, Christophe Desclaux, Guillaume Erétéo, and Nicolas Delaforge for being such good colleagues and friends. I would also like to extend my thanks to the I3S component of the team, especially to Catherine Faron-Zucker, Michel Buffa, Alban Gaignard, Isabelle Mirbel, and Andrea Tettamanzi.

This work would not have been possible without the financial and logistic support of Inria. In this regard, Christine Foggia deserves a special thanks for the continuous support and precious help provided from the very first day of my experience in the team.

Ça va sans dire, nobody has been more important in the pursuit of my PhD than Sumita, my loving and supportive wife, and soon-to-be tender and caring *mommy*.

Abstract

This thesis discusses the influence of mobile context awareness on Web of Data access from handheld devices. The work dissects this issue into three research questions: how to declaratively describe context by complying with Linked Data best practices, how to enable context-aware adaptation for Linked Data consumption, and how to protect access to RDF stores from context-aware devices.

The first challenge is the adoption of a proper context model: the thesis presents the PRISSMA lightweight context vocabulary, a mandatory step for the other contributions of this work.

Content adaptation consists in the process of selection, generation, or modification that produces content units in response to a requested URI. The thesis contribution to this second research activity is PRISSMA. PRISSMA is a presentation-level framework that extends the Fresnel rendering process with context awareness. This is done by selecting the most appropriate RDF presentation according to mobile context. Such operation is performed by an error-tolerant subgraph matching algorithm based on the notion of graph edit distance. The algorithm takes into account the discrepancies between context descriptions and the sensed context, supports heterogeneous context dimensions, and runs on the client-side - to avoid disclosing sensitive context information.

The third research direction addresses triple stores access control in pervasive environments. The contribution of this thesis is the Shi3ld access control framework. Shi3ld adds client context in control enforcement for RDF, thus enabling context-aware access policies. Shi3ld is designed as a pluggable filter for SPARQL endpoints and RDF stores that support HTTP operations on Linked Data. It adopts exclusively Semantic Web languages and reuses existing proposals, thus it does not add new policy definition languages, parsers nor syntax validation procedures. Shi3ld provides protection up to triple level.

The thesis describes both PRISSMA and Shi3ld prototypes. Test campaigns show the validity of PRISSMA algorithm, along with memory and response time performance. The Shi3ld access control module has been tested on different triple stores, with and without SPARQL engines. Results show the impact on response time, and demonstrate the feasibility of the approach.

Contents

1 Introduction **1**

2 Background and Challenges **7**
2.1 Introduction . . . 7
2.2 The Web of Data . . . 8
2.2.1 Towards a "Giant Data Graph" on the Web . . . 8
2.2.2 Linked Data Principles . . . 9
2.2.3 Serving Linked Data . . . 11
2.2.4 Accessing and Consuming the Web of Data . . . 12
2.3 Context Awareness . . . 15
2.3.1 Origins . . . 15
2.3.2 Context Definition . . . 16
2.3.3 Research Themes . . . 16
2.3.4 Applications . . . 18
2.4 Motivations and Challenges . . . 19

3 A Declarative Model for Mobile Context **21**
3.1 Introduction . . . 21
3.2 Ontology-Based Context Models . . . 22
3.3 The PRISSMA vocabulary . . . 25
3.3.1 Design Rationale . . . 25
3.3.2 Vocabulary Description . . . 26
3.3.3 Examples . . . 28
3.4 Conclusions . . . 29

4 Context-Aware Presentation of Linked Data on Mobile **33**
4.1 Introduction . . . 33
4.2 Definitions . . . 35
4.3 Related Work . . . 36
4.3.1 Context-Aware Web Content Adaptation . . . 36
4.3.2 Presentation-level Frameworks for Linked Data. . . . 39
4.3.3 Error-tolerant matching for RDF Graphs . . . 44
4.4 PRISSMA Presentation Framework: Overview . . . 46
4.4.1 Combining Fresnel and PRISSMA Vocabularies . . . 46
4.4.2 The Problem of Prism Selection . . . 50
4.4.3 Resource Rendering . . . 51
4.5 Prism Selection Algorithm . . . 52
4.5.1 Definitions . . . 52
4.5.2 Decomposition . . . 54
4.5.3 Search Algorithm . . . 59

4.5.4 Computational Complexity . . . 62
4.5.5 Cost of Edit Operations . . . 64
4.6 Evaluation . . . 70
4.7 PRISSMA Browser . . . 72
4.8 Conclusions . . . 75

5 Context-Aware Authorization for Graph Stores 79
5.1 Introduction . . . 79
5.2 Access Control Frameworks for Linked Data . . . 81
5.2.1 Access Control for SPARQL . . . 82
5.2.2 Access Control for HTTP Access to Linked Data . . . 84
5.2.3 Context-Aware Access Control . . . 85
5.3 A Model for Context-Aware Access Control Policies . . . 87
5.4 Shi3ld-SPARQL . . . 93
5.4.1 Access Control Enforcement . . . 93
5.4.2 Context Attribute Privacy . . . 96
5.4.3 Evaluation . . . 98
5.5 Shi3ld-HTTP . . . 100
5.5.1 Shi3ld for SPARQL Graph Store Protocol . . . 101
5.5.2 Shi3ld-LDP with Internal SPARQL Engine . . . 101
5.5.3 SPARQL-less Shi3ld-LDP . . . 103
5.5.4 Evaluation . . . 105
5.6 Policy Manager . . . 108
5.6.1 Policy Manager Features . . . 108
5.7 Conclusions . . . 110

6 Conclusions and Perspectives 115
6.1 Summary of Contributions . . . 115
6.2 Limitations and Open Issues . . . 117
6.3 Publications . . . 118
6.4 Perspectives . . . 119

Appendix A PRISSMA Vocabulary 121

Bibliography 125

Chapter 1

Introduction

Mobile access to the Web is daily routine for millions of users and their always-connected devices. New generation networks, optimized user interfaces and ad-hoc interaction paradigms all contribute to the rising popularity of ubiquitous access. Since Linked Data is part of the Web, such factors will influence the way we interact with the Web of Data, and will foster novel mobile scenarios and applications, thus leading to a *Ubiquitous Web of Data*. Ubiquitous Web of Data applications will offer novel ways of consuming and contributing to Linked Data, thanks to the adoption of compelling interaction modalities driven by deeper awareness of the surrounding physical environment (e.g. enhanced interfaces, vocal access to the Web, augmented reality, etc). Such awareness will be the result of detecting the conditions in which Linked Data consumption takes place, thanks to data captured by embedded sensors. Thus, we can prefigure follow-your-nose mobile browsers that deliver audio playback when used by illiterate or blind users, and applications such as augmented reality sightseeing guides that seamlessly reconfigure to adapt Linked Data visualization to the time of the day or to the interests of the current user.

The thesis discusses the impact of mobile context awareness on Web of Data access. In other words, the dissertation addresses the following research question: *how does the mobile context influence Linked Data access and consumption?* The thesis breaks down such question into three sub-problems. The first research activity deals with the design and of a context ontology. In other words, the research question is *how to declaratively describe context by complying with Linked Data best practices?* Second, given that on-board sensors detect the conditions in which Linked Data consumption takes place, we can prefigure mobile applications that *adapt* Linked Data visualization to the surrounding mobile context. Hence, the question: *how to enable context-aware adaptation for linked data consumption?* The third research direction addresses the problem of access control for Linked Data servers. More specifically, the thesis tackles the problem of protecting triple stores when queried by context-aware, mobile devices.

Consider the following scenario: a museum wants to create a Linked Data-powered mobile guide for visitors. Museum and artwork metadata are modelled with RDF and published in an triple store coupled with a SPARQL endpoint. The triple store serves as a backend for the client application, the mobile guide of the museum. A series of stakeholders must be considered when designing such service: artwork metadata dataset administrators, end users, and mobile application developers. Each stakeholder introduces a series of constraints: dataset administrators

must enforce the museum policy of disclosing artwork metadata only to on-site visitors, while museum general information can be accessed from anywhere. Hence, they require RDF access control with context-aware features. Besides, mobile end-users determine the need for optimized, tailored content visualization according to their real-world situations: for instance, when visualizing general information on the museum, tablet-equipped users might be provided with a textual description of the museum, the city, the director, the curator name, and the establishment year. On the other hand, the same museum RDF entity could be presented differently to people strolling in the museum town, using a smartphone. In such context (walking in the museum town with a smartphone) users will benefit more from practical informations such as the museum address, metro station, opening hours, and ticket fees. Moreover, address and public transportation information could be highlighted, since considered more important in the current context. Mobile application developers introduce requirements as well: for example, such content adaptation process could be managed by a third-party presentation engine, that relies on a declarative approach, thus offloading developers from the burden of managing context adaptation in a hard-coded, ad-hoc fashion.

Both RDF adaptation and context-based access control need a proper context model. The Linked Data scenario determines a series of constraints that must be matched by such ontology: first of all, the adoption of a lightweight vocabulary instead of a vast, monolithic context ontology must be envisioned. The open world assumption determines the need for an extensible ontology. Moreover, the model must reuse existing vocabularies components, and be published on the Web according to the Linked Data principles. Such context ontology paves the way for RDF content adaptation, the process of selection, generation, or modification that produces content units in response to a requested URI. This feature is essential for the mobile Web and is driven by the multifaceted notion of client context. Semantic Web mobile applications might not have built-in assumptions about the schemas of the data they consume, as the data model could be unknown a-priori, and provided by heterogeneous sources: users might consume any type of information, as long as it is relevant to their context. These features require content adaptation, such as context-based filtering, ordering, grouping and formatting of triples. Content adaptation reduces the fan-out of linked data connections and provides coherent information by using context as dynamic filter. In other words, it creates "optimized" content units ready for user consumption. The most relevant challenges faced by content adaptation are, first of all to describe context at RDF presentation-level in a declarative way, and second, to select the presentation knowledge that must be activated for a given sensed context. Due to the imprecise and incomplete nature of context data, the task of matching declared to sensed contexts requires an error-tolerant approach. Besides content adaptation, another open issue on the Web of Data is access control: the open nature of the current Web of Data may give providers the impression that their content is not safe, thus preventing further publication of datasets. The awareness of the surrounding context enables interesting features, above all expressive, context-aware access policies. Nevertheless, protect-

ing RDF stores in pervasive environments poses a series of questions, namely how to define a fine-grained, context-aware access control model for graph stores, and how to target both SPARQL endpoints and HTTP operations on Linked Data. Finally, dataset administrators must be assisted in creating and managing access policies.

Although the Linked Data community spent a considerable effort on defining best practices for *publishing* data, and achieved concrete results, promising research on the front of *consuming* such information is still open. A number of works in literature discuss techniques and strategies to query the Web of Data, clean results, assess provenance and trust. Nevertheless, little work has been done to analyse the impact of the rising mobile computing to the Web of Data. In particular, the role of sensor-equipped devices in accessing Linked Data has not been investigated. Yet, accessing data in ubiquitous environments means using devices, equipped with a wide range of sensors, whose information can be useful to improve data consumption and foster novel services and applications. A large number of ontology-based context models have been proposed in the latter years, but for chronological reasons they are far from the Web of Data best practices. None of them follow a lightweight approach (many being monolithic ontologies), little or no interlinking with other vocabularies is present, and many vocabularies are not even published on the Web, thus discouraging the adoption and re-use in the Web community. While a number of recent works deal with adaptation of Web resources on mobile devices, none of them specifically target Linked Data resources. On the other hand, several presentation-level frameworks for Linked Data exist, but none of them address the problem of adaptation to context. As a consequence, no Linked Data work deals with matching real context data with context declarations with an error-tolerant approach. Error-tolerant techniques for RDF matching are mostly related to ontology matching strategies, and off-the-shelf SPARQL engines designed for error-tolerant matching neither support heterogeneous dimensions (such as time and location), nor they are designed for computational-constrained mobile platforms. A number of access control frameworks for Linked Data have been proposed. Unfortunately, none of them matches all our requirements. In particular, no access control model in literature has been designed to support full-fledged context aware policies. Works proposed in the ubiquitous computing community are not Web-based, thus they do not fit our Linked Data scenario.

The thesis includes three main contributions, that answer the aforementioned research questions. First, we designed a core mobile context ontology, where only key concepts are explicitly modelled, leaving refinements and extensions to domain experts. The vocabulary re-uses classes and properties proposed in the semantic web community. We reuse and combine well-known vocabularies: more precisely, we are based on the widely-accepted formalization of context proposed by Dey [Dey 2001] and we extend the W3C Model-Based User Interface Incubator Group[1] proposal where mobile context is described as an encompassing term, an information space defined as the sum of three main axis: the mobile *User* model, the *Device* features

[1] http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui/

and the *Environment* in which the action is performed. The challenge of context-aware presentation for Linked Data brought to the design of PRISSMA. PRISSMA consists in a Fresnel-based [Pietriga 2006] RDF rendering engine that selects the most appropriate presentation of RDF triples according to mobile context. In other words, PRISSMA extends the Fresnel rendering process with context awareness by adding a preliminary step, the *presentation data selection* The problem posed by the second research question is answered with two main contributions: i) a lightweight ontology for describing context conditions in a declarative, Fresnel-compatible style, and ii) an error-tolerant algorithm to determine whether the sensed context is compatible with context declarations, hence the most appropriate data pre-processing and visualization can be activated. The algorithm must satisfy a series of requirements: first, the intrinsic imprecision of contextual data determines the need for an error-tolerant strategy that takes into account possible discrepancies between context descriptions and the actual context. Second, this error-tolerant mechanism must support heterogeneous context dimensions (e.g. location, time, strings). Third, since the procedure must run on the client-side - to avoid disclosing sensitive context information - we must design a mobile-friendly algorithm, with acceptable time and space complexity. Finally, the adopted strategy must support runtime updates of RDF graphs, as context descriptions might be fetched from remote repositories and added to the selection process at runtime, and the sensed context may change at any time. The Shi3ld framework is the contribution of this thesis to the issue of access control for Linked Data. Shi3ld comes in two flavours: Shi3ld-SPARQL, designed for SPARQL endpoints, and Shi3ld-HTTP, designed for HTTP operations on triples. Shi3ld-SPARQL protects RDF stores by changing the semantics of the incoming SPARQL queries, whose scope is restricted to triples included in accessible named graphs only. The list of accessible graphs is determined by evaluating pre-defined access policies against the actual mobile context of the requester. The Shi3ld authorization framework for HTTP derives from the SPARQL scenario, and has been designed to work in conjunction with the SPARQL 1.1 Graph Store HTTP Protocol and the emerging Linked Data Platform.

The thesis contribution provides a number of advances, and paves the road for the Ubiquitous Web of Data. The proposed context model is domain-independent and supports both content adaptation and context-aware access control for RDF stores. The ontology complies with Linked Data best practices: it respects the open world assumption, since it allows extensions. Furthermore, in the light of the Web of Data philosophy, it does not provide an exhaustive set of context classes and properties: the vocabulary delegates extensions to domain specialists who can adapt the vocabulary to their needs (e.g. indoor location). The PRISSMA framework enables context-based adaptation of resources fetched from Linked Data. It offers full backward compatibility with Fresnel, thus supporting any Linked Data access strategy. Relying on Fresnel favours the sharing and reuse of prisms across applications, and does not introduce new formalisms other than RDF. The error-tolerant algorithm that selects the most appropriate visualization according to the context features a compact index and has a sublinear dependence on the number of possible visual-

izations. Hence, it can be run on resource-constrained mobile devices even in the presence of many context declarations. Operating on the client side guarantees privacy preservation, because context data does not have to be disclosed to third-party adaptation servers. Moreover, the index supports incremental updates, thus allowing on-the-fly addition of context declarations. Beyond the support for context in control enforcement, the Shi3ld access control framework has the advantage of being a pluggable filter for generic triple stores (with or without SPARQL interface), with no need to modify the endpoint itself. It adopts exclusively Semantic Web languages and reuses existing proposals, thus it does not add new policy definition languages, parsers nor syntax validation procedures. Shi3ld provides protection up to triple level.

The thesis is organized as follows: Chapter 2 provides an overview of the the Web of Data and of Context Awareness, along with a list of challenges. Chapter 3 describes the first contribution of the thesis, the PRISSMA lightweight vocabulary, i.e. the ontology used to represent context data. The second main contribution is presented in Chapter 4: this part of the thesis describes the design rationale, and the evaluation campaign of the PRISSMA presentation engine. Chapter 5 deals with the other main research question of the thesis, i.e. *how to protect triple stores in ubiquitous and pervasive environments*. Hence, the chapter contains the description of the Shi3ld context-aware access control framework, the third contribution of the thesis. Finally, Chapter 6 presents the summary of contributions, along with a discussion on open issues. The chapter includes a debate about future perspectives on enhancing Web of Data access with context awareness.

Chapter 2

Background and Challenges

Contents

2.1 Introduction 7
2.2 The Web of Data 8
2.2.1 Towards a "Giant Data Graph" on the Web 8
2.2.2 Linked Data Principles 9
2.2.3 Serving Linked Data 11
2.2.4 Accessing and Consuming the Web of Data 12
2.3 Context Awareness 15
2.3.1 Origins 15
2.3.2 Context Definition 16
2.3.3 Research Themes 16
2.3.4 Applications 18
2.4 Motivations and Challenges 19

2.1 Introduction

The Web has evolved from an information space for sharing textual documents into a medium for publishing structured data, among many other resources. The Linked Data initiative aims at fostering the publication and interlinking of data on the Web, giving birth to the Web of Data, an interconnected global dataspace where data providers publish their content publicly. At the same time, ubiquitous connectivity enables new scenarios in consuming and contributing to the Web of Data: in the latter years the importance of context has increased, as mobile devices such as smartphones have become more pervasive. The Mobile computing community introduced the concept of Context Awareness, a research activity that encourages the use of environment information to adapt mobile services and applications.

This chapter provides an overview of the Web of Data basic principles and main features. Besides, it introduces the concept of context and of context-awareness, two important facets of ubiquitous computing research. This background information is useful to explain the challenges and the motivations of this thesis, a work designed to enhance the interaction with the Web of Data from mobile devices using context-awareness principles.

2.2 The Web of Data

2.2.1 Towards a "Giant Data Graph" on the Web

In the latter years, a growing number of institutions, governments, and companies have been making data available on the Internet. Such information spans a multitude of topics, such as pollution level in urban areas, public transportation timetables, product catalogues, etc. This data is accessed by third parties, either free of charge[2] or behind paywalls[3]. Third parties consume data by building new business models, discover hidden knowledge by mixing datasets, nurture socially-relevant trends such as data-journalism and government accountability, and so on. The Web is the natural candidate platform for both publishing and consuming these data. Indeed, the Web has evolved from a document-oriented space of HTML pages to a *Web of Data*, a global dataspace of interconnected data entities.

Semantic Web[4] researchers have been dealing with a number of questions stemmed from building and nurturing such Web of Data, i.e. how to publish data so that reuse is encouraged, or how to foster data integration from large number of potentially unknown data sources. To achieve these results, two key factors have been taken into account: first, sharing and reuse is made possible only if data is *structured*. Existing proposals such as microformats or Web APIs are either too domain specific, or need ad-hoc consumption techniques. Second, data integration and discovery is possible only when a shared data model is adopted across systems, along with common data schemas. For instance, the Web APIs paradigm is riddled by the "data silo" problem, i.e. set of APIs expose single datasets, but no external datasets connections are provided, thus losing an appealing feature of the Web, the hyperlinks between entities. The Semantic Web community tackles such issues, and contributed to the birth of *Linked Data*[5] [Berners-Lee 2006a, Heath 2011], a term that identifies practices aimed at fostering the publication and interlinking of data on the Web, thus nurturing the aforementioned global information space known as the Web of Data. Linked data techniques favours information sharing and reuse, and they consist in concrete solutions to overcome the issues of vertical data silos by promoting a network of interconnected datastores on the Web. [Heath 2011][6].

Efforts in the Semantic Web community, notably around Linked Data techniques, brought to a real-world deployment of the Web of Data. This growing network of datasets is deployed in the current Web, and consists in hundreds of interlinked datasets. A popular (non-exhaustive) graphic representation of the Web of Data is the *Linked Data Cloud* diagram (Figure 2.2). The figure shows a number of interconnected datasets adopting Linked Data techniques. As seen in the

[2] http://datahub.io/

[3] http://www.opencalais.com/ needs a subscription for professional use.

[4] http://www.w3.org/standards/semanticweb/

[5] http://linkeddata.org/

[6] An in-depth analysis of the benefits of the Web of Data for data producers and consumers is out of the scope of this thesis. The reader is encouraged to read the comprehensive book by Heath and Bizer [Heath 2011].

picture, datastores belong to different thematic domains (academic publications, user-generated content, government data, life science, etc). Recent figures estimate the size of the Web of Data in several billion data statements (known as RDF triples, see Section 2.2.2) [Heath 2011].

2.2.2 Linked Data Principles

Before dealing with Linked Data features, it is useful to introduce a number of technologies used in the Web of Data. As shown in Figure 2.1, such items are a subset of Semantic Web technologies.

HTTP Universal Resource Identifiers (URI) HTTP URIs[7] generalize Universal Resource Locators (URLs). Instead of referring to Web pages only, URIs identify any kind of resource (e.g. real-world objects), thus acting as a general-purpose namespace mechanism.

Resource Description Framework (RDF). RDF[8] is a data model based on directed labelled graphs. Coupled with URIs, it acts as a core component of the Semantic Web. The atomic piece of knowledge in RDF is the *triple*, a (subject, predicate, object) construct.

Resource Description Framework Schema (RDFS). RDFS[9] is a general-purpose language to define lightweight RDF vocabularies. RDFS provides the notion of class and property for a resource, along with the domain and the range of a relationship. A RDFS vocabulary can contain subclasses and subproperties.

Ontology Web Language (OWL). If RDFS expressivity is not sufficient, Web of Data vocabularies can be designed with OWL[10], a formal ontology language for the Semantic Web. OWL supports features such as class intersection, union and cardinality restriction.

SPARQL is the Query and Update language for the Web of Data[11]. Query services adopting SPARQL are called *SPARQL endpoints* and lie on top of an RDF knowledge base.

In a 2006 note [Berners-Lee 2006a], Tim Berners-Lee relies on the aforementioned languages to describe four best practices for data publication on the Web, known as the *Linked Data Principles*. Note that, although the principles extend the Web, they share the same design features as the Web of documents. The principles are the following:

[7] http://www.ietf.org/rfc/rfc2616.txt
[8] http://www.w3.org/standards/techs/rdf
[9] http://www.w3.org/TR/rdf-schema/
[10] http://www.w3.org/standards/techs/owl
[11] http://www.w3.org/standards/techs/sparql

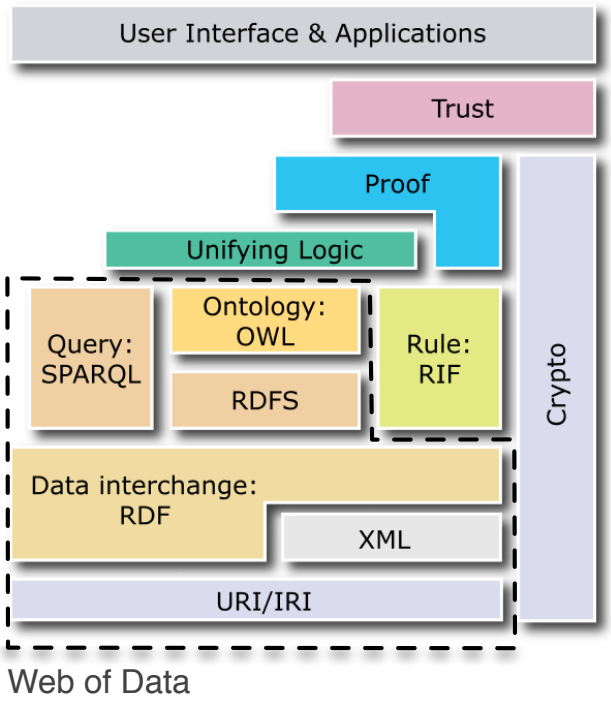


Figure 2.1: The Semantic Web stack. The dotted area includes the technologies used in the Web of Data.

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL, HTML, HTTP content negotiation, etc).
4. Include links to other URIs, so that they can discover more things.

The first principle states that URIs must be used to name "things". Unlike traditional Web, in the Web of Data URIs are used to identify real world objects (`http://example.org/john-doe`), abstract concepts (e.g. `http://dbpedia.org/resource/Speed`), and relationships between objects or resources (e.g. being friend with someone). HTTP is the Web cornerstone. Combined with URIs, the HTTP protocol provides a well-established mechanism for retrieving resources. The second Linked Data principle recommends the combined use of HTTP and URIs to ease the dereferencing of the desired resource. The third principle regulates the dereferencing mechanism of an URI, by proposing the adoption of the RDF data model, and of HTTP content negotiation. The fourth Linked Data principle underlines the need for linking resources to others, mimicking the classic Web hyperlinks in HTML pages. Links connecting resources are typed (using RDF), thus unlimited types of relationships might be created (e.g. the resource

`http://example.org/john-doe` might be friend of `http://example.org/alice`). Note that RDF links may interconnect resources hosted in different datasets, thus creating a global interconnected data space, the Web of Data.

2.2.3 Serving Linked Data

Describing the best practices for publishers on the Web of Data is out of the scope of this thesis. Nevertheless, this Section contains a list of well-established approaches that should be followed in exposing Linked Data to client applications. There exists two main strategies to expose Linked Data: running a SPARQL endpoint, and serving RDF over HTTP.

SPARQL endpoints. Linked Data providers might want to provide a standardized search API over their exposed triples. As mentioned earlier in the chapter, the SPARQL query language has been designed for this purpose. Such API is called SPARQL endpoint, and processes queries written in the SPARQL language. Such queries will be executed on the underlying RDF dataset. This is typically stored in a *triple store*, i.e. a database engine specifically designed for RDF triples. Nevertheless, SPARQL queries could be issued to services that convert relational database tuples into triples. Recent SPARQL upgrades (SPARQL 1.1) allow read/write interaction with triples[12].

Serving RDF over HTTP. Serving RDF triples relying only on HTTP operations is a pattern closer to the original Web architecture. Data providers that adopt this pattern often publish data coming from static RDF files and hosted in regular Web servers. Nevertheless, data might be served by wrapping SPARQL endpoints with Web APIs (e.g. SPARQL HTTP Graph Store Protocol[13]), or by triple stores exposing data directly on HTTP, without the SPARQL layer. Indeed, the Semantic Web community is recently emphasizing the need for a substantially *Web-like* interaction paradigm with Linked Data. For instance, the W3C Linked Data Platform initiative (LDP)[14] promotes the use of read/write HTTP operations on triples, thus providing a *basic profile* for Linked Data servers and clients. According to W3C, the Linked Data Platform initiative is a *set of best practices and simple approach for a read-write Linked Data architecture, based on HTTP access to Web resources that describe their state using the RDF data model*. In other words, the recommendation specifies the behaviour of Linked Data servers when exposing RDF on the Web. This approach dovetails with SPARQL and envisions a more direct access to the data. The Linked Data Platform is envisioned as an enterprise-ready collection of standard techniques and services based on using RESTful APIs and the W3C Semantic Web stack. LDP applications can be developed and deployed using only RDF and conventional HTTP infrastructure, and other elements of the stack can be included, i.e., RDFS, SPARQL, OWL, RIF, and the Provenance vocabulary. The Linked Data Platform defines atomic resources (Linked Data Platform Resources

[12] `www.w3.org/TR/sparql11-query/`
[13] `http://www.w3.org/TR/sparql11-http-rdf-update/`
[14] `http://www.w3.org/TR/ldp/`

- LDPRs) and resource containers (Linked Data Platform Containers - LDPCs). Both are HTTP resources, modelled with RDF. The LDP initiative specifies how to access, modify, create or delete LDPRs and LDPCs with HTTP operations, and describes how LDP servers must accept and process such requests.

2.2.4 Accessing and Consuming the Web of Data

A number of applications of various nature access the Web of Data[16], thus consuming the data published by Linked Data providers. Examples of such applications include general-purpose Linked Data browsers - applications that mimic the features of classic Web browsers (thus allowing users to navigate links from one resource to another), e.g. Tabulator [Berners-Lee 2006b], Haystack [Quan 2003], Longwell[17], IsaViz[18], etc. Other applications might consist in services such as Web of Data search engines (e.g. Sindice[19]). Besides, a number of mobile applications that use Linked Data have been released. DBpedia Mobile [Becker 2009] is one of the first attempts to bring the Web of Data on mobile. Designed as a location-centric, mobile Web application for tourists, DBpedia mobile consists in a map-based visualization of point of interests (POIs), retrieved from DBpedia[20]. David et al. present an Android implementation of a component aimed at ease mobile applications interoperability [David 2010]. Mobile devices are a relevant valuable repository of personal user information, stored in isolated applications, thus suffering from the problem of data silos. The goal is therefore to let each mobile application expose its data to other applications. RDF representation has been chosen and the Web of data paradigm has been embraced. Information can be therefore exchanged between applications without knowing the schemas in advance. Moreover, RDF data could be directly published on the Web of data. Le-Phuoc et al. present *RDF On the Go*, a mobile prototype featuring an on-board, full-fledged RDF storage and a SPARQL query processor [Le Phuoc 2010]. Processing data on mobile side is meant to ease computation scalability, address privacy concerns, and reduce transmission costs. The prototype has been developed on Jena and the ARQ Semantic Web Toolkit and it shares the same APIs of their full-fledged counterparts. The local RDF store is based on a scaled-down Berkeley DB version. RDF On the Go supports standard and spatial SPARQL queries. Links between Augmented Reality and Linked Data have been proposed in a position papers by Reynolds et al. [Reynolds 2010].

Heath and Bizer describe the anatomy of a typical Linked Data application [Heath 2011]. Figure 2.3 shows the main components, i.e. the operations performed on fetched data by a Web of Data application before executing the application-specific business logic.

[15] http://lod-cloud.net/

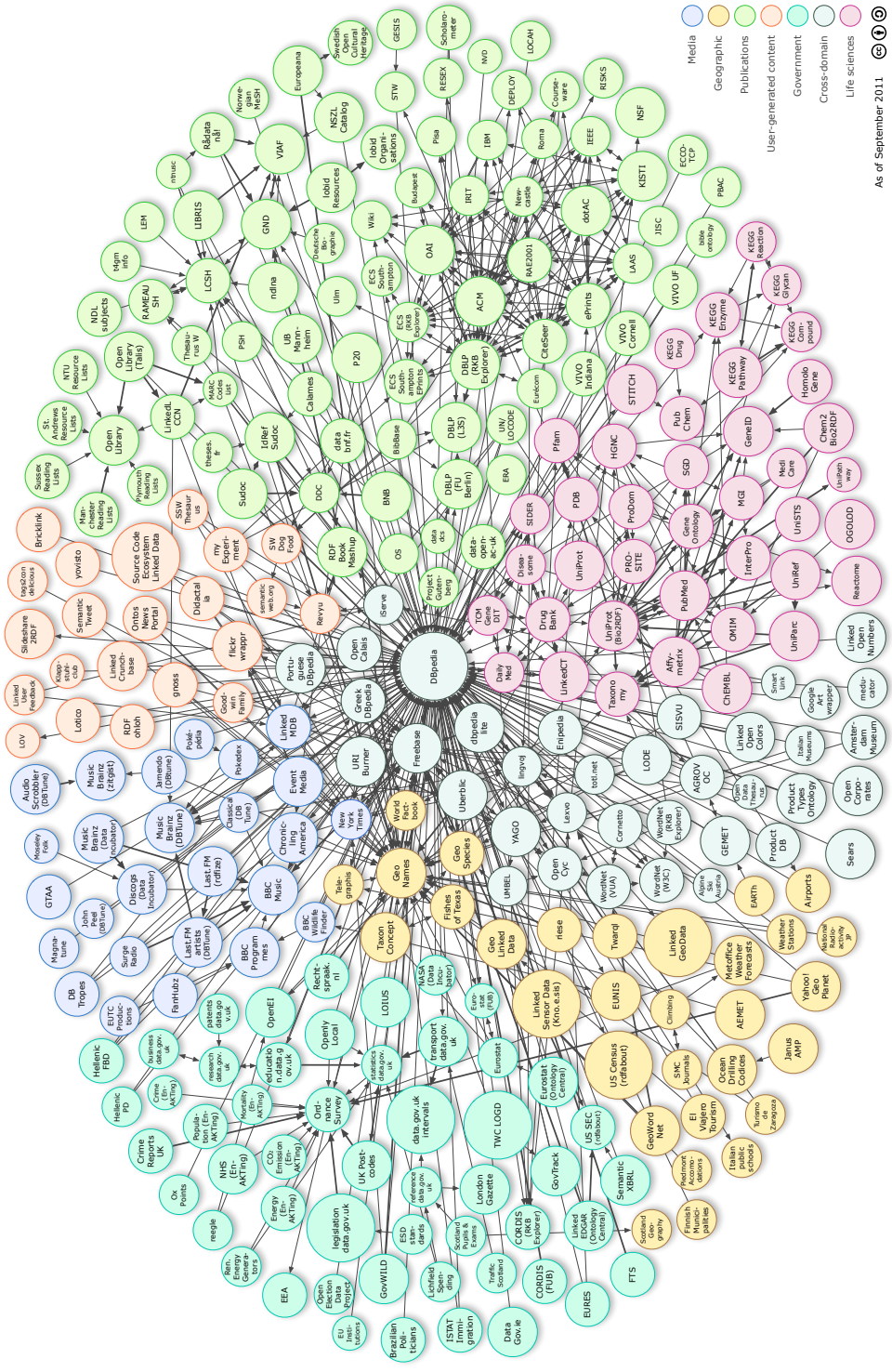
[16] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/Applications

[17] http://simile.mit.edu/longwell/.

[18] http://www.w3.org/2001/11/IsaViz/

[19] http://sindice.com/

[20] DBpedia is the RDF-ized version of Wikipedia, http://dbpedia.org


Media
Geographic
Publications
User-generated content
Government
Cross-domain
Life sciences
As of September 2011
DBpedia
Linked Open Numbers
Linked GeoData
Linked Sensor Data (Kno.e.sis)
Linked Open Colors
data.gov.uk intervals
legislation data.gov.uk
London Gazette
Eurostat
US Census (rdfabout)
US SEC (rdfabout)
Semantic XBRL
Yahoo! Geo Planet
Sears
Airports
Weather Stations
Metoffice Weather Forecasts
Janus AMP
Ocean Drilling Codices
Reactome
OGOLOD
Homolo Gene
Chem2 Bio2RDF
KEGG Reaction
KEGG Glycan
KEGG Enzyme
KEGG Pathway
KEGG Drug
KEGG Compound
Product Types Ontology
Product DB
Open Corporates
Amsterdam Museum
Google Art wrapper
Linked Open Colors
Open Cyc
Yago
Freebase
Uberblic
Geo Species
Geo Names
Taxon Concept
Lexvo
Lingvoj
Eurostat (Ontology Instance)
Eurostat
Twarql
riese
Fishes of Texas
Taxonomy
LOCAH
RISKS
NSF
KSTI
JISC
PBAC
ECO-TCP
Course-ware
DEPLOY
IBM
Roma
IEEE
RESEX
Pisa
NVD
Scholarometer
GESIS
STW
Europeana
Swedish Open Cultural Heritage
Norwegian MeSH
NSZL Catalog
VIAF
Rådata nå!
lobid Organisations
lobid Resources
GND
LIBRIS
LCSH
RAMEAU SH
NDL subjects
Open Library (Talis)
LinkedL CCN
Open Library
Idref Sudoc
Sudoc
DDC
BNB
RDF Book Mashup
OS
Project Guten-berg
DBLP (RKB Explorer)
DBLP (FU Berlin)
DBLP (L3S)
Eurécom
CiteSeer
ACM
OAI
RAE2001
ePrints
VIVO Cornell
VIVO Indiana
VIVO UF
LAAS
dotAC
New castle
IRIT
Budapest
Wiki
Ulm
ndlna
UB Mann-heim
P20
PSH
Calames
data bnf.fr
Bricklink
Source Code Ecosystem Linked Data
yovisto
Semantic Tweet
Slideshare 2RDF
Linked User Feedback
LOV
Lotico
RDF ohloh
Audio Scrobbler (DBTune)
Music Brainz (zitgist)
Jamendo (DBTune)
Poké-pédia
Pokedex
Linked MDB
Event Media
URI Burner
Classical (DB Tune)
Chronic-ling America
New York Times
Last.FM (rdfize)
Last.FM artists (DBTune)
BBC Music
BBC Program mes
BBC wildlife Finder
Discogs (Data Incubator)
Music Brainz (Data Incubator)
Music Brainz (DBTune)
Magna-tune
John Peel (DBTune)
Surge Radio
DB Tropes
Peel (dbtune)
Moseley Folk
GTAA
EUTC Produc-tions
Hellenic FBD
Hellenic PD
Crime Reports UK
Ox Points
Crime (En-AKTing)
Popula-tion (En-AKTing)
NHS (En-AKTing)
Mortality (En-AKTing)
CO2 Emission (En-AKTing)
Energy (En-AKTing)
Ord-nance Survey
reegle
Renew-able Energy Genera-tors
EEA
Open Election Data Project
EU Institu-tions
Brazilian Poli-ticians
ISTAT Immi-gration
Data Gov.ie
Scotland Pupils & Exams
Traffic Scotland
Lichfield Spend-ing
CORDIS (FUB)
EURES
FTS
GovWILD
ESD stan-dards
reference data.gov.uk
CORDIS (rkb Explorer)
Linked EDGAR (Ontology Gateway)
Scotland Geo-graphy
Finnish Munici-palities
Portu-guese DBpedia
Greek DBpedia
Open Calais
Drug Bank
Daily Med
UniProt (Bio2RDF)
UniProt
Disea-some
SIDER
PDB
ProDom
PRO-SITE
Gene Ontology
HGNC
Pfam
STITCH
SGD
Pub Chem
GeneID
MGI
Medi Care
UniSTS
InterPro
UniRef
UniParc
OMIM
ChEMBL
Affy-metrix
PubMed
Taxono my
LinkedCT
KEGG
Enzyme
Linked Open Numbers
Smart link
Google Art wrapper
medi-cator
Amster-dam Museum
Linked Open Colors
Open Corpo-rates
Product Types Ontology
Product DB
Sears
Airports
Weather Stations
Metoffice Weather Forecasts
Janus AMP
Ocean Drilling Codices
Reactome
OGOLOD
Homolo Gene
Chem2 Bio2RDF
Unichem
Medi Care
UniSTS
InterPro
UniRef
UniParc
OMIM
ChEMBL
Affy-metrix
PubMed
Taxono my
LinkedCT
Enipedia
dbpedia lite
tcm (dbtune)
Daily Med
Disea-some
SIDER
PDB
ProDom
PRO-SITE
Gene Ontology
HGNC
Pfam
STITCH
SGD
Pub Chem
GeneID
MGI
KEGG Pathway
KEGG Drug
KEGG Compound
KEGG Enzyme
KEGG Glycan
KEGG Reaction

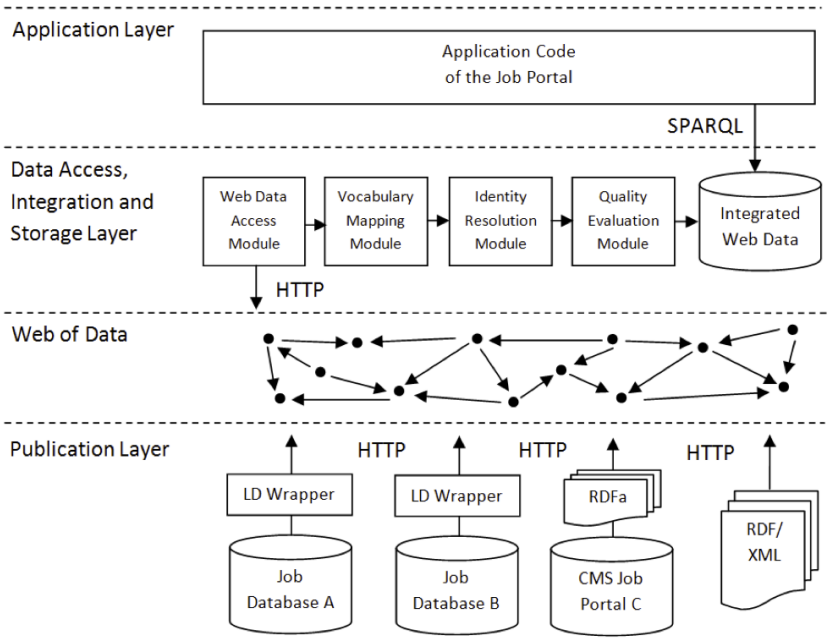


Figure 2.3: The architecture of a Linked Data client application [Heath 2011]

Access Module. Heath and Bizer [Heath 2011] mention three different strategies for accessing the Web of Data from applications. Such techniques are compared and evaluated by Hartig and Langegger in [Hartig 2010] according to the number of data sources, the required data freshness, response time, and the need for runtime data discovery. Applications might access the Web of Data with the following strategies:

Crawling. Applications that adopt this strategy crawl the Web of Data beforehand by dereferencing RDF resources and following RDF links. A local dump of the desired portion of Web of Data is performed. Crawling provides faster response time (the application works on a local copy of data), but suffers from freshness problems and for the need to replicate data.

On-The-Fly Dereferencing. If adopted, applications perform HTTP dereferencing of RDF resources at runtime. This strategy guarantees to operate on up-to-date, non-replicated data, but suffers from potentially long response time and a significant network usage.

Query Federation. This strategy uses SPARQL queries instead of HTTP access. Applications send queries (or part of queries) to the desired SPARQL endpoint, chosen among a pre-determined, application-dependent set.

Vocabulary Mapping. Linked Data applications might query different datasets, thus fetching the same kind of triples, but described by different schemas. This

determines the need to *align* vocabularies, i.e. associate similar classes and properties. For a complete analysis of this problem, see the book by Euzenat and Shvaiko [Euzenat 2007].

Identity Resolution. When querying different datasets, entities might be identified by more than one URIs, even if the referred entity is the same. The role of this component is to group triples associated to the same logical resource (e.g. using `owl:sameAs` properties).

Quality Evaluation. Fetched data might origin from untrusted sources (e.g. Hasnain et al. describes the threats of Linked Data spam in [Hasnain 2012]). Provenance management must therefore be handled properly, according to the application needs.

2.3 Context Awareness

2.3.1 Origins

In a 1991 paper, Mark Weiser introduces the concept of *Ubiquitous Computing* [Weiser 1991]. Weiser's vision prefigures a computing paradigm where electronic devices becomes so omnipresent and necessary that they "*disappear*", *weaving themselves into the fabric of everyday life until they are indistinguishable from it* [Weiser 1991]. In Weiser's vision, electronic devices vanish from a cognitive point of view, meaning that users need reduced attention levels to operate them. To disappear, devices and their applications must "blend" with the physical reality by adopting a proactive behaviour. Such feature cannot be achieved without the consciousness of the surrounding context, i.e. *context awareness*. In the latter years, consumer electronics evolved towards Mark Weiser's vision. Pervasive access and pocket-size, sensor-rich devices have deeply changed networking, interaction paradigms, and user habits. A crucial aspect has been the introduction of sensor-packed mobile devices, appliances that paved the way for context awareness.

Context Awareness has been first introduced by Schilit, Adams, and Want in a 1994 paper [Schilit 1994]. The idea is to create systems that react and adapt to the sensed context, thus providing tailored services and customized behaviours to users (e.g. context-based search, content adaptation, etc.). Context awareness is a multi-faceted research area, spanning from networking (e.g. mobile networks, sensor networks) and hardware design (e.g. onboard sensor performance), to ubiquitous computing, algorithm design, and knowledge representation. Although context awareness does not pose restrictions on the nature of adopted computing devices, in the latter years the focus has shifted towards popular mobile devices like smartphones and tablets. Broadband access networks and growing device capabilities, such as computation power, storage, screen quality, and sensors, contributed to make mobile devices the reference platform for context-aware services. For what concerns sensors, modern smartphones embed a GPS transceiver, Wi-Fi and Bluetooth interfaces (useful for indoor location techniques), cameras, microphones, near

field sensors, light sensors, accelerometer, compass, magnetic field sensors, etc. Aside from sensor abundance, another important factor is the introduction of developer-friendly operating systems (such as iOS and Android), that lowered the bar in the required application development effort. Lovett et al. define this trend *mobile context awareness*, i.e. a computing paradigm where context sensing and the proactive reaction are performed by the device itself, without the need of middleware infrastructure. [Lovett 2012].

2.3.2 Context Definition

Ubiquitous computing literature does not provide a strict, universally accepted definition of context. Indeed, this notion varies according to scenarios, services, and applications. Nevertheless, context-awareness research converged on the popular context definition provided by Anind K. Dey in 2001 [Dey 2001]:

Definition 1 (Context) *Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

Dey's definition is sufficiently wide to embrace a large amount of information, and it is sufficiently general to withstand technology enhancements in portable devices hardware equipment. According to Dey, a large number of information might be included in context description, e.g. location, nearby entities, user activity, time, user preferences, etc. These dimensions can be instantiated with a wide range data, e.g. location can be represented with geographical coordinates (latitude and longitude), or described by user-defined categories and labels (e.g. "office", "Inria"). Context is therefore an ambiguous concept, and it depends on user perception and interpretation.

2.3.3 Research Themes

A number of research themes gather around the notion of context awareness, as explained by Hong et al. in their comprehensive literature review and classification of context-awareness research topics [Hong 2009]. Indeed, the most relevant subjects are gathered in the following categories:

Context Modelling. Context-aware systems must store context information in a machine-readable data, thus schemas for context data (i.e. context models) are required. Over the years, the context awareness community proposed many context modelling paradigms: approaches to this problem vary considerably, according to scenarios and requirements. A number of context models surveys have been published in literature, including works focused exclusively on ontology-based solutions. Such surveys underline the role of ontology-based knowledge representation. In his context-aware systems survey, Baldauf provides a list of context-aware architectures, along

with their models [Baldauf 2007]. Although the survey compares context-aware systems under multiple aspects, a section is dedicated to the paradigms adopted by context models. Interestingly, the majority of context models relies on an RDFS/OWL ontology. Similar conclusions are provided by Perera et al. [Perera 2013], in their comprehensive analysis of context aware systems, that includes a comparison of context models paradigms. Strang and Linnhoff-Popien compare approaches to modelling context, ranging from simple key-value models to markup schemes, object-oriented models and logic-based methods [Strang 2004]. Among the proposed methods, ontology-based models stand out and are appreciated for their expressiveness and the support for heterogeneous data. According to the aforementioned surveys, ontology-based context models provide a fair tradeoff between expressiveness and flexibility.

Sensing. Working at sensor level means dealing with heterogeneous issues, ranging from hardware design to information acquisition and datamining. Working with sensor-related issues determines the need to deal with sensor errors and battery efficiency. The definition of *sensor* overcomes the physical sense: "virtual sensors" might consists querying users calendar or emails. Many context information might be the result of onboard processing to extract refined information from raw sensed data, thus creating "logical sensors" (e.g. an "activity" sensor might consist in coupling calendar data and location information to infer that the user is at work) [Baldauf 2007].

Context Matching. In context aware applications, information originated by sensors must be matched to a set of pre-determined context descriptions. Such declarations describe the contexts that trigger applications actions and typically contain a set of context features, that comply with a pre-defined model. Such features must be matched to real-world data captured by sensors: this operation is normally executed by data mining algorithms (e.g. classifiers, association rule extraction), machine learning algorithms, or other strategies such as nearest neighbour matching. The strategy is chosen according to the nature of the adopted data model. The validity of context matching algorithm is assessed by analysing precision and recall, using a set of pre-defined test context description.

Privacy and Security. Context awareness deals with sensitive information. Location, nearby entities or current activity are personal data, that must be processed, transmitted, and stored securely and in a privacy-preserving manner. Indeed, this is a very active research direction, as the amount of sensed context-related personal data increases. Recent surveys describe strategies to introduce privacy in location-based and context-aware services [Duckham 2010, Krumm 2009].

Infrastructure and middlewares. A significant part of the context-aware community has been working on middlewares for context-aware systems. Describ-

ing such systems is out of the scope of this thesis (a detailed comparison of frameworks is presented in the survey by Baldauf et al. [Baldauf 2007]). An example of such frameworks is the context management infrastructure proposed by Euzenat, Pierson, and Ramparany [Euzenat 2008]. The framework is designed to enable and support pervasive applications in context-aware environments, and features a distributed architecture where each device embeds a context management component. The authors model context with RDF and adopt an ontology-based context representation. However, instead of defining an ad-hoc context model, the system is able to support a number of general-purpose, OWL-based context ontologies, thus resulting effective in heterogeneous environments. Such result is achieved with the use of ontology alignment techniques.

Evaluation techniques. As suggested by Hong et al. [Hong 2009], assessing the utility and the advantages of context-aware services and applications requires both user interface (UI) evaluation and usability studies. UI tests include, among all, A/B testing or multivariate testing, used to assess adaptation to mobile context. Cui et al. adopts such techniques on context-adaptive interface evaluation and discuss the related issues and pitfalls [Cui 2010]. Usability studies involve field campaigns with real users and user interviews. Work by Adipat et al. compares the perceived ease of use and the perceived usefulness of context-adapting systems [Adipat 2011]. Their experience shows that assessing the cognitive load on context-aware services users need hybrid approaches.

2.3.4 Applications

Although providing a comprehensive list of context-aware applications is out of the scope of this thesis, the following list provide a snapshot of popular application categories, and shows examples of context-aware services, as described by Hong et al. [Hong 2009].

Incoming notifications filters. Context-awareness enables smarter ways of dealing with asynchronous communications. The knowledge of current situation can limit the intrusiveness of push-like messages, i.e. by postponing incoming messages and calls, thus delivering improvements known as *context-aware communication* [Schmidt 2000].

Automatic tagging for user-generated content. Context awareness has been recently used to enrich user content (e.g. pictures, status updates, videos, etc.) with information related to current context, such as location, nearby buddies, and event details. Such metadata is automatically attached to content, thus obtaining in richer user generated content, without the burden of adding metadata manually.

Context-aware resource management. Context-awareness can be exploited to manage mobile device limited resources, such as battery level. Thus, energy-efficiency frameworks may choose to activate or deactivate certain network interfaces, sensors, reducing screen brightness, etc. Such operations must consist in a tradeoff between effectiveness and transparency to users, as explained by Lovett et al. [Lovett 2012].

Adaptive user interfaces. The idea of context-aware, adaptive interfaces dates back to the original context-awareness paper by Schilit et al. [Schilit 1994]. Tailoring user interfaces to surrounding context provides clear benefits (e.g. reduce information clutter, adaptation to current screen resolution, etc.). Nevertheless, the introduction of context-aware features must be associated to user field studies, as acknowledged by by Hong et al. [Hong 2009].

2.4 Motivations and Challenges

The rising popularity of sensor-rich mobile devices determines a growing mobile access to the Web. Since it is an emerging part of the Web, the Web of Data will see an increasing influence of such mobile trend, thus paving the way for novel mobile scenarios and applications, eventually leading to a Ubiquitous Web of Data. As ubiquitous connectivity spreads, mobile users interact with the classic Web with heterogeneous, always-connected devices and under novel circumstances (e.g. in crowded areas, on public transportation systems, etc). Ubiquitous Web of Data applications will offer novel ways of consuming and contributing to Linked Data, thanks to the adoption of compelling interaction modalities driven by deeper awareness of the surrounding physical environment (e.g. enhanced interfaces, vocal access to the web, augmented reality, etc). Such awareness will be the result of detecting the conditions in which Linked Data consumption takes place, thanks to data captured by embedded sensors. Such scenario motivates a series of challenges that have been discussed in this thesis, tasks that must be faced to move towards the vision of a Ubiquitous Web of Data.

First, bringing context-aware features to Web of Data consumption requires domain-independent, adaptive representation of Linked Data to the surrounding context. The first problem we face is the lack of shared and declarative tools for RDF user interfaces. This affects all Web of Data applications, mobile included. RDF representation is deliberately dismissed as an application-related feature and user interface generation for Linked Data is delegated to application-specific code. Yet a uniform, declarative RDF presentation layer seems legit given that (i) presentation is needed by all applications (we have plenty of libraries for accessing and parsing RDF but we rewrite presentation-level code in each application) and (ii) when browsing the Web of Data we do not know beforehand the vocabulary adopted by a resource (e.g. when applying the follow-your-nose principle with a Linked Data browser).

Consuming Linked Data on the go benefits from the adoption of access control mechanisms that take into account the conditions in which data consumption is

performed, as uncontrolled access in given situations may be undesired by data providers. Moreover, the open nature of current Web of Data information and the consumption of web resources from mobile devices may give providers the impression that their content is not safe, thus preventing further publication of datasets, at the expense of the growth of the Web of Data itself. Thus, another important challenge is introducing access control, adding mobile context as part of the access control evaluation. Such challenge needs a number of steps: first, the definition of a fine-grained access control model for graph stores, then the modelling of context-aware, mobile consumption of such information, and finally the integration of mobile context in the access control model.

Both challenges, adaptation of Linked Data presentation to context, and access control for RDF stores accessed by mobile devices, need a preliminary step: the definition of a context model that fits into a Web of Data scenario. Many existing ontology-based context models are expressive and flexible, and among them, several rely on RDFS/OWL. Nevertheless, many do not support the open world assumption (thus, they are not extensible), nor they reuse terms from existing ontologies. Moreover, none of the existing context ontologies features a real lightweight approach, as these models are made of a large number of terms and they often rely on advanced OWL statements.

Chapter 3

A Declarative Model for Mobile Context

Contents

3.1 Introduction **21**
3.2 Ontology-Based Context Models **22**
3.3 The PRISSMA vocabulary **25**
3.3.1 Design Rationale 25
3.3.2 Vocabulary Description 26
3.3.3 Examples 28
3.4 Conclusions **29**

3.1 Introduction

The first challenge faced by any context-aware application is the adoption of a proper context model. Context-aware adaptation of Linked Data and context-aware access control for RDF stores are no exception, as both applications need a schema to represent, store, and exchange context information. Hence, this chapter answers the first research question of the thesis: *how to declaratively describe client context in a Linked Data scenario?*

The question opens a number of challenges: Modelling client context is a complex task, and many issues must be faced. First, a proper context definition must be chosen: "Context" is a faceted and widely-used term, that must be formally defined; the work adopts Dey's context definition [Dey 2001], as explained in Chapter 1. Then, the most suitable schema paradigm must be adopted, according to the requirements determined by the scenario (e.g. attribute-relation, object-oriented schema, ontology-based, etc.). A Semantic Web setting suggests the adoption of an ontology-based context model. Hence ontology engineering requirements must be met. Moreover, a Linked Data scenario introduces a series of additional constraints that must be matched by the context model (e.g. the adoption of a lightweight vocabulary instead of a vast, monolithic context ontology).

Several context ontologies have been proposed, both in the Semantic Web community and in Context Awareness research (Section 3.2). Nevertheless, none of them meet Linked Data best practices.

The PRISSMA vocabulary has been created to fill the gap between context ontologies and the Web of Data: it consists in a domain-independent, lightweight ontology that models core context concepts and re-uses classes and properties proposed in well-known, third-party vocabularies, thus providing a small number of new classes and properties. Despite the size, the vocabulary is designed to cover a variety of context dimensions. The vocabulary supports future extensions, thus complying with the open world assumption.

Section 3.2 presents a state of the art of ontology-based context models, with a particular focus on the requirements posed to the context model by a Web of Data scenario. Section 3.3 discusses the design principles of the PRISSMA vocabulary and describes the model, along with examples of context descriptions.

3.2 Ontology-Based Context Models

Many context ontologies have been proposed in the context awareness community. This section presents a number of ontology-based context models, and compares them along a set of requirements coming from context awareness, ontology engineering, and the Web of Data scenario. Context awareness and ontology engineering requirements are thoroughly discussed in context models surveys: Bolchini et al. [Bolchini 2007] compare context models created with different paradigms (e.g. key-value models, markup schemes, object-oriented, ontology-based). The survey provides a set of general requirements for context models, and proposes an assessment framework for comparing existing models and creating new ones. Among the requirements, the authors underline the set of supported context dimensions (e.g. space, time, user profile, etc.), representation features (e.g. formality, flexibility, granularity), and context management aspects (context reasoning, context ambiguity management, context quality monitoring, etc.). Nevertheless, their approach is rather general and it is not specific to ontology-based systems. Korpipää and Mäntyjärvi [Korpipää 2003] describe a list of requirements of ontology-based context models: among all, simplicity for application developers, expandability, and expressiveness. Krummenacher and Strang provides assessment criteria specific for ontology-based context models [Krummenacher 2007], borrowing from context modelling requirements (quality, incompleteness, traceability etc.), and ontology engineering criteria (reusability, flexibility, granularity, etc.). The aforementioned surveys help defining a subset of context awareness and ontology engineering requirements that has been used in this thesis to compare off-the-shelf models:

Domain independence. A number of context ontologies have been created to model a given domain-specific scenario. Others adopt a domain-independent approach.

Coverage. The ontology must guarantee a proper level of completeness for what concerns the desired contextual dimensions. In particular, the model must

support multiple context dimensions such as device features, user preferences, location and time.

Formality. Some ontology-based context models rely on formal definitions, while others adopt a more intuitive approach.

Variable Context Granularity. Certain ontologies model context dimensions at different level of granularity. For example, location might be expressed in terms of latitude and longitude, or with a label assigned to a place (e.g. office, beach, cinema, etc.).

User Friendliness Evaluation. Context-aware application developers must spend a reasonable amount of effort dealing with the context model, thus the ontology must be sufficiently easy to adopt and well documented. The presence of a user evaluation campaign to assess such feature is assessed by certain context models.

Core ontology approach. The vocabulary must adopt a modular design, thus focusing on modelling core classes and properties that will be extended by third-party domain specialists.

Beyond context awareness and ontology engineering requirements, a series of specifications must be taken into account when working in a Web of Data scenario. In other words, off-the-shelf context models must be analysed under a Web of Data point of view. Thus, a number of additional requirements must be taken into account:

Open World Assumption. The Web of Data is an open environment, and describing context in this scenario must consider third-party extensions unknown beforehand. Extensibility must be obtained with a low effort, thus add-ons must not impact on the already existing model.

Lightweight Ontology. According to Linked Data best practices [Heath 2011], the ontologies used in the Web of Data are defined with the RDFS language. Simple OWL extensions are accepted (e.g. `owl:equivalentClass`, `owl:InverseFunctionalProperty`), but the goal is to keep ontologies small and simple. Context ontologies are no exceptions: this simplifies context modelling.

Reuse of Existing Terms. Linked Data best practices favour the reuse and the combination of classes and properties of existing vocabularies. This is done to prevent the proliferation of terms and reduce the range of choices when modelling data.

Availability on the Web. Web of Data vocabularies are published on the Web, and accessible according to Web of Data best practises[21]. Moreover, they are

[21] For a complete description of how to publish an RDF vocabulary on the Web see http://www.w3.org/TR/swbp-vocab-pub/.

associated to an HTML page, the "namespace document", whose task is to provide a textual description of the vocabulary rationale, along with classes and properties explanation and examples.

A number of ontology-based context models relying on Dey's definition have been proposed in the latter years[22]: The SOUPA ontology [Chen 2004] is an OWL model for representing very general context information. SOUPA is extensible, thus supporting the open world assumption. SOUPA reuses third-party ontologies terms (such as FOAF), but many external ontologies do not comply with Linked Data best practices (e.g. ontology publicly available on the Web). Although the ontology is designed as separated modules, it cannot be considered a lightweight vocabulary due to its size. CoOL [Strang 2003] is a modular OWL-based context ontology that relies on OWL features typically not present in lightweight ontologies. Moreover, it is grounded in the F-Logic logic language. It is a general-purpose context model, not explicitly designed for client-based context attributes. It does not reuse existing vocabularies. CONON [Wang 2004] is a modular OWL context ontology. The expressivity of CONON is guaranteed by a modular approach, where each module consists in domain-specific set of context classes and properties. CONON is designed for extensibility. The CONON ontology is not published on the Web and does not reuse existing vocabularies. The CoDaMoS [Preuveneers 2004] OWL ontology is designed to be extensible and to model client-side context attributes, but it does not reuse or integrate other vocabularies. It does not consists in a lightweight vocabulary. The ontology is available on the Web, but no namespace vocabulary is present. Korpipää et al. [Korpipää 2003] present a context model designed for mobile, context-aware applications. Their approach is rather general, they do not reuse existing terms and their ontology is not designed to support extensions. Hervás and Bravo propose a modular context model [Hervás 2011] composed by independent ontologies and support future extensions. Nevertheless, they do not reuse already existing linked data ontologies, thus not adopting a lightweight approach. Initiatives such as the W3C Model-Based User Interface Incubator Group[1] deal with classic Mobile Web content adaptation. The declared goal is to 'evaluate research on model-based user interface design' for the authoring of Web applications. The authors propose the Delivery Context Ontology (DCO)[23], a modular, fine-grained vocabulary to model mobile platforms. The ontology is modular, but although published on the Web, it cannot be considered a lightweight vocabulary because of the lack of interlinking with other vocabularies - the ontology reuses concepts modelled by the CC/PP ontology[24], although not a vocabulary level (i.e. it does not reuses third-party RDF terms).

Table 3.1 shows a comparison of ontology-based context models, according to the aforementioned context awareness, ontology engineering, and Web of Data re-

[22] What follows is a non-exhaustive list of literature works, since this task is out of the scope of this thesis. See Bolchini et al. [Bolchini 2007] and Krummenacher and Strang [Krummenacher 2007] for more complete surveys.

[23] http://www.w3.org/TR/2009/WD-dcontology-20090616/

[24] http://www.w3.org/2006/09/20-ccpp-schema

quirements. Although created with similar goals, none of the works match all our requirements. All the analysed works adopt a domain independent approach, and their coverage is sufficiently complete to consider a satisfactory number of context dimensions. Interestingly, none of the reviewed works is grounded on a formally defined context models. Another missing feature is a user-friendliness evaluation of such ontologies. Nearly all of the reviewed works are far from the Web of Data best practices: although the majority of them supports third-party extensions, none adopts a lightweight approach (although DCO and SOUPA are heavily modular). SOUPA is the only ontology to reuse existing, third-party terms, but such feature is rather limited. Finally, only DCO is published on the Web according to the Linked Data best practices. Such shortcomings discourage the adoption and reuse of these ontologies in the Linked Data community.

	SOUPA[Chen 2004]	CoOL[Strang 2003]	CONON[Wang 2004]	CoDaMoS[Preuveneers 2004]	Korpipää et al. [Korpipää 2003]	Hervás and Bravo[Hervás 2011]	DCO[23]	**PRISSMA Vocabulary**
Domain independence	●	●	●	●	●	●	●	●
Coverage	●	○	●	●	○	●	○	●
Formality								
Variable Context Granularity		○	●					○
User Friendliness Evaluation								
Core ontology approach	●	●	●			●	●	●
Open World Assumption	●	○	●	●	○		○	●
Lightweight Ontology								●
Reuse of Existing Terms	○							●
Availability on the Web				○			●	●

Table 3.1: A comparison of ontology-based context models. Full support is identified by ●, partial support by ○, no support by the empty cell.

3.3 The PRISSMA vocabulary

3.3.1 Design Rationale

To overcome the limitations of the existing, off-the-shelf ontology-based context models, this Section introduces the PRISSMA vocabulary (Figure 3.1). PRISSMA is a domain-independent vocabulary designed to model client-generated context data. It is aimed at covering a number of heterogeneous context dimensions. The ontology heavily reuses well-known Web of Data vocabularies and W3C recommendations. It follows a lightweight approach, since it relies exclusively on RDFS and OWL basic statements and it contains a low number of terms. Moreover, although specifically

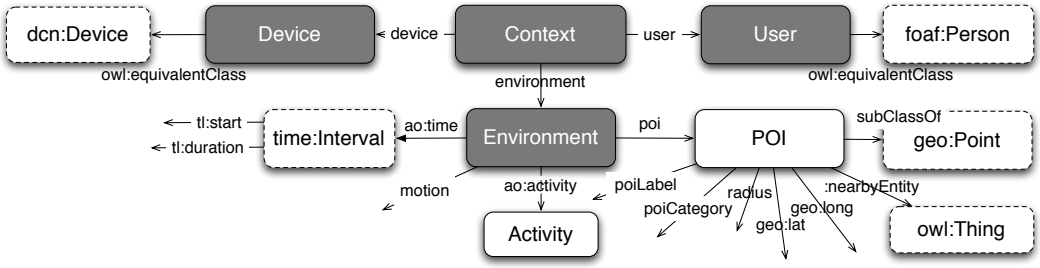


Figure 3.1: The PRISSMA vocabulary at a glance. Grey boxes represent core classes, i.e. the *context dimensions*. Dotted classes picture reused terms from third-party ontologies.

designed to model client-generated context data, the vocabulary does not provide a comprehensive, exhaustive model: the approach is to delegate refinements and extensions to domain specialists, thus embracing the open world assumption. The PRISSMA vocabulary has been published on the Web[25], according to Linked Data best practices. The ontology triples are dereferencable, and the associated HTML document namespace is returned if HTML content-type is required.

3.3.2 Vocabulary Description

The PRISSMA vocabulary is based on the widely-accepted formalization of context proposed by Dey [Dey 2001]. More precisely, it extends the W3C Model-Based User Interface Incubator Group proposal[1] where mobile context is described as an encompassing term, an information space defined as the sum of three different dimensions: *User* model and preferences, *Device* features, and the *Environment* in which the action is performed. The complete vocabulary is presented in Appendix A. A graph-based representation is provided Figure 3.1. The core of the PRISSMA vocabulary consists in classes and properties that models the aforementioned context dimensions:

`Context.` The Context class represents the mobile context, according to Dey's context definition [Dey 2001].

`User.` Represents the target client user associated to a `Context` and consists in a `foaf:Person` equivalent class. To provide more flexibility, the class can be used to model both user stereotypes and individuals.The property `user` associates a User to a Context.

`Device.` Represents the mobile device on which Web of Data resource consumption takes place, enabling device-specific data representation. The class is equivalent to the W3C Delivery Context Ontology[23] `dcn:Device` that provides an extensible and fine-grained model for mobile device features. The property `device` associates a Device to a Context.

[25] `http://ns.inria.fr/prissma/v2`

`Environment`. Models the physical context in which the Web of Data resource consumption takes place, enabling customized resource representation according to specific situations. Different dimensions are involved in modelling the surrounding environment. The property `environment` associates an Environment to a Context.

Aside from the core classes mentioned above, the vocabulary contains other classes and properties used to enrich the `Environment` dimension:

`POI`. Location is modelled with the notion of Point of Interest (POI). The class consists in a simplified, RDFized version of W3C Point of Interest Core specifications[26]. POIs are defined as entities that *describe information about locations such as name, category or unique identifier*. Each POI consists in a `geo:Point` subclass that can be associated to a latitude, longitude and a physical radius[27].

`Activity`. The class consists in a placemark to connect third-party solutions focused on inferring high-level representations of user actions (e.g. 'driving', 'working', 'shopping', etc.).

`radius`. The property specifies the geographic extension of a POI. Value is expressed in metres.

`poi`. The property associates a POI to a prissma:Environment.

`poiLabel`. The property assigns a custom name to a point of interest. The name can be both a URI (e.g. `http://dbpedia.org/resource/Harrods`) or a literal.

`poiCategory`. The property associates a custom category to a POI (e.g. monument, restaurant, etc.). The vocabulary does not constrain the range of the property in any kind. Hence, both URIs (e.g. `http://dbpedia.org/resource/Shopping_mall`) and literals are valid. For string literals, a list of place types is presented in RFC 4589[28].

`motion`. The property associates any given high-level representation of motion to a prissma:Environment.

The PRISSMA vocabulary reuses terms from third-party ontologies, a common practice in the Web of Data. The following list presents a high-level description of such vocabularies.

Friend of a Friend Ontology. FOAF is a collection of terms that describe people, groups, documents. It is composed by a core set of terms, that describe the characteristics of people and group of individuals, and a "social web" set of

[26] `http://www.w3.org/TR/poi-core`

[27] More refined strategies can be adopted to model a geographical location, but this is out of scope of this work.

[28] `https://www.ietf.org/rfc/rfc4589.txt`

classes and properties, useful to model online accounts[29]. The vocabulary prefix is `foaf`.

The Delivery Context Ontology. The Delivery Context Ontology (DCO) provides a formal model of the characteristics of the environment in which devices interact with the Web or other services. The Delivery Context includes the characteristics of the Device, the software used to access the service and the Network providing the connection among others[30]. Since the ontology is modular, it includes a series of vocabulary prefixes. PRISSMA context declarations reuse the device-related part of the ontology, thus the prefixes are `dcn` - the core ontology, `hard` for device hardware features, `soft` for device software components description, and `common` for common utility terms.

The Association Ontology. The vocabulary specification provides *basic concepts and properties for describing specific association statements to something*, e.g. time, current activity[31]. The vocabulary prefix is `ao`.

WGS84 Geo Positioning Vocabulary. A vocabulary for representing latitude, longitude and altitude information, according to the WGS84 specifications[32]. The vocabulary prefix is `geo`.

The Time Ontology. This OWL ontology describes facts and relations among instants and intervals[33]. The vocabulary prefix is `time`.

The Timeline Ontology. The ontology models the notion of timeline, and can be used to annotate sections of any temporal object[34]. The vocabulary prefix is `tl`.

The third-party terms that have been reused by the PRISSMA vocabulary are listed in Table 3.2.

3.3.3 Examples

Figure 3.2 shows two sample context declarations expressed with the PRISSMA vocabulary. The triples in Figure 3.2a model a user by his interests (lines 13-15). The modelled user is interested in "computer programming" and Star Trek. The geographic location (lines 21-23) is determined by a (latitude, longitude) couple and by a radius that determines the extension of the area. The context contains a six-hour time interval that begins at 10:00 on a specific day (lines 25-26). Figure 3.2b represents a user that knows Jack (identified by his FOAF profile, `http://jack.example.org#me`, line 18). The context also states that the user uses

[29] `http://xmlns.com/foaf/0.1/`
[30] `http://www.w3.org/TR/dcontology/`
[31] `http://purl.org/ontology/ao/core#`
[32] `http://www.w3.org/2003/01/geo/wgs84_pos#`
[33] `http://www.w3.org/2006/time#`
[34] `http://motools.sourceforge.net/timeline`

`time:Interval`	The class represents a temporal interval.
`ao:time`	The property is used to add a time to an Environment.
`tl:start`	This property refers to the beginning of a time interval. Property value must be a `xsd:dateTime` literal.
`tl:duration`	The property adds the duration of a time interval. Property value must be a `xsd:duration` literal.
`geo:lat`	The property to add the WGS84 latitude of a SpatialThing (decimal degrees).
`geo:lon`	The property to add the WGS84 longitude of a SpatialThing (decimal degrees).
`ao:activity`	The property links an activity to an Environment (e.g. walking, working, driving).
`foaf:based_near`	The environmental proximity of a generic real-world entity can trigger different resource representations. The property is therefore used to associate nearby objects to the Environment class.

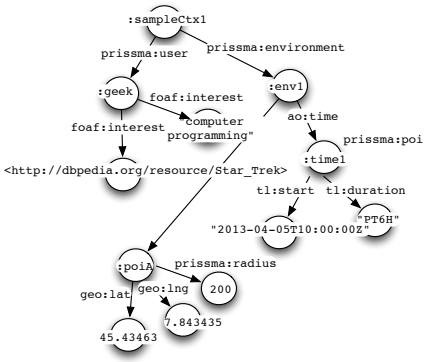
Table 3.2: Third-party classes and properties reused by the PRISSMA vocabulary.

a mobile device with a resolution of 1024x600 (lines 24-26) and an Android 4.2.2 operating system (line 28). Both examples shows the reuse of third-party vocabularies, e.g. `foaf:interest`, `ao:time`.

3.4 Conclusions

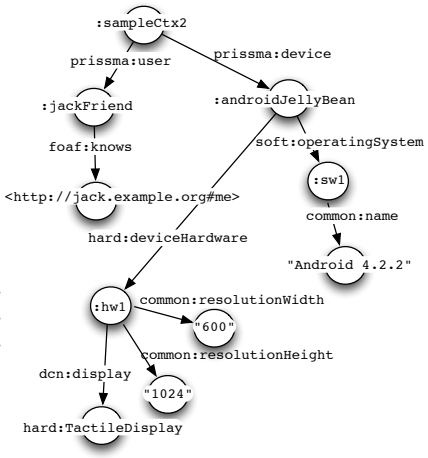
Representing context data needs a proper schema, in other words a context model is needed. Over the years, context awareness research proposed different context model paradigms. Ontology-based context models are a popular way of modelling context data, thanks to their expressivity and flexibility. Although many RDFS/OWL context ontologies have been proposed, a Web of Data scenario requires features that are not matched by off-the-shelf works. Some existing models do not support the open world assumption (thus, they are not extensible), nor they reuse terms from existing ontologies. Moreover, none of the existing context ontologies features a real *lightweight* approach, as these models are made of a large number of terms and they often rely on advanced OWL statements. Furthermore, existing context ontologies do not feature a modular design, thus preventing their adoption since they would introduce terms that model concepts beyond client-generated context data (e.g. classes and properties to model context-aware Web services, out of the scope of this work). Finally, no existing context model complies with the Linked Data publishing best practices. Such issues determine the need for a Linked Data-compliant context vocabulary: the PRISSMA lightweight ontology presented in this chapter is the first contribution of this thesis and lays the foundation of both context-aware adaptation of Linked Data (Chapter 4) and context-aware access control for RDF stores (Chapter 5). The PRISSMA vocabulary models context as an information space defined as the sum of the mobile User model, the Device features, and the Environment. The vocabulary reuses terms from third party Linked Data ontolo-

```
@prefix : <http://example.org#> .
@prefix prissma: <http://ns.inria.fr/prissma/v2#> .
@prefix fresnel: <http://www.w3.org/2004/09/fresnel#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .

:sampleCtx1 a prissma:Context ;
   prissma:user :geek ;
   prissma:environment :env1 .

:geek a prissma:User ;
   foaf:interest "computer programming" ,
                 <http://dbpedia.org/resource/Star_Trek> .

:env1 a prissma:Environment ;
   prissma:poi :poi1 ;
      ao:time :time1 .

:poi1 geo:lat "45.43463" ;
     geo:long "7.843435" ;
     prissma:radius "200" .

:time1 tl:start "2013-04-05T10:00:00Z"^^xsd:dateTime ;
      tl:duration "PT6H"^^xsd:duration .
```



(a)

```
@prefix : <http://example.org#> .
@prefix prissma: <http://ns.inria.fr/prissma/v2#> .
@prefix fresnel: <http://www.w3.org/2004/09/fresnel#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcn:
  <http://www.w3.org/2007/uwa/context/deliverycontext.owl#>.
@prefix hard: <http://www.w3.org/2007/uwa/context/hardware.owl#>.
@prefix common: <http://www.w3.org/2007/uwa/context/common.owl#>.
@prefix soft: <http://www.w3.org/2007/uwa/context/software.owl#>.


:sampleCtx2 a prissma:Context ;
   prissma:user :jackFriend ;
   prissma:device :androidJellyBean ;

:jackFriend a prissma:User;
       foaf:knows <http://jack.example.org#me> .

:androidJellyBean a prissma:Device;
   hard:deviceHardware :hw1 ;
   soft:operatingSystem :sw1 .

:hw1 dcn:display hard:TactileDisplay;
    common:resolutionHeight "1024" ;
    common:resolutionWidth "600" .

:sw1 common:name "Android 4.2.2" .
```



(b)

Figure 3.2: Sample PRISSMA vocabulary context representations: (a) a "geek" user in a location with a given latitude, longitude and radius, in a specific time interval. (b) A friend of Jack on an Android device.

gies and is extensible with domain-specific classes and properties. Yet, PRISSMA has a number of limitations: the concepts represented by PRISSMA classes and properties are not supported by a formal model: concepts are empirically derived from the adopted definition of context. The model does not support disjunctive conditions, thus multiple declarations are needed to express disjunctions. Moreover, the user friendliness of the PRISSMA vocabulary has not been tested yet. A future evaluation campaign would assess the ease of adoption of such vocabulary by developers.

Chapter 4

Context-Aware Presentation of Linked Data on Mobile

Contents

4.1 Introduction . . . 33
4.2 Definitions . . . 35
4.3 Related Work . . . 36
4.3.1 Context-Aware Web Content Adaptation . . . 36
4.3.2 Presentation-level Frameworks for Linked Data . . . 39
4.3.3 Error-tolerant matching for RDF Graphs . . . 44
4.4 PRISSMA Presentation Framework: Overview . . . 46
4.4.1 Combining Fresnel and PRISSMA Vocabularies . . . 46
4.4.2 The Problem of Prism Selection . . . 50
4.4.3 Resource Rendering . . . 51
4.5 Prism Selection Algorithm . . . 52
4.5.1 Definitions . . . 52
4.5.2 Decomposition . . . 54
4.5.3 Search Algorithm . . . 59
4.5.4 Computational Complexity . . . 62
4.5.5 Cost of Edit Operations . . . 64
4.6 Evaluation . . . 70
4.7 PRISSMA Browser . . . 72
4.8 Conclusions . . . 75

4.1 Introduction

Content adaptation on the Web consists in the process of selecting, generating, or modifying content units in response to a requested URI[35]. This feature is essential for the mobile Web and is driven by the multifaceted notion of *client context* [Dey 2001]. As proposed in [Heath 2008], being aware of the surrounding physical environment also improves the effectiveness of Linked Data consumption. Semantic Web mobile applications might not have built-in assumptions about the schemas of the

[35] http://www.w3.org/TR/di-gloss/

data they consume, as the data model could be unknown a-priori, and provided by heterogeneous sources: users might consume any type of data, as long as it is relevant to their context [m. c. schraefel 2010]. These features require content adaptation, such as context-based filtering, ordering, grouping and formatting of triples. Content adaptation reduces the fan-out of linked data connections, and provides coherent information by using context as dynamic filter. In other words, it creates "optimized" content units ready for user consumption. The most relevant challenge faced by content adaptation is the *selection* of the presentation knowledge that must be activated for a given sensed context. This is due to the imprecise and incomplete nature of context data, that complicates the matching procedure between declared and sensed contexts, and requires an error-tolerant approach.

In this chapter, we address the question of *how to enable context-aware adaptation for linked data consumption*. We split up the problem in two sub-questions: i) *how to use context descriptions at RDF presentation-level* and ii) *how to deal with context imprecision to select the proper context description at runtime*. In the following sections we present PRISSMA[36], a context-aware presentation framework for Linked Data. PRISSMA[37] consists in a Fresnel-based [Pietriga 2006] RDF rendering engine that selects the most appropriate presentation of RDF triples according to mobile context. In other words, PRISSMA extends the Fresnel rendering process with context awareness by adding a preliminary step, the *presentation data selection* (i.e. it enables Fresnel engines to choose the best representation according to a given context). We answer our two-fold research question with two main contributions: i) a lightweight ontology for describing context conditions in a declarative style, compatible with Fresnel (by adopting the PRISSMA vocabulary described in Chapter 3), and ii) an error-tolerant algorithm to determine whether the sensed context is compatible with context declarations, hence the most appropriate data pre-processing and visualization can be activated. The algorithm must satisfy a series of requirements: first, the intrinsic imprecision of contextual data determines the need for an error-tolerant strategy that takes into account possible discrepancies between context descriptions and the actual context. Second, this error-tolerant mechanism must support heterogeneous context dimensions (e.g. location, time, strings). Third, since the procedure must run on the client-side - to avoid disclosing sensitive context information - we must design a mobile-friendly algorithm, with acceptable time and space complexity. Finally, the adopted strategy must support runtime updates of RDF graphs, as context descriptions might be fetched from remote repositories and added to the selection process at runtime, and the sensed context may change at any time. In Section 4.3 we present state-of-the-art presentation frameworks for the Semantic Web, along with an overview of error-tolerant matching techniques. In Section 4.4 we describe design principles of PRISSMA and we introduce the PRISSMA model for presentation-level context data. Section 4.5 explains the error-tolerant selection algorithm. Selection algorithm experimental

[36] Presentation of Resources for Interoperable Semantic and Shareable Mobile Adaptability
[37] From latin *prisma*. In optics, a Fresnel lens can be considered as a series of prisms.

evaluation results are described in Section 4.6. A proof-of-concept mobile application adopting PRISSMA is presented in Section 4.7. We discuss the limitations of our approach and future work in Section 4.8.

4.2 Definitions

A number of definitions must be provided to introduce concepts described in this chapter.

According to the ISO/OSI model the presentation layer is formally defined as follows [Zimmermann 1980]:

Definition 2 (Presentation Layer.) *The presentation layer is in charge of the formatting and the delivery of the data for the application layer.*

Typically, it can contain data compression, conversion, and encryption operations, along with character encoding conversions. This work adopts the conceptual definition of presentation layer, although operating at a different level of the ISO/OSI stack. The presentation layer includes the task of content adaptation, that has formally defined by W3C Device Indepence Activity[38]:

Definition 3 (Content Adaptation.) *The process of selection, generation or modification that produces one or more perceivable units in response to a requested uniform resource identifier in a given delivery context.*

The definitions of presentation layer and content adaptation lead to the following definition:

Definition 4 (Presentation Framework.) *A presentation framework is a component in charge of executing data adaptation tasks on behalf of the application layer.*

The goal of this chapter is enable context-aware content adaptation for Linked Data. Context information must be expressed according to the most appropriate context definition (this work adopts Dey's definition [Dey 2001]). Furthermore, such context data must be serialized in meaningful elements. In this work such elements are defined as follows:

Definition 5 (Context Declaration.) *A context declaration is an RDF graph that contains triples about a* `prissma:Context` *instance.*

The term multimodality has a wide range of definitions. This work adopts the definition proposed by Nigay and Coutaz [Nigay 1993]:

Definition 6 (Multimodality.) *Multimodality is the capacity of the system to communicate with a user along different types of communication channels [. . .] automatically.*

[38] http://www.w3.org/TR/2003/WD-di-gloss-20030825/#def-adaptation

4.3 Related Work

In this section, three groups of works are surveyed. First, a state-of-the-art of existing context-aware systems for adapting web content is provided. Second, a survey of presentation-level frameworks for RDF is presented. The survey includes the overview of Fresnel, the framework extended by PRISSMA. Third, the issue of error-tolerant RDF matching is addressed.

4.3.1 Context-Aware Web Content Adaptation

A number of recent works deal with adaptation of Web resources on mobile devices. This section underlines the main features along with the motivations for Web content adaptation on mobile. The review focuses exclusively on works that adapt Web content, and on mobile devices only. Table 4.1 compares the aforementioned works, that are reviewed and compared according to the following criteria:

Linked Data Support. Indicates if the work has been designed to adapt RDF resources, and to address a Web-of-Data-specific scenario. Partial support for Linked Data means that the framework uses Semantic Web technologies in the adaptation process, but to a very limited extent.

Context Awareness. Specifies if the system supports full-fledged context awareness, as defined by Schilit et al. [Schilit 1994]. Partial support means that context features are presented, but limited to a small set of context dimensions.

Standard Languages. The adoption of standard languages (e.g. XML, CSS, XForms) is taken into consideration. If the adaptation process uses ad-hoc languages besides standard solutions, it is considered to be partially standard compliant.

Runtime Adaptation. Some works feature content adaptation that reconfigures itself on-the-fly, triggered by context updates. Some works provide such features only under certain conditions, thus they are labelled in the survey with partial support.

Multimodality. An adaptive framework is considered to be multimodal if it has the ability of providing heterogeneous output formats (e.g. text, audio, etc). Some works provide partial support for such feature. Such works can be typically extended to be multimodal with low effort, or their model supports multimodality but this feature has not been implemented by the system.

Client-side. Indicates if a system performs content adaptation on the client, thus not relying on a server-side infrastructure.

Evaluation. The presence of an evaluation campaign is assessed. Evaluation is considered to be complete only if authors provide a quantitative user interac-

tion campaign. A partial evaluation contains only performance results, such as response time analysis.

Note that among the systems that provide context adaptation of Web resources, none of them address a Web-of-Data-specific scenario, or targets RDF resources.

An early attempt has been proposed by Lemlouma and Layaida, who deal with content adaptation for mobile devices, and present NAC [Lemlouma 2004]. NAC supports adaptation of HTML documents to client context (environment and device capabilities). They rely on a context ontology called UPS, and they provide a phase of context extraction used by the content adaptation step. The adaptation phase transforms the content of a Web page into a different layout according to the context, and is performed on server-side. NAC uses profile repositories for fetching device features.

Chen et al. (2005) present an adaptation technique for Web pages [Chen 2005] that scans the HTML structure and splits content in smaller parts that fit small screens. Context is not taken into account.

Zhang [Zhang 2007] discusses selective, context-based Web content delivery and Web content adaptation on early models of Web-enabled mobile devices. Their system, *Mobile Web*, uses a fisheye view and adapts HTML content to device type and environmental context.

Nathanail et al. present a multimodal and adaptive system for Web interfaces called Chamaleon [Nathanail 2007]. It adapts to client devices, user profiles and surrounding environments. Chamaleon supports heterogeneous platforms, and adapt content in a seamless fashion (transparent to users). The approach is server side: adaptation is delegated to the Web server, that must know context parameters of the client. Chamaleon relies on previously defined presentation rules and it uses XSLT. The paper does not present a multimodal framework (thus providing HTML adaptation only), but the system hints to voiceXML. Policies are expressed in an ad-hoc, rule-based, Adaptation Policy Language (APL), based on XML. Policies define how the system must build CSS for formatting adapted pages, according to sensed context parameters. Automatic adaptation sends context changes to the server, that re-creates an updated version of the page, thus delivering continuous, real-time adaptation. Conflicts between rules are not handled and should be resolved by rule authors. The abstract user interface language uses XForms.

Butter et al. [Butter 2007] provide a XUL-based context-aware adaptation framework for mobile devices (an attempt similar to Mitrovic [Mitrovic 2002]). The framework separates adaptation from application logic and is an extension to XUL to adapt application UIs to different devices and user contexts. The structure of the interface is specified in XUL, but the appearance is specified with CSS. They extend CSS to use a given XUL definition and CSS according to user context. For CSS, the authors added the proprietary tag *contextstyle*. Dey's definition of context is adopted. Selection of relevant stylesheets is done dynamically, so that the UI changes on-the-fly. The work does not deal with context fetch, and contains partial evaluation (the authors measure the user interface creation time).

In a comprehensive survey, Laakko discusses Web content adaptation for context-aware devices [Laakko 2008]. The paper discusses context ontologies relying on Dey's context definition, and user profiling issues. He discusses also the W3C and OMA proposals for delivery context, in systems where adaptation is performed on the server-side. Laakko discusses client side, server-side and proxy-side adaptations approaches. He deals with content selection, although the focus is on content transformation. In a previous paper the same author provides also the description of an adapting proxy for Web content for mobile devices [Laakko 2005]. Adaptation is performed according to user agent profiles (thus being partially context-aware), and supports HTML content and media (e.g. images). First, the original Web page is analysed, then decomposed according to the interpretation of the original content (in other words, original Web pages are scaled down to low-end devices).

CAMB is a mobile context-aware browser that adapts HTML pages according to a predetermined set of environmental situations [Gasimov 2010]. The authors first discuss the notion of mobile context and classify interesting scenarios. They then proceed with context fetch and estimation and they finally adapt Web pages using context-aware CSS definitions (they use the CSS "media" property, to support any kind of context, so that more than one context-based CSS can be associated by Web developers to a HTML page). They adopt a context definition that includes device information, user profile and environment data. The work adopts a server-side solution to deal with context identification (Context estimation server) and it is able to detect situations such as walking, or users under stress conditions. The browser continuously sends raw data to the context server, that returns the refined context detection, so that the browser can apply the right CSS.

Woensel et al. [Woensel 2011] describe an adaptation framework for adding context-aware features to existing Web pages accessed from mobiles, and provide a prototype, COIN. They extract semantic annotations (e.g. RDFa) from HTML pages, match these metadata with the retrieved user context to detect relevant content in the Web page, and adapt the Web page conversely. They rely on a notion of context including device features, user preferences and environment data. They discuss the effect of adaptation on users. Having extracted semantic information from a Web page, a context matching algorithm is executed and context-adaptive features are added to the Web page. Although the matching algorithm is not described in detail, the author say that it matches user interests with RDF resources found in page metadata. They also hint to instance matching techniques. The technique basically boils down to a SPARQL query with a FILTER clause executed on a local RDF repository containing context information (SCOUT local SPARQL endpoint). Eventually, content adaptation is applied, i.e. they apply a list of HTML modifications stemmed from adaptive hypermedia (i.e. showing, hiding, altering content, emphasizing, dimming, layout changes, sorting, ordering). The approach followed by the COIN prototype is completely client-side. COIN relies on a context-fetcher framework called SCOUT, that organizes data in RDF and allows SPARQL queries over it. The authors do not provide a way for designers to specify the type of visual adaptation on the page.

The MIMOSA framework [Malandrino 2010] includes a content adaptation component for HTML pages and Web APIs results. MIMOSA supports context-awareness, and relies on adaptation policies. Conflict between policies are handled, and an adaptation policy editor is provided. Experimental evaluation is described, both on system performance and user experience). Adaptation is performed on the server side: raw context attributes are sent from clients to the server, where context aggregation and reasoning is performed. Context data is represented with the CC/PP specifications[39] and contains user preferences, device features (using UAProf) and the surrounding environment. The MIMOSA adaptation module, developed as an Apache extension, applies the adaptation rules specified by the client context (e.g. extracting relevant parts of Web pages, image downscaling). From an user-interaction point of view, the authors are aware that adaptation is a tricky issue, as users do not want the system to behave in unexpected ways. Adaptation rules are defined both by the service provider and by users (this is why conflict resolution is provided).

Paternò et al. [Paternò 2010] present a model-based framework for adapting Web pages to mobiles. Their work does not consider context and the approach is based on a proxy server. The framework relies on MARIA, a model-based language to specify user interfaces. Content adaptation is performed to meet device features only (e.g. screen size), and the process can be configured by users with an ad-hoc editor.

Adipat et al. [Adipat 2011] present a system for adapting Web content on mobile devices, although not context-dependent. They apply tree-view, hierarchical text summarization, and coloured keyword highlighting. Evaluation results show the benefit of their approach in terms of user perception.

Several W3C initiatives have been created around CSS device adaptation[40]. Among all, the most popular are CSS Media Queries[41]. Media queries offer layout and behaviour adaptation of an HTML page, according to device features, in particular to the device screen resolution.

4.3.2 Presentation-level Frameworks for Linked Data.

Several presentation-level frameworks for Linked Data exist. The following review, summoned in Table 4.2, compares the works along a number of features:

Declarative Approach. Indicates if the work relies on declarations modelled by an ontology, and it does not depend on an hardwired logic.

Domain Independence. Works are considered domain-independent if they have not been designed for specific application scenarios or domains.

[39] http://www.w3.org/TR/CCPP-struct-vocab/
[40] http://www.w3.org/Mobile/mobile-web-app-state/#Device_Adaptation
[41] http://www.w3.org/TR/css3-mediaqueries/

	NAC [Lemlouma 2004]	Laakko et al. [Laakko 2005]	Chen et al. [Chen 2005]	Mobile Web[Zhang 2007]	Chamaleon[Nathanail 2007]	Butter et al.[Butter 2007]	Paternò et al. [Paternò 2010]	MIMOSA [Malandrino 2010]	CAMB[Gasimov 2010]	Adipat et al.[Adipat 2011]	COIN [Woensel 2011]	CSS Media Queries[41]	**PRISSMA Framework**
Linked Data support											○		●
Context-awareness	●	●		○	●	●	○	●	●		●		●
Standard Languages	●	○	●		●	○			●		●	●	●
Runtime adaptation			○		●	○	○				●	●	●
Multimodality					○	○						●	○
Client-side only						●				●	●	●	●
Evaluation	●		●		○	○		●		○		●	○

Table 4.1: A comparison of adaptive visualization frameworks for Web resources on mobile devices. Full support is identified by ●, partial support by ○, no support by the empty cell.

Standard Languages. The adoption of standard languages (e.g. RDF, RDF-S/OWL, SPARQL) is assessed. If the work uses ad-hoc languages besides standard solutions, it is considered to be partially standard compliant.

Context Awareness. Specifies if the system supports full-fledged context awareness, as defined by Schilit et al. [Schilit 1994]. Partial support means that context features are presented, but limited to a small set of context dimensions.

Automatic Stylesheets. Assesses the ability of generating presentation-level declarations automatically, without the need for an a-priori template creation phase.

Evaluation. The evaluation is considered to be present when performance analysis has been carried out.

Distribution. The presence of a distribution system for presentation knowledge is assessed, i.e. a mechanism to share and exchange definitions.

Multimodality. It indicates if the framework supports multimodality, i.e. more than one output format, e.g. HTML text, audio, etc.

The Haystack platform UI, called Ozone, is one of the earliest works targeting RDF presentation [Huynh 2002]. Gandon focuses on creating effective UI representations of SPARQL queries over RDF repositories [Gandon 2005]. The focus of the paper is to provide UI designers with a tool to transform RDF to interface representations, under the runtime variability of the semantic Web scenario (where data schema are not known in advance). Quan and Karger present Xenon [Quan 2005],

an RDF stylesheet language based on XSL transformations (XSLT), and on a stylesheet ontology. The peculiarity of Xenon is the composition of stylesheets created by different authors. Xenon generated output is in HTML format. Champin presents Tal4RDF, a template language for textual output of RDF [Champin 2009]. Auer et al. propose an end-to-end template-based approach to consume linked data [Auer 2010]. Their system, LESS, is a template-based framework for RDF that manages template definition, template processing, integration, authoring and sharing. Dadzie et al present a template-based visualization framework for Linked Data [Dadzie 2011]. The proposal takes into account user context, but only in a hardwired, implicit manner. Fernéandez et al. propose a formal Linked Data Visualization Model (LDVM) [Fernéandez 2012]. They focus on visualizing data from large-scale RDF datasets, without being domain-dependent.

None of the reviewed works address the problem of adaptation to context.

	Haystack Ozone[Huynh 2002]	Noadster[Rutledge 2005]	Surrogates[Gandon 2005]	Fresnel[Pietriga 2006]	Xenon[Quan 2005]	Tal4Rdf[Champin 2009]	LESS[Auer 2010]	Hide the Stack[Dadzie 2011]	LDVM[Fernéandez 2012]	**PRISSMA Framework**
Declarative approach	•		•	•	•	•	•	•	•	•
Domain Independence	•	•	•	•	•	•			•	•
Standard Languages	•		•	•	•			•	•	•
Context Awareness			○							•
Automatic stylesheets			○							
Evaluation								•		•
Distribution							•			○
Multimodality			•	•		○	○			•

Table 4.2: A comparison of presentation layers for the Semantic Web. Full support is identified by •, partial support by ○, no support by the empty cell.

Among the existing presentation-level frameworks for RDF, an important role is played by Fresnel [Pietriga 2006]. Fresnel is a rendering engine for RDF, backed by an ontology of presentation-level concepts. Fresnel has been created to ease the display of RDF in Linked Data applications, and avoid dealing with presentation-level issues in application logic. The philosophy of Fresnel has largely influenced the work presented in this Chapter, although no support is given to contextual adaptation, since Fresnel is not designed for pervasive environments. Fresnel is built on the assumption that data and its related schema do not carry sufficient information for representing triples, hence it provides additional presentation-level knowledge in the Fresnel ontology[42], a vocabulary of concepts useful for displaying triples. Linked data application developers create Fresnel declarations for the instances (or classes

[42] http://www.w3.org/2004/09/fresnel

of instances) that will be displayed by their applications. The process is somehow similar to writing CSS for HTML pages. The Fresnel vocabulary is built on the separation between *data selection* and *formatting*. Data selection (including filtering) is implemented by Fresnel *Lenses*, while *Formats* define how to present data. Lenses and Formats are arranged in *Groups*, for the sake of clarity[43] (Figure 4.1):

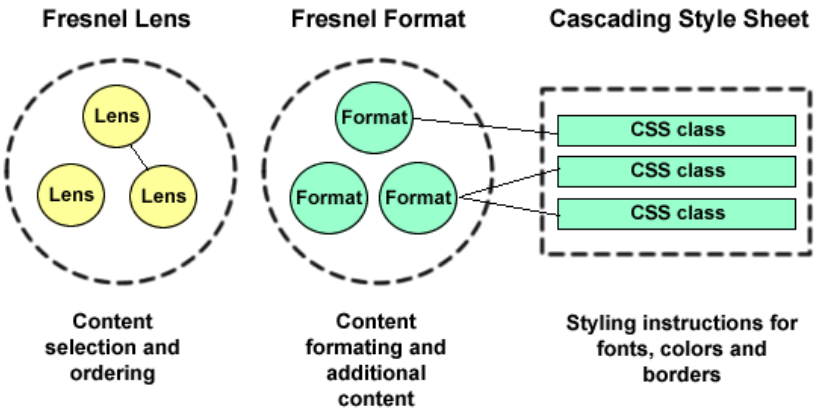


Figure 4.1: Fresnel Lenses and Formats at a glance[44]

Lens. A Fresnel lens specifies which properties must be shown and their order. A lens can be associated to a specific RDF instance or to a class using the properties `fresnel:instanceLensDomain` and `fresnel:classLensDomain`. The properties to show are specified with the `fresnel:showProperties` property and they respect a given ordering.

Format. Formats describe how to label properties, how to display property values, add additional content to properties (e.g. commas and periods, useful when listing values), and reference to external CSS files. The latter feature allows to adopt standard CSS declarations when formatting triples in HTML output. Formats are associated to target resource with the `fresnel:propertyFormatDomain` property. Property labels can be added with `fresnel:label`. The type of output element is specified with `fresnel:value` (e.g. HTML `img` tag). Values and labels can be assigned external CSS classes with the `fresnel:valueStyle` and `fresnel:labelStyle` properties.

Group. A Fresnel Group is a wrapper for Lenses and Formats defined on related targets.

Figure 4.2 shows a complete Fresnel declaration for visualizing a `foaf:Person`: the Group `:foafGroup` (line7) includes a Lens `:foafLens` (lines 10-15) and two formats,

[43] An more comprehensive description of the Fresnel ontology can be retrieved from http://www.w3.org/2005/04/fresnel-info/manual/

[44] Image retrieved from http://www.w3.org/2005/04/fresnel-info/manual/

```
1  @prefix : <http://example.org#> .
2  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
4  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5  @prefix fresnel: <http://www.w3.org/2004/09/fresnel#> .
6
7  :foafGroup a fresnel:Group;
8       fresnel:stylesheetLink <http://www.example.org/example.css> .
9
10 :foaflens a fresnel:Lens;
11          fresnel:group :foafGroup;
12          fresnel:classLensDomain foaf:Person;
13          fresnel:showProperties ( foaf:name
14                                   foaf:surname
15                                   foaf:depiction ) .
16
17 :depictionFormat a fresnel:Format ;
18          fresnel:propertyFormatDomain foaf:depiction ;
19          fresnel:label fresnel:none;
20          fresnel:value fresnel:image ;
21          fresnel:valueStyle "depiction"^^fresnel:styleClass ;
22          fresnel:group :foafGroup.
23
24 :nameNickFormat a fresnel:Format ;
25         fresnel:propertyFormatDomain foaf:name, foaf:surname ;
26         fresnel:label "Name:";
27         fresnel:labelStyle "label"^^fresnel:styleClass ;
28         fresnel:valueStyle "value"^^fresnel:styleClass ;
29         fresnel:group :foafGroup .
```

Figure 4.2: A sample Fresnel declaration. The declaration affects `foaf:Person` instances: foaf:name, foaf:surname and foaf:depiction properties are displayed, according to CSS classes defined in an external CSS file.

`:depictionFormat` (lines 17-22) and `:nameNickFormat` (lines 24-29). `:foafGroup` references to an external CSS (line 8). `:foafLens` is defined on foaf:Person entities (line 11), and is supposed to show the name, the surname, and the depiction of the `foaf:Person` (lines 13-15). `:depictionFormat` defines how the depiction must be presented. First, it adds an empty label (line 19). Then it defines the type of unit, `fresnel:image` (line 20) and associates it to a CSS class defined in the external CSS referenced by `:foafGroup` (line 21). The format `:nameNickFormat` is activated for the properties `foaf:name` and `foaf:surname` (line 25), and formats them according to the external CSS class `value` (line 28).

An interesting additional feature of Fresnel is the support for different output media[45] with the property `fresnel:purpose`, that can be associated to groups, lenses, or formats. Predefined values are provided, such as `screen`, `print` and `projection`, thus supporting limited media-based adaptation.

	iSPARQL[Kiefer 2007]	Silk[Volz 2009]	Zou et al.[Zou 2012]	Messmer and Bunke[Messmer 1998]	**PRISSMA Framework**
RDF-specific	•	•	•		•
Data Heterogeneity		○		○	•
Client-side Execution				○	•
Incremental index updates	•			•	•
Selective matching cache				○	○

Table 4.3: A comparison of error-tolerant matching techniques for RDF. Full support is identified by •, partial support by ○, no support by the empty cell. The table compares the works along a number of features, starting from the native support for RDF. Dealing with heterogeneous data is crucial when handling context information. Client-side execution capabilities are assessed, along with the capacity of index updates at runtime, and the option for selective matching cache.

4.3.3 Error-tolerant matching for RDF Graphs

Context declarations are graph patterns that must be matched to the actual context graph: given that both context declarations and the actual context are RDF structures, the operation consists in testing a series of RDF subgraph equivalences. Several requirements must be taken into account (Table 4.3 compares the reviewed works against such requirements):

Error-tolerance. Context declarations might be ambiguous or incomplete, and their interpretation might vary according to users. Besides, a portion of context data is originated by onboard sensors, that are error-prone and might provide imprecise information. Hence, the need for an error-tolerant matching strategy, i.e. a technique that takes into account possible discrepancies between context declarations and actual context.

RDF-specific. The adopted solution must be designed to work in conjunction with RDF graphs.

Data Heterogeneity. The mechanism must support heterogeneous context dimensions (e.g. location, time).

Client-side Execution. Since the Prism selection procedure must run on the client-side - to avoid disclosing sensitive context information - we must design an algorithm suitable for resource-constrained devices.

[45] http://www.w3.org/2005/04/fresnel-info/manual/#MultipleOutputs

Incremental Index Update. The adopted strategy must support runtime updates of the number of RDF graphs, as context declarations might be fetched from remote repositories and added to the selection process at runtime.

Selective matching cache. Due to the changing nature of context data, certain context dimensions vary more often than others (e.g. location vs user profile information). The matching strategy must preferably cache the results for slowly-changing context data, thus concentrating on newly changed information.

Error-tolerant techniques for RDF instance matching are surveyed in [Castano 2011]; they are mostly related to ontology matching strategies, reviewed by Euzenat and Shvaiko [Euzenat 2007]. Off-the-shelf SPARQL engines do not match all our requirements: for instance, the most suitable work for our prerequisites, iSPARQL [Kiefer 2007], is designed for error-tolerant matching, but it neither supports heterogeneous dimensions (such as location), nor is it designed for computationally-constrained mobile platforms. The Silk framework [Volz 2009], designed to interlink Linked Data instances, uses similarity metrics for RDF. Although it includes geographical and time distances, such metrics do not consider data imprecision. Furthermore, Silk is not designed to run on mobile devices. RDF semantics states that two RDF graphs are semantically equivalent if they entail one another[46], and, as underlined by Carrol [Carroll 2002], the important concept for entailment between RDF graphs is *subgraph isomorphism*, that, incidentally, is known to be a NP-complete problem. Subgraph isomorphism is at the heart of a recent pattern matching engine for SPARQL by Zou et al. [Zou 2012]. Unfortunately, the authors do not provide an error-tolerant version of their algorithm. The problem of *error-tolerant* subgraph isomorphism has been extensively studied in graph theory and is mentioned in the area of pattern recognition, for example in the comprehensive survey by Conte et al. [Conte 2004]. It has been proved [Conte 2004] that finding the optimal error-tolerant subgraph isomorphism between two graphs can be reduced to the computation of *graph edit distance*: the idea is that differences between graphs can be modelled in terms of operations to apply to graphs, such as adding a node or modifying an arc. Graph edit distance provides the required flexibility for building an error-tolerant subgraph matching algorithm, and supports customized and heterogeneous cost functions (comparing contexts means dealing with heterogeneous data such as location, time, string literals, URIs). Nevertheless, computational complexity is exponential in the number of graph nodes (i.e. it can be computed only for small graphs): the reason for this high complexity lies on the fact that graph edit distance algorithms assume that every node can be mapped on every node of another graph [Gao 2010]. Although context descriptions are rather small graphs, computing graph edit distance remains a computationally expensive task, in particular on mobile devices. Traditional approaches to compute graph edit distance between an input graph and a set of reference graphs apply a pairwise

[46] http://www.w3.org/TR/rdf-concepts/

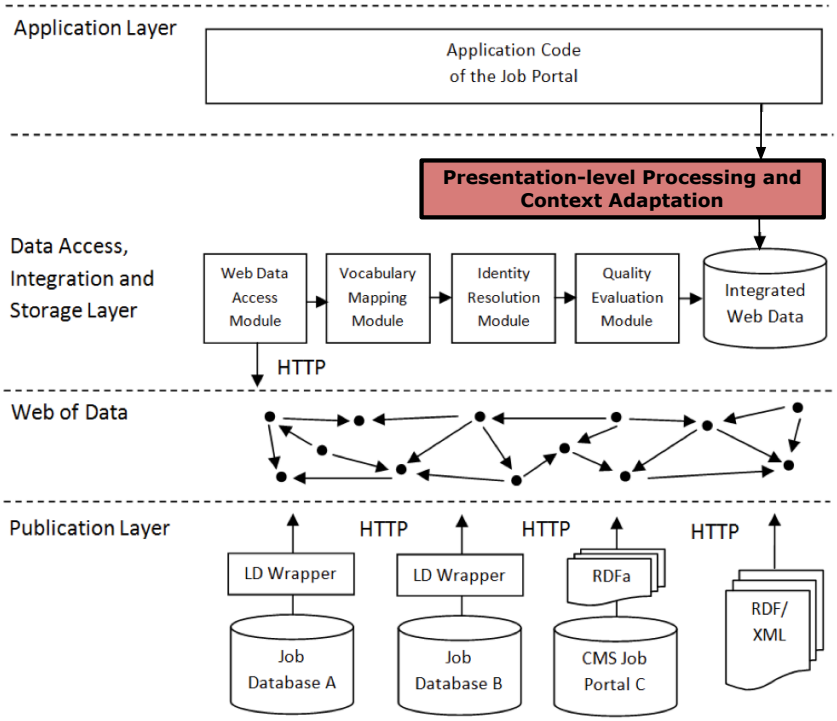
comparison, as shown by the survey by Gao et al. [Gao 2010]. Such methods do not scale well and badly perform with runtime updates [Messmer 1998]. A series of works compute graph edit distance by merging graphs into a single data structure, thus avoiding pairwise comparison [Gao 2010]: One of these works is the algorithm proposed by Messmer and Bunke for directed, labeled graphs [Messmer 1998]: the core idea of this technique is to fragment offline model graphs into smaller subgraphs that are stored into a single data structure, the so-called *decomposition*, and apply a search algorithm at runtime on this decomposition structure. More precisely, each graph is decomposed recursively in two subgraphs, until the remaining subgraphs consist in single nodes. The resulting subgraphs are stored in a tree-like structure, a common storage for all decomposed model graphs. The advantage of this approach is that subgraphs that are repeated in different graphs are collapsed in the decomposition and represented only once, thus providing a compact representation of model graphs. This feature is important in a memory-constrained mobile scenario, especially when the stored graph structures share the same background ontology (thus having a high chance of having triple patterns in common). Given an input graph, an online search algorithm searches in the decomposition for the error-tolerant subgraph isomorphisms with the lowest edit costs, starting from smaller subgraphs. The chosen subgraphs are recursively combined to find *optimal* (i.e. least expensive) error-tolerant subgraph isomorphisms from model graphs to the input graphs. Since common subgraphs are stored only once, the strategy guarantees sublinear complexity with respect to the number of model graphs in the system.

4.4 PRISSMA Presentation Framework: Overview

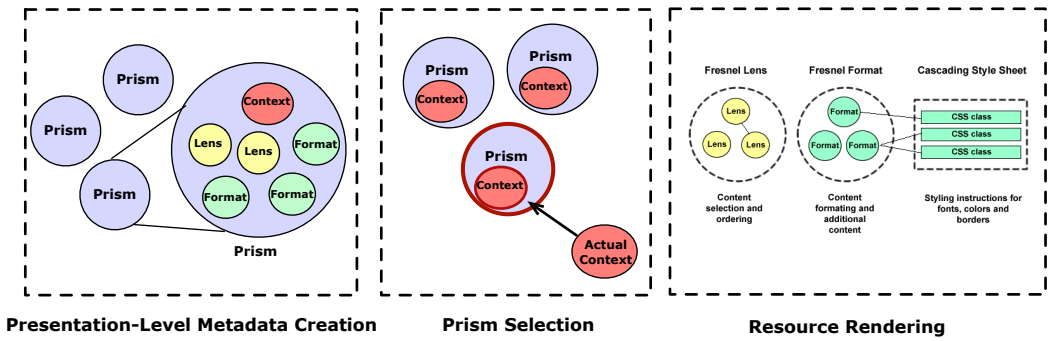
The process of rendering RDF resources with the PRISSMA presentation framework includes three steps (Figure 4.3b). First, presentation-level directives must be created, along with the description of the client context in which these directives must be used for rendering. Second, PRISSMA selects the most appropriate presentation-level directive, according to the current client context. The selected presentation-level information is used to drive the third step, the rendering process, carried out by Fresnel.

4.4.1 Combining Fresnel and PRISSMA Vocabularies

PRISSMA adopts Fresnel declarations for describing *how* RDF resources must be visualized. Apart from declaring these presentation-level directives, the first step needed by a developer of context-aware RDF interfaces is *describing the contextual conditions* in which a given group of Fresnel directives must be activated. This problem brings to our first research question, i.e. *how to use context descriptions at RDF presentation-level*. We propose to turn the PRISSMA vocabulary (Chapter 3) into an extension of the Fresnel presentation ontology, so that the resulting vocabulary specifies *when* (i.e. in which mobile context) Fresnel Lenses and Formats must



(a)



(b)

Figure 4.3: (a) The architecture of a Linked Data application, as described by Heath and Bizer [Heath 2011], with the adding of the presentation layer component. (b) The components of the PRISSMA presentation layer framework. The "Resource Rendering" block corresponds to the Fresnel engine [Pietriga 2006].

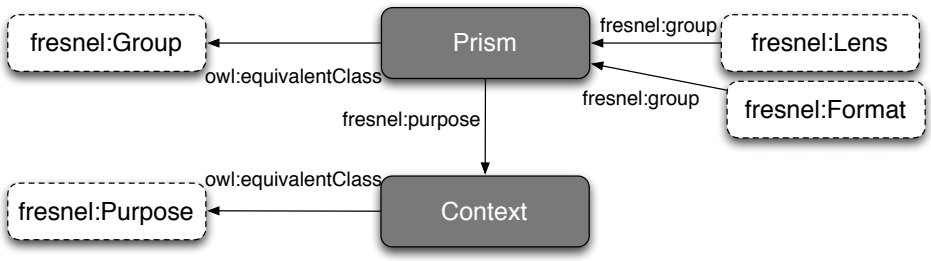


Figure 4.4: The interplay of PRISSMA and Fresnel vocabularies

be selected. Our solution (Figure 4.4) extends the semantics of `fresnel:Purpose` to delegate the selection of Lenses and Formats to a broader and more expressive definition of mobile context, modeled by the `prissma:Context` class. To wrap up each context-aware presentation-level unit of information, the concept of *Prism* is introduced (a Prism is `owl:equivalentClass` to a `fresnel:Group`):

Definition 7 (Prism) *A Prism P is an RDF graph that describes the contextual conditions under which a given RDF presentation must be activated.*

Figure 4.5 shows the sample Prism `:foafPrism` (lines 11-14). The Prism, equivalent to a `fresnel:Group`, styles a `foaf:Person` in the client context `:sampleCtx1`. Fresnel-related triples (lines 17-36) are coupled to the PRISSMA context description of lines 39-56 with the `fresnel:purpose` property in line 13, thus creating a context-dependent Fresnel declaration.

Operating content adaptation at presentation level has a number of benefits. First, as we explained in [Costabello 2011], context data must be considered as first-class presentation knowledge, and modelled at presentation level to support multiple outputs: mobile devices support multiple heterogeneous output paradigms that may change dynamically according to context (e.g. switching from HTML to text-to-speech[47]). Another benefit of working at presentation level is the freedom from the adopted linked data access strategy: HTTP resources dereferencing, SPARQL query results and semantic search engines APIs responses are all supported. Besides, presentation knowledge must be *declarative*, to favour presentation data reuse between different applications. As shown by Fresnel [Pietriga 2006], the need for sharing and reusing contextual presentation is not compatible with hard-wired, programmatic approaches. This is why PRISSMA presentation-level metadata is expressed in RDF. Moreover, we do not introduce a new language or formalism, and we thus benefit from standard RDF tools, that can be used to process presentation knowledge. Adopting RDF means that presentation knowledge can be embedded inside applications, provided as metadata by datasets administrators or published on the web as linked data by third parties. In the latter case, mobile applications should be able to discover such *linked presentation knowledge* using context-aware distribution heuristics. For instance, PRISSMA might be coupled to follow-your-nose strategies

[47] `http://www.w3.org/Voice`

```
@prefix : <http://example.org#> .
@prefix prissma: <http://ns.inria.fr/prissma/v2#> .
@prefix fresnel: <http://www.w3.org/2004/09/fresnel#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .
@prefix ao: <http://purl.org/ontology/ao/core#> .

# Prism to style a FOAF profile in :sampleCtx1
:foafPrism a prissma:Prism ;
          a fresnel:Group ;
         fresnel:purpose :sampleCtx1 ;
         fresnel:stylesheetLink <http://www.example.org/example.css> .

# Fresnel presentation-level triples
:foaflens a fresnel:Lens;
         fresnel:group :foafPrism;
         fresnel:classLensDomain foaf:Person;
         fresnel:showProperties ( foaf:name
                                  foaf:surname
                                  foaf:depiction ) .

:depictionFormat a fresnel:Format ;
         fresnel:propertyFormatDomain foaf:depiction ;
         fresnel:label fresnel:none;
         fresnel:value fresnel:image ;
         fresnel:valueStyle "depiction"^^fresnel:styleClass ;
         fresnel:group :foafPrism.

:nameNickFormat a fresnel:Format ;
        fresnel:propertyFormatDomain foaf:name, foaf:surname ;
        fresnel:label "Name:";
        fresnel:labelStyle "label"^^fresnel:styleClass ;
        fresnel:valueStyle "value"^^fresnel:styleClass ;
        fresnel:group :foafPrism .

# PRISSMA context description
:sampleCtx1 a prissma:Context ;
   prissma:user :geek ;
   prissma:environment :env1 .

:geek a prissma:User ;
   foaf:interest "computer programming" ,
                 <http://dbpedia.org/resource/Star_Trek> .

:env1 a prissma:Environment ;
   prissma:poi :poi1 ;
       ao:time :time1 .

:poi1 geo:lat "45.43463" ;
     geo:long "7.843435" ;
     prissma:radius "200" .

:time1 tl:start "2013-04-05T10:00:00Z"^^xsd:dateTime ;
      tl:duration "PT6H"^^xsd:duration .
```

Figure 4.5: A sample PRISSMA Prism. The Prism consists in the interplay of the `fresnel:Group` defined in Figure 4.2 with the `prissma:context` shown in Figure 3.2a.

to retrieve and fetch the most suitable set of PRISSMA declarations for a given domain in a specific context.

4.4.2 The Problem of Prism Selection

Before rendering an RDF resource with Fresnel, PRISMA-equipped applications search the available Prisms and select the better match for the context in which the desired resource is accessed, thus our second research question: *how to select the proper context description at runtime?* The most relevant challenge of this task is dealing with the imprecise and incomplete nature of context data, that complicates the matching procedure between declared and sensed contexts, and requires an error-tolerant approach. As mentioned in Section 4.4, since context is a vague concept, Prisms might be ambiguous and incomplete. Moreover, the interpretation of a context description might vary according to the developer. Part of context data is originated by onboard sensors, that might be imprecise and error-prone. To summon up, context data is riddled by the following issues:

Ambiguity. Some RDF Entities and literals used in PRISSMA declarations might not match with the actual context entities. Nevertheless, in some cases entities and literals might be *similar*. As explained in Section 4.3, this is a well-known problem in the Web of Data community (e.g. ontology and instance matching).

Incompleteness. The authors of PRISSMA context declarations might omit or forget certain properties, when describing a context. Nevertheless, in certain cases the context graph, although topologically different, should still be considered as a valid candidate by the selection algorithm.

Sensor noise. Onboard sensors might provide erroneous information that will be part of the actual context graph [Henricksen 2004]. This is a well-known problem when determining geographic location (e.g. when GPS signal is weak).

Figure 4.6 contains two PRISSMA context declarations. Figure 4.6a describes a user interested in "computers", located in downtown Manhattan close to Jack. Figure 4.6b describes a similar context situation: this time the user is interested in "computer programming" and "computer science" (in terms of string similarity, both close to "computers"). The user is still close to Jack, but location is slightly different (the point is 20 meter far from the location in Figure 4.6a). Moreover, instances identifiers for objects modelled by the same classes differ. Although the graphs differ in their topology and in the node values, under certain scenarios they might still be considered as representing two very similar context situations, if not the same one.

To solve the the problem of Prism selection, a dedicated algorithm has been designed. The Prism selection algorithm handles context matching with an *error-tolerant* strategy that takes into account the aforementioned issues. The algorithm is thoroughly described in Section 4.5.

```
:prismA a prissma:Prism;
    fresnel:purpose :ctxA.

:ctxA a prissma:Context ;
     prissma:environment :envA ;
     prissma:user :usrA .

:usrA a prissma:User ;
     foaf:interest "computers" .

:envA a prissma:Environment ;
     prissma:nearbyEntity
              <http://jack.example.org> ;
     prissma:poi :poiA .

:poiA geo:lat "40.7489476" ;
     geo:long "-73.9844227" ;
     prissma:radius "100" .
```

(a)

```
:ctxActual a prissma:Context ;
     prissma:environment :envActual ;
        prissma:user :usrActual .


:usrActual a prissma:User ;
     foaf:interest "computer science",
       "computer programming" .


:envActual a prissma:Environment ;
     prissma:nearbyEntity
       <http://jack.example.org> ;
     prissma:poi :poiActual .


:poiActual geo:lat "40.7584403" ;
     geo:long "-73.9861822" .
```

(b)

Figure 4.6: Two similar contexts (prefixes are omitted). A context entity wrapped in a `prissma:Prism` declaration, left, and a sample client context, right. Although not identical, both context graphs represent similar context descriptions.

4.4.3 Resource Rendering

Once the selection algorithm has chosen the most appropriate Prism, the embedded Fresnel declarations are fed to the Fresnel rendering engine. The Fresnel rendering engine is backed by the Fresnel ontology, and is designed to create different outputs (e.g. HTML, PDF, plain text, etc.) along with different paradigms (HTML-like nested boxes, node-link diagrams, etc.). More specifically, the Fresnel rendering process is split in the following steps:

Content Selection. RDF resources have a number of properties that might be filtered before delivering content to users. Fresnel uses lenses to select and reorder the desired properties. This selection step is carried out by declaring Fresnel lenses using the Fresnel ontology. No formatting information is inserted by the rendering engine up to this point.

Content Formatting. Having selected the desired triple to show, developers define formatting directives, called Fresnel formats. As for lenses, formats are RDF triples and are defined by the Fresnel ontology.

Output Generation. The Fresnel engine uses pre-declared lenses and formats to generate the appropriate output and paradigm, when an RDF resources must be visualized. Fresnel does not specify neither output nor paradigm, as this is left to Fresnel implementations[48]. Nevertheless, all current implementations support the generation of HTML output, with nested-box paradigm.

[48] http://www.w3.org/2005/04/fresnel-info/#implementation

4.5 Prism Selection Algorithm

As mentioned in Section 4.4.2, the Prism selection step must take into account the ambiguity, the incompleteness, and the noise of context data. Hence, the Prism selection algorithm must handle context matching with an error-tolerant strategy that takes into account the aforementioned issues. Prisms are graph patterns that must be matched to the actual context graph: given that both context declarations and the actual context are RDF structures, the operation consists in testing a series of RDF subgraph equivalences. To comply with the algorithm requirements described in Section 4.3.3, we extended and adapted to RDF the Messmer and Bunke error-tolerant algorithm for finding *optimal* subgraph isomorphisms for labelled, directed graphs [Messmer 1998]. This section provides the adopted definitions, the data structures, the algorithm, and a series of examples. The strategy relies on the fact that the optimal error-tolerant subgraph isomorphism problem can be reduced to the computation of *graph edit distance* [Conte 2004]: retrieving the optimal error-tolerant subgraph isomorphism problem is therefore solved by determining the least expensive sequence of edit operations. More precisely, each graph is decomposed recursively in two subgraphs, until the remaining subgraphs consist in single nodes. Every subgraph is stored in a tree-like structure, called *decomposition*, a common storage for all decomposed model graphs. The advantage of this approach is that subgraphs that are repeated in different graphs are collapsed in the decomposition and represented only once, thus providing a compact representation of model graphs. This feature is important in a memory-constrained mobile scenario, especially when the stored graph structures share the same background ontology (thus having a high chance of having triple patterns in common). Furthermore, such approach supports runtime updates of RDF graphs: context descriptions are added to the decomposition in an incremental manner, and the structure does not have to be re-built from scratch. Given an input graph, an online search algorithm searches in the decomposition for the error-tolerant subgraph isomorphisms with the lowest edit costs, starting from smaller subgraphs. The chosen subgraphs are recursively combined to find *optimal* (i.e. least expensive) error-tolerant subgraph isomorphisms from the model graphs to the input graphs. Since common subgraphs are stored only once, the strategy guarantees sublinear complexity with respect to the number of model graphs in the system.

4.5.1 Definitions

Before describing the adapted algorithm, we remind some useful definitions provided in Messmer [Messmer 1998], adjusting them to our scenario:

Definition 8 (RDF Graph) *An RDF graph is a set of RDF triples* $G = \{(s_1, p_1, o_1) \dots (s_n, p_n, o_n)\} = (V, E)$ *where* s_t *is the subject,* p_t *the property and* o_t *the object of each triple* t*.* V *is the set of labelled vertices and contains the elements* s_t *and* o_t*, that are entities or literals.* E *is the set of directed edges and contains all the triple properties* p_t*.*

Definition 9 (Graph Edit Operation) *Given an RDF graph $G = (V, E)$, a graph edit operation $\delta(G)$ is one of the following:*

- $v \rightarrow v',\ v \in V, v' \in V$ *(substituting an RDF entity or literal)*
- $e \rightarrow e',\ e \in E, e' \in E$ *(substituting an RDF property)*
- $v \rightarrow \varepsilon,\ v \in V$ *(deleting an RDF instance or literal)*
- $e \rightarrow \varepsilon,\ e \in E$ *(deleting an RDF property)*
- $\varepsilon \rightarrow e,\ e \in E$ *(adding an RDF property between existing nodes)*

where ε is an empty RDF entity, literal, or property.

These five edit operations are sufficient to transform any graph G into the subgraph of any graph G'. Note that the algorithm searches for subgraph isomorphisms from a model graph to the input graph, hence there is no need to consider exterior RDF instances or literals in the input graph, i.e. there is no need for a $\varepsilon \rightarrow v,\ v \in V$ operation.

Definition 10 (Edited Graph) *Given an RDF graph G and a sequence Δ of edit operations, the edited graph $\Delta(G) = (\Delta(V), \Delta(E))$ is the graph $\Delta(G) = \delta_n(\dots \delta_1(G))$.*

Definition 11 (Error-Tolerant RDF Subgraph Isomorphism) *Given two RDF graphs $G = (V, E)$ and $G' = (V', E')$, an error-tolerant RDF subgraph isomorphism f from G to G' is a two-tuple $f = (\Delta, f_\Delta)$ where:*

- *Δ is a sequence of graph edit operations that transforms G in $\Delta(G)$.*
- *f_Δ is an injective function $f_\Delta : \Delta(V) \rightarrow V'$ such that $\exists$ a graph isomorphism[49] from $\Delta(G)$ to a subgraph $S \subseteq G'$.*

We now introduce the definition of *cost of error-tolerant subgraph isomorphism*, preceded by the *cost* of an edit operation:

Definition 12 (Cost of Edit Operation) *Given an edit operation δ_i, the cost of δ_i is a value $C(\delta_i) \in [0, 1]$.*

The cost $C(\delta_i)$ of an edit operation δ_i varies according to the type of edit operation (e.g. instance substitution, property deletion, etc.) and the nature of the involved RDF element. We cover in more details $C(\delta_i)$ in Section 4.5.5.

Definition 13 (Cost of Error-Tolerant RDF Subgraph Isomorphism) *Given an error-tolerant RDF subgraph isomorphism $f = (\Delta, f_\Delta)$, its cost $C(f)$ is defined as the normalized cost of the sequence of edit operations $\Delta = (\delta_1, \dots \delta_n)$, $C(f) = \frac{C(\Delta)}{n} = \frac{\sum_{i=1}^{n} C(\delta_i)}{n}$.*

[49] We rely on the definition of graph isomorphism provided in [Messmer 1998].

The cost of error-tolerant subgraph isomorphism described in Definition 13 adopts the arithmetic mean to normalize the cost of the sequence of edit operations. Other strategies might be adopted, such as using a weighted mean (different weights might be associated to each edit operation type), or choosing the maximum cost in the sequence. To date, this work does not assess the impact of different strategies to compute the cost of a sequence of edit operations, and such task is left for future work.

It is evident that there might exist multiple sequences Δ of edit operations from graph G to graph G', each with a different cost: we are interested in finding the *optimal error-tolerant subgraph isomorphism*, i.e. the error-tolerant subgraph isomorphism with the least expensive sequence of edit operations. In other words, we want to find the minimum amount of distortion needed to transform a Prism into the actual mobile context, thus computing their graph edit distance [Riesen 2010]:

Definition 14 (Optimal Error-Tolerant RDF Subgraph Isomorphism) *Given a set of error-tolerant subgraph isomorphisms $F = f_1 \dots f_n$ between two graphs, the optimal error-tolerant subgraph isomorphism f_{opt} is the element of F with cost $C(f_{opt}) = \min_{f_i \in F} C(f_i)$.*

In the remainder of the Section we explain how we adapted to our scenario the original graph edit distance-based error-tolerant subgraph isomorphism algorithm by Messmer and Bunke [Messmer 1998]. We describe i) the construction of the decomposition structure and ii) the runtime search algorithm.

4.5.2 Decomposition

The context-related triples included in each Prism are split in subgraphs and saved in a structure called *decomposition*, a recursive partitioning of a set of RDF models (Prisms). The decomposition algorithm works on the set of Prisms pre-loaded by the PRISMA-equipped mobile application. The idea is building the decomposition by detecting and merging common subgraphs: in the decomposition, subgraphs duplicated in different Prisms are collapsed and represented only once, thus providing a compact representation of possible contexts. As remarked by Messmer and Bunke, there exists more than one decomposition for a set of graphs: the adopted strategy does not provide an optimal decomposition (e.g. in the number of elements), but it is computationally inexpensive compared to other strategies [Messmer 1998].

The elements of a decomposition are tuples that include graph patterns sharing the same topology and whose RDF elements have the same classes. Among the decomposition elements, some consist in groups of non-decomposable, atomic graph patterns called *context units*:

Definition 15 (Context Unit) *A context unit is an RDF graph $U = (V_U, E_U)$ representing atomic context information. A context unit U consists in either a single class, or a single RDF entity, or a single literal, or in a graph that describes an atomic context information.*

In the original proposition, Messmer and Bunke deal with graphs with a limited range of discrete values, thus they decompose graphs up to single nodes. In our scenario we must compare more complex structures, hence the need to preserve context units. For instance, we cannot split latitude, longitude, and radius without compromising the comparison of two geographic locations. Thus, different types of context units have been defined, according to the type of context information: "Class" context units consist in core PRISSMA classes (e.g. `prissma:Context`, `prissma:User`, `prissma:Environment`, and `prissma:Device`). "Entity" context units are RDF entities, whose class is not included in PRISSMA core classes. Entity context units may be blank nodes. "Geo" context units represent a geographic location, while "Time" elements include temporal information. Both Geo and Time context units may be blank nodes. "String" and "Numeric" context units are associated to string and numeric literals. "Class" context units are created by a preliminary step, where instances of core PRISSMA classes are substituted by their class. This is done to decrease the size of the decomposition structure without losing information, since the values (URIs) of such core instances are not important for matching purposes.

Definition 16 (Decomposition) *Given a set of prissms* $P = \{P_1, \ldots, P_n\}$*, the decomposition* $D(P)$ *is a set of 4-tuple* (G, G', G'', E) *where:*

1. G, G', G'' *are RDF graphs, with* $G', G'' \subset G$
2. E *is a set of RDF properties such that* $G = G' \cup_E G''$
3. *for each* P_i *there exists a 4-tuple* $(P_i, G', G'', E) \in D(P)$
4. *for each 4-tuple* (G, G', G'', E) *there exists no other 4-tuple* $(G_1, G'_1, G''_1, E) \in D(P)$ *with* $G = G_1$
5. *for each 4-tuple* $(G, G', G'', E) \in D(P)$
   (a) *if* G' *is not a context unit, there exists a 4-tuple* $(G_1, G'_1, G''_1, E) \in D(P)$ *such that* $G' = G_1$
   (b) *if* G'' *is not a context unit, there exists a 4-tuple* $(G_2, G'_2, G''_2, E) \in D(P)$ *such that* $G'' = G_2$
   (c) *if* G' *is a context unit, there exists no 4-tuple* $(G_3, G'_3, G''_3, E) \in D(P)$ *such that* $G' = G_3$
   (d) *if* G'' *is a context unit, there exists no 4-tuple* $(G_4, G'_4, G''_4, E) \in D(P)$ *such that* $G'' = G_4$

Algorithms 1 and 2 describe the decomposition procedure: in Algorithm 1, the recursive function *decompose*() is executed on each Prism P_i in the set P. The procedure *decompose*() described in Algorithm 2 searches in the decomposition for S_{max}, the biggest subgraph of G (lines 3-5): the goal is to determine if there exists

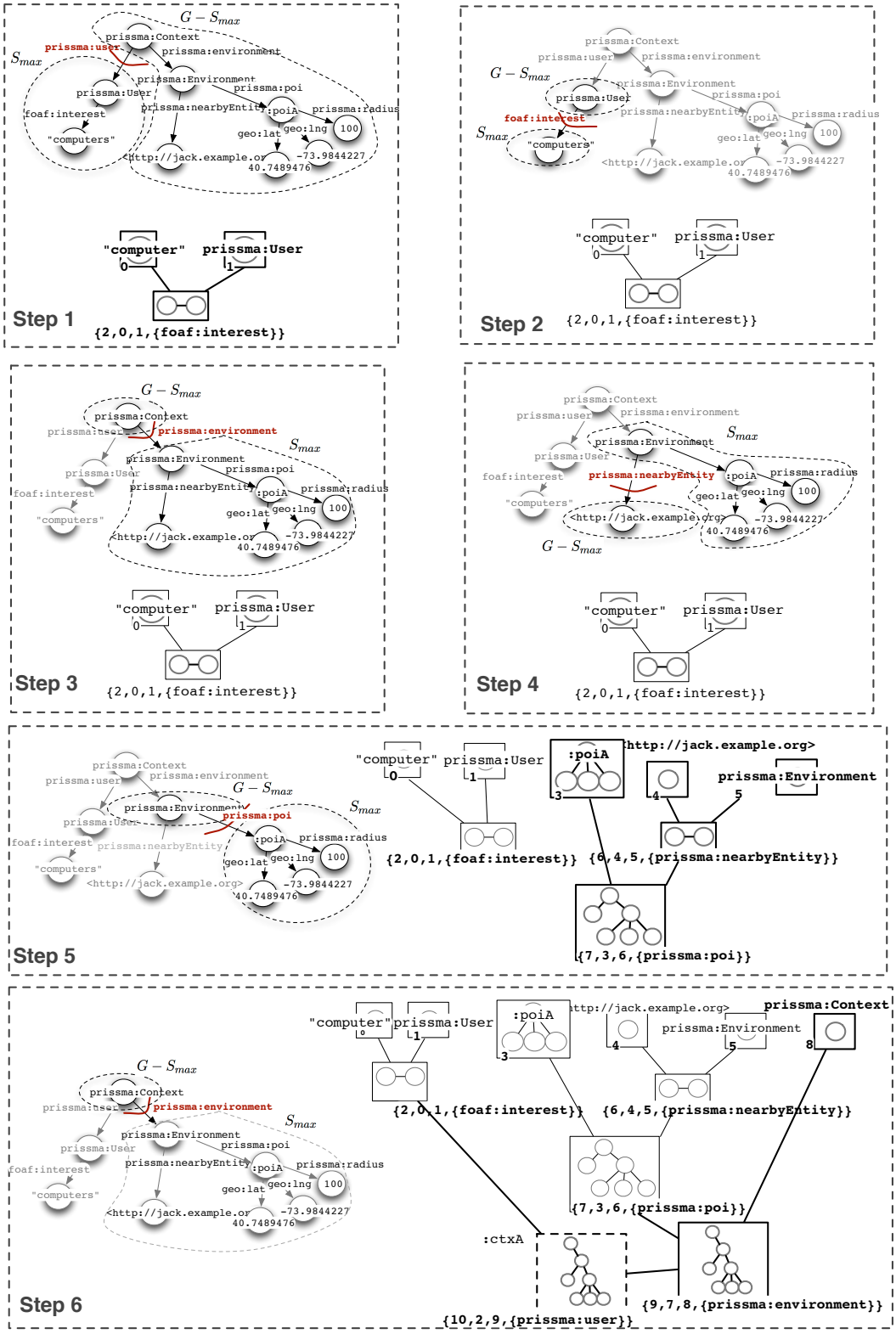


Figure 4.7: The *decomposition()* algorithm executed on `:PrismA` (triples in Figure 4.6a). Each step includes the choice of S_{max} (the cut property that connects S_{max} with $G - S_{max}$ is marked in red) and the decomposition (newly added elements are in bold.)

```
:prismA a prissma:Prism;
    fresnel:purpose :ctxA.

:ctxA a prissma:Context ;
      prissma:environment :envA ;
      prissma:user :usrA .

:usrA a prissma:User ;
      foaf:interest "computers" .

:envA a prissma:Environment ;
      prissma:nearbyEntity
               <http://jack.example.org> ;
      prissma:poi :poiA .

:poiA geo:lat "40.7489476" ;
      geo:long "-73.9844227" ;
      prissma:radius "100" .
```

(a)

```
:prismB a prissma:Prism;
      fresnel:purpose :ctxB.

:ctxB a prissma:Context ;
      prissma:environment :envB ;
      prissma:user :usrB .

:usrB a prissma:User ;
      foaf:interest "basketball" ;
      foaf:interest "cinema" .

:envB a prissma:Environment ;
      prissma:nearbyEntity
               <http://jack.example.org> ;
      ao:time :timeB .

:timeB tl:start "14:30:00"^^xsd:time ;
       tl:duration "PT2H"^^xsd:duration .
```

(b)

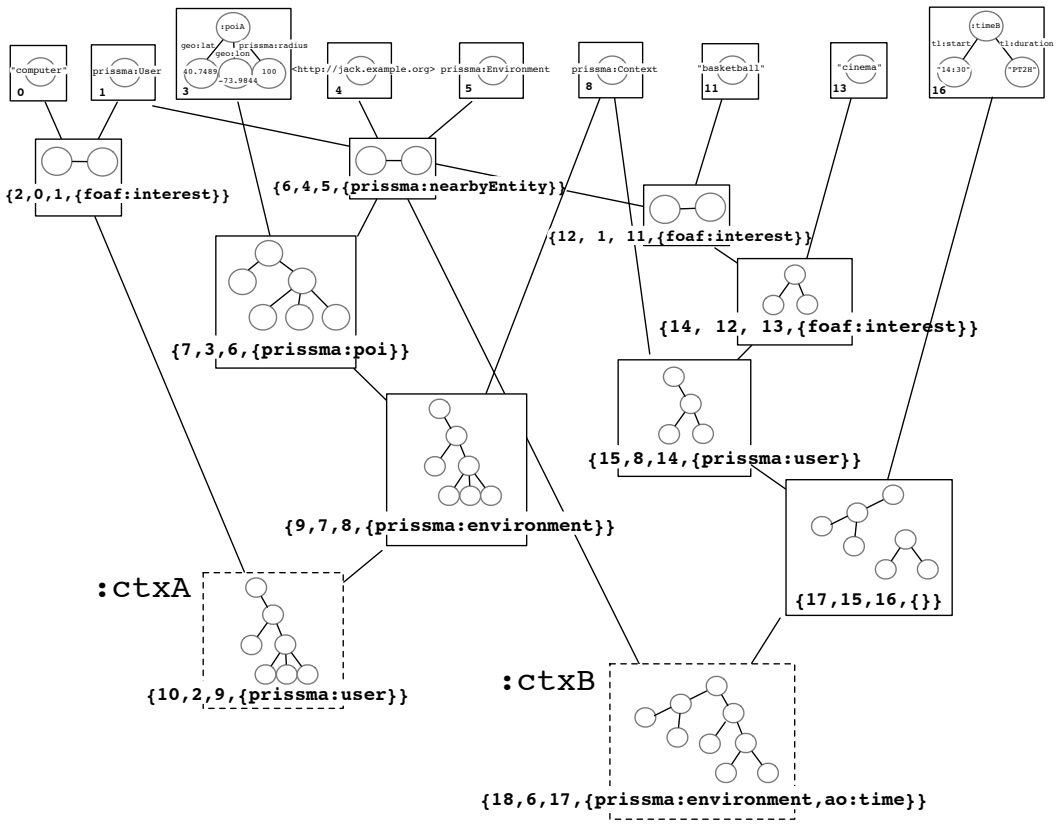


Figure 4.8: A graphical representation of the decomposition of `:PrismA` (a) and `:PrismB` (b).

a graph pattern in common with the decomposition[50]. If S_{max} is isomorphic[50] to G, then G is already represented in D and the algorithm stops (lines 6-7).

If no subgraph is found, and G can be further decomposed (i.e. is not a context unit), the procedure chooses S_{max} (line 9) and recursively decomposes it (line 10). The choice of S_{max} is determined by a list of ordered RDF properties, with a priority for PRISMA background ontology core properties. This enhances the chances of merging decomposition elements, thus resulting in a more compact structure. The procedure is invoked recursively on $G - S_{max}$, the part of G not yet decomposed (line 11). Finally, $(G, S_{max}, G - S_{max}, E)$ is added to D.

Example. Figure 4.7 shows the step-by-step decomposition of the context graph included in the Prism presented in Figure 4.6a. Note that the context graph has been pre-processed to create the "Class" context units (`prissma:Context`, `prissma:User`, and `prissma:Environment`). The first cut edge chosen is the core PRISSMA property `prissma:user` (note that core PRISSMA properties have higher priority), so that S_{max} and $G - S_{max}$ are defined as shown in the picture. The procedure *decompose*() is called recursively on S_{max}, thus generating S_{max} and $G - S_{max}$. The algorithm is then performed on S_{max}. The latter is a context unit ("class" type), and is therefore added to the decomposition and assigned a unique ID. Conversely, $G - S_{max}$ is added to the decomposition as context unit (type "string"). The algorithm backtracks and decomposes $G - S_{max}$ following the same principle: first, the core property `prissma:environment` is chosen as cut edge. S_{max} is decomposed to S_{max} and $G - S_{max}$, that are both context units (type "entity" and "geo", respectively), and thus added to the decomposition. Note that, once a "geo" context unit is detected, its structure is preserved. Backtracking, the algorithm adds to the decomposition $G - S_{max}$, (`:ctxA`, turned into a `prissma:Context` "class" type context unit). Finally, the element 4 and 9 of the decomposition are joined to create the element 10, that represents the original Prism context definition.

Figure 4.8 shows the decomposition of `:PrismA` and `:PrismB` (see Figure 4.6b for triples). The *decompose*() functions updates the current decomposition by searching for the largest common S_{max} in the decomposition. The algorithm finds that element 6 is isomorphic with a subgraph in the current input Prism, thus only $G - S_{max}$ needs to be decomposed, following the same procedure described above (note that `<http://jack.example.org>` is already present in the decomposition. After the execution of *decompose*() on the two Prisms, the decomposition is made of a number of context units, i.e. "entity" context units (4), "class" (1, 5, 8), "geo" (3, 16), and "string" (0, 11, 12). The remaining decomposition elements include the two Prisms (10 and 18).

[50]The graph isomorphism and the exact subgraph isomorphism operations are delegated to off-the-shelf algorithms, such as [Ullmann 1976, Weber 2012] whose description is out of the scope of this work.

Algorithm 1 decompose-set(P)

Data: a set of Prisms $P = P_1 \dots P_n$
Result: The Prisms Decomposition D

```
1  D = ∅
2  foreach P_i ∈ P do
3      decompose(P_i, D)
4  return D
```

Algorithm 2 decompose(G,D)

Data: a Prism G, the decomposition D
Result: The updated decomposition D

```
1  S_max = ∅
2  if G not context unit then
3      foreach (G_i, G_i', G_i'', E_i) do
4          if G_i is a subgraph of G and S_max smaller than G_i then
5              S_max = G_i
6      if S_max is isomorphic to G then
7          exit
8      if (no subgraph S_max is found) then
9          choose subgraph S_max, priority to Prisma ontology properties
10         decompose(S_max)
11     decompose(G − S_max)
12     add (G, S_max, G − S_max, E) to D
```

4.5.3 Search Algorithm

Every significant context change detected by the device triggers the search for Prisms that fit the updated context requirements. PRISMA carries out this operation with an adapted version of Messmer and Bunke online search algorithm [Messmer 1998]. The algorithm detects *optimal* error-tolerant subgraph isomorphisms between the graph of the sensed context and the Prisms stored in the decomposition. The algorithm first computes edit operations between context units in the decomposition D and context units of the input graph. Second, it combines such edit operations to obtain optimal error-tolerant subgraph isomorphisms for larger patterns, up to complete Prisms (Algorithm 4). To avoid combinatorial explosion, the concatenation of error-tolerant subgraph isomorphisms includes only the cheapest error-tolerant graph isomorphisms: this guarantees to find optimal error-tolerant subgraph isomorphisms.

Algorithm 3 presents the *search* procedure: first, it finds the error-tolerant subgraph isomorphisms from each context unit S of the decomposition D to the input context graph G_I and stores them in the list $candidates(S)$. This operation is performed by the *context_unit_matching* function. From line 3 to 12 such error-tolerant subgraph isomorphisms are concatenated to find error-tolerant subgraph isomorphisms for larger graphs, up to Prisms: in line 3 we select the subgraph S_1 whose error-tolerant subgraph isomorphism f_1 has the minimum cost in D. Note that $C(f_1)$ must be lower than a threshold $T \in [0, 1]$. The error-tolerant subgraph isomorphism f_1 is removed from $candidates(S_1)$ in line 5 and added to the list

Algorithm 3 *search*(G_I, D)

Data: a Decomposition D, a context graph G_I
Result: the result set R containing selectable Prisms

```
1  foreach S context unit in D do
2      candidates(S) = context_unit_matching(S, G_I)
3  while choose S_1 | ∃ f_1 ∈ candidates(S_1) with C(f_1) minimal in D and C(f_1) ≤ T do
4      winners(S_1) = winners(S_1) ∪ {f_1}
5      candidates(S_1) = candidates(S_1) − {f_1}
6      if S_1 is a Prism then
7          R = R ∪ {S_1}
8      foreach (S, S_1, S_2, E) ∈ D || (S, S_2, S_1, E) ∈ D do
9          foreach f_2 ∈ winners(S_2) do
10             f = combine(S_1, S_2, E, f_1, f_2)
11             if f ≠ ∅ then
12                 candidates(S) = candidates(S) ∪ {f}
13 return R
```

$winners(S_1)$, the container of error-tolerant subgraph isomorphism chosen to be combined. If S_1 is a Prism, the algorithm has found a result (lines 6-7). Otherwise, we generate error-tolerant subgraph isomorphisms for each subgraph S having S_1 as ancestor (lines 8-12). Such generation is done with the *combine* function that concatenates f_1 to each $f_2 \in winners(S_2)$, where S_2 is the other ancestor of S. If a combination is feasible, the resulting error-tolerant subgraph isomorphism is added to $candidates(S)$ (line 12).

Algorithm 4 details the *combine* procedure: first (line 1), the function tests if f_1 and f_2 do not contain mappings to the same node in G_I (this is necessary because subgraph isomorphisms are injective functions [Messmer 1998]). If this condition is satisfied, an error-tolerant subgraph isomorphism is constructed as a concatenation of the edit operations of f_1 and f_2 and of the edit operations on the edge between S_1 and S_2, Δ_E (line 8). Mappings are chosen among the mappings of f_1 and f_2 (lines 3-7).

We now discuss in further detail *context_unit_matching*, the function used by the search algorithm to compute error-tolerant subgraph isomorphisms for context units (Algorithm 5). Given a context unit U and an input context graph G_I, the procedure finds the edit operations from U to each context unit of G_I (line 2-3) and stores them as error-tolerant subgraph isomorphisms. Moreover, the deletion of U is considered (line 5).

Example. Consider the decomposition of `:PrismA` in Figure 4.7. The *search*() algorithm is given the actual context G_I in Figure 4.6b as input. Search threshold is set to $T = 0.5$. The search process is explained in Figure 4.9. The uppermost part of the picture shows the numbered context units included in G_I. At step 1, the algorithm computes subgraph isomorphisms from each input context units to every decomposition context unit, i.e. it builds *candidates* vectors C. In the figure, only the least expensive subgraph isomorphisms are shown (the remaining isomorphisms are represented by "..."). For instance, context unit 0 has a subgraph isomorphism containing a substitution operation with input context unit $I3$ that costs 0.45 (this is

Algorithm 4 *combine*$(S_1, S_2, E, G_I, f_1, f_2)$

Data: $S_1, S_2, E, G_I, f_1 = (\Delta_1, f_{\Delta_1}), f_2 = (\Delta_2, f_{\Delta_2}), \Delta_1 = (\Delta_{V_1}, \Delta_{E_1}), \Delta_2 = (\Delta_{V_2}, \Delta_{E_2})$
Result: f
1 **if** $f_{\Delta_1}(\Delta_{V_1}) \cap f_{\Delta_2}(\Delta_{V_2}) \neq \emptyset$ **then**
2 exit
3 **foreach** $v \in (\Delta_{V_1} \cup \Delta_{V_2})$ **do**
4 **if** $v \in V_{\Delta_1}$ **then**
5 $f_\Delta(v) = f_{\Delta_1}(v)$
6 **else if** $v \in V_{\Delta_2}$ **then**
7 $f_\Delta(v) = f_{\Delta_2}(v)$
8 $\Delta = \Delta_1 + \Delta_2 + \Delta_E$
9 return $f = (\Delta, f_\Delta)$

Algorithm 5 *context_unit_matching*(U, G_I)

Data: context unit U, input context graph $G_I = (V_I, E_I)$
Result: the list of error-tolerant subgraph isomorphisms F
1 $F = \emptyset$
2 **foreach** *context unit* $U_I \in G_I$ **do**
3 generate an error-tolerant subgraph isomorphism f between U and U_I
4 $F = F \cup \{f\}$
5 $f' = (\Delta', f'_\Delta)$ with $\Delta' = (v \rightarrow \varepsilon)$ and $f'_\Delta = \emptyset$
6 $F = F \cup \{f'\}$
7 return F

the result of the similarity function between the string "computer" and "computer programming"). The geographical context unit 3 has a substitution cost of 0.21 with input context unit $I7$. Context units 1, 4, 5, and 8 have least expensive subgraph isomorphisms of cost 0 (a perfect match with some input context unit has been found). The algorithm chooses the decomposition item with the lowest error-tolerant subgraph isomorphism. At step 2, this item is 8 (`prissma:Context`). The subgraph isomoprhism is moved to the *winners* vector W. Since at this stage no descendant of item 8 has elements in *winners*, the algorithm repeats the same steps for elements with ascending minimum costs. Step 2 shows that the least expensive subgraph isomorphisms for items 1, 4, and 5 are moved to W. When working on item 4, the algorithm detects that its descendant 6 whose other ancestor (item 5) has values in W. A subgraph isomorphism for 6 is then built, as the concatenations of subgraph isomorphisms of 4, 5, and their combination with the *combine*() function. The resulting subgraph isomorphism is stored in C, with cost=0 (cost is the average of the concatenated isomorphisms costs). At step 3, the W vector of the geo context unit 3 is filled. Since the descendant 7 has both ancestors with elements in their Ws, a subgraph isomorphism is created as the concatenation of ancestor W items, and stored in C, with cost 0.042. At step 4, the least expensive operation for element 0 is moved to W, and it is concatenated into the subgraph isomorphism of element 2, that has average cost 0.15. In step 5, the least cost of 2 is still below T, so W is filled up. The other ancestor of descendant 10 has a value in W, therefore a subgraph isomorphism for 10 is computed. At step 6 this subgraph isomorphism cost is compared to the threshold T. The cost is below T, thus, given that 10

represents the complete context `:ctxA`, the input context matches to `:PrismA`.

4.5.4 Computational Complexity

To analyse the computational complexity of the search algorithm, we report what described in the original Messmer and Bunke version [Messmer 1998]. First, the following definitions need to be introduced:

L = the number of Prisms in the decomposition
m = the number of context units in the incoming context graph
$n_{S_{max}}$ = the number of context units in a Prism common subgraph
$n_{(G-S_{max})}$ = the number of context units in a Prism unique subgraph
n = the number of context units in a Prism

The search algorithm runs on a decomposition built with the algorithm described in Section 4.5.2. For a common subgraph containing $n_{S_{max}}$ context units, and a unique graph of $n_{(G-S_{max})}$ context units, the decomposition contains $O(n_{S_{max}} + Ln_{(G-S_{max})})$ context units. To compute worst case computational complexity[51], we consider the case in which Prisms in the decomposition do not collide. We first consider a decomposition including one Prism only. Three components must be taken into account: First, the search algorithm finds all the possible error-tolerant subgraph isomorphisms for each decomposition element. Finding the error-tolerant subgraph isomorphism is an $O(m^n)$ operation, where n is the number of context units of a Prism. Second, since there are $O(n)$ Prisms in the decomposition, the total number of error-tolerant subgraph isomorphisms found by the algorithm is bounded by $O(nm^n)$. Finally, for each error-tolerant subgraph isomorphism, $O(n)$ context units and edges must be tested. Thus in the worst case, the steps needed to process a single Prism have exponential complexity in n and are bounded by $O(m^n n^2)$. When the decomposition contains L Prisms, three components must be considered: first, the error-tolerant subgraph isomorphisms for the common subgraph S_{max} must be found in $O(m^{n_{S_{max}}} n_{S_{max}}(n_{S_{max}} + L(G - S_{max})))$ steps. Second, the subgraph isomorphisms for the unique complementary subgraph $(G - S_{max})$ must be found in $O(Lm^{n_{(G-S_{max})}} n_{(G-S_{max})} n)$ steps. The third step requires to consider the combination of the subgraph isomorphisms previously found, operation that requires $O(Lm^n n_{S_{max}} n_{(G-S_{max})})$ steps. Putting all together, the worst case computational complexity of the search algorithm is $O(m^{n_{S_{max}}} n^2_{S_{max}} + Lmn_{(G-S_{max})} n_{(G-S_{max})} n + Lm^n n_{(S_{max})} n_{(G-S_{max})})$. In the extreme case where Prisms do not share common subgraphs ($n_{(S_{max})} = \emptyset$, $n_{(G-S_{max})} = n$), the computational complexity becomes $O(Lm^n n^2)$. Another extreme case is where all the Prisms are highly similar, i.e. $n_{(S_{max})} = n$, $n_{(G-S_{max})} = \emptyset$ and the complexity becomes $O(m^n n^2)$. Hence, in this

[51] Messmer and Bunke provides also a detailed best case computational complexity analysis [Messmer 1998]

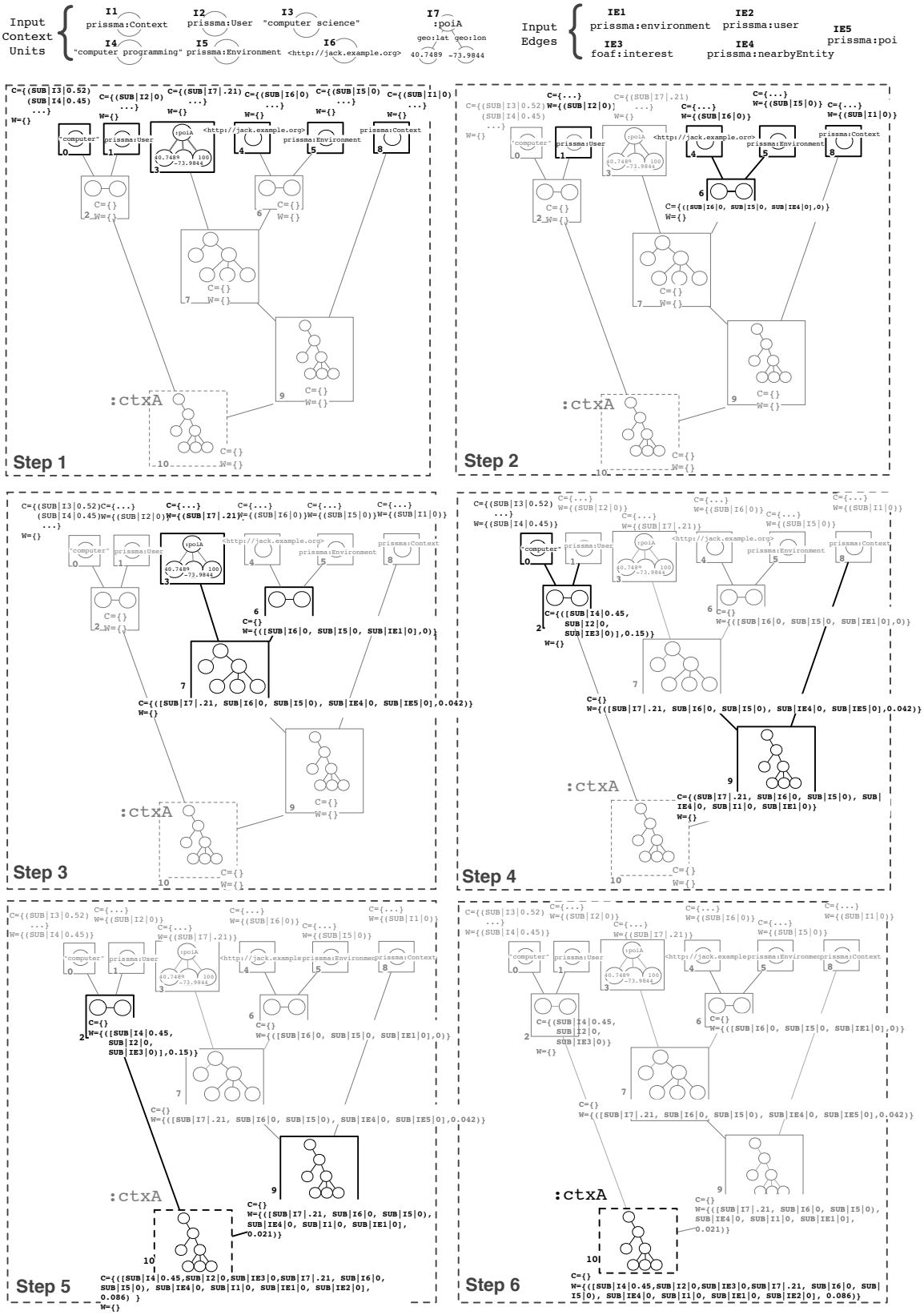


Figure 4.9: Search algorithm example

case the search algorithm is independent from the number L of Prisms included in the decomposition.

4.5.5 Cost of Edit Operations

Each graph edit operation δ computed by the Prism selection algorithm is associated to a cost $C(\delta) \in [0, 1]$. Although in their original proposition Messmer and Bunke do not specify how to compute it, such cost is customizable to our mobile context scenario, where we match incomplete, imprecise, and heterogeneous context declarations to input context graphs. Such prerequisites influence the cost computation for operations on both graph properties and context units.

For what concerns topology, the Prism selection algorithm assigns the highest cost $C(\delta) = 1$ to the substitution of core PRISSMA vocabulary properties (such as `prissma:environment`), so that whenever in the input context graph a core property is missing or does not correspond with the compared property in the decomposition, the cost of the resulting error-tolerant subgraph isomorphism will be higher than the threshold T. Although the algorithm adopts strict matching for core properties, it assigns lower costs for edit operations on non-core properties (e.g. a missing `foaf:interest` property may not prevent a Prism match). The cost $C_{missing}$ of a missing entity is a fixed parameter chosen beforehand that varies between 0 and 1.

Beside property substitution, deletion, and addition, we need to deal with the cost of edit operations on context units. Unlike Messmer and Bunke that only consider topological differences and limit to graphs with discrete node values, in our scenario cost functions are influenced by the imprecision of context data and the support for heterogeneous context dimensions. The substitution of a context unit needs the comparison of two items according to their contextual dimension but also to data imprecision. We list the error-tolerant techniques adopted by PRISMA to compute the cost of the substitution of a context unit, according to each supported contextual dimension:

Location. A "Geo" context unit is a subgraph composed by `geo:lat`, `geo:lng` and a `prissma:radius` (Figure 4.8, context unit 3). The cost of the substitution of a location context unit depends on the geographic distance between the compared context units (assuming complete pattern topologies). The comparison of two geographical context units includes two steps: we first compute the distance d of the two points using the Haversine formula. The Haversine equation computes the distance of two points on a sphere, given their latitudes lat_1 and lat_2, longitudes lng_1 and lng_2, and the Earth radius r:

$$d = 2r \arcsin(\sqrt{\sin^2(\frac{lat_2 - lat_1}{2}) + \cos(lat_1)\cos(lat_2)\sin^2(\frac{lng_2 - lng1}{2})}$$

If d is within the declared `prissma:radius`, the edit operation has cost $C(\delta) = 0$. Otherwise, PRISSMA features an exponential decay function to smooth the

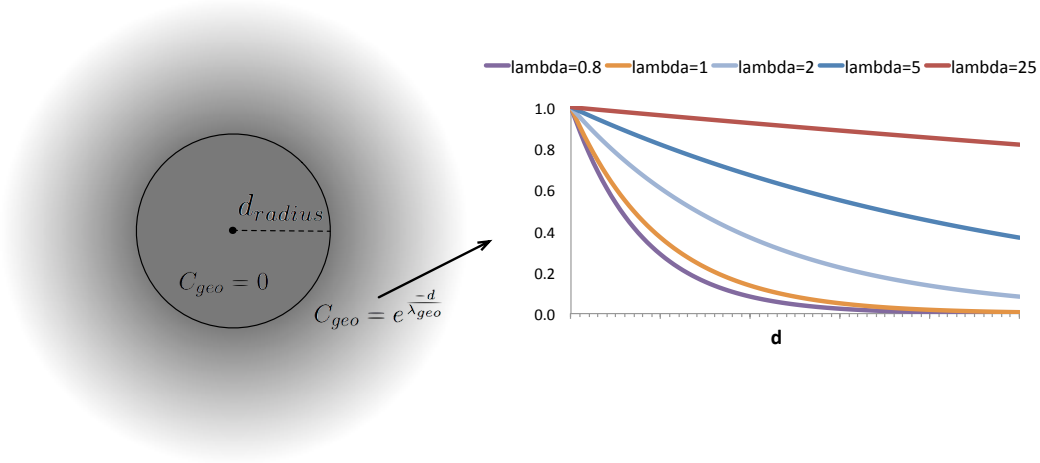


Figure 4.10: The exponential decay function used to compute the substitution of a geo context unit.

transition between a perfect match and a mismatch. This is done to enable the selection of Prisms when users are located not far from the radius boundaries. The exponential decay is characterized by the λ_{geo} parameter, that determines how fast the cost C of the edit operation must rise. More precisely, the cost C_{geo} of substituting a location context unit is defined by:

$$C_{geo}(d) = \begin{cases} 0 & \text{if } d < d_{radius} \\ e^{\frac{-d}{\lambda_{geo}}} & \text{if } d > d_{radius} \end{cases} \tag{4.1}$$

More refined geospatial matching techniques can be used by PRISSMA, but this is left as future work and is therefore out of the scope of this work[52].

Time. Temporal context units include a start timestamp t_{start} and a duration Δ_t (Figure 4.4). The cost of the substitution of a temporal pattern is computed similarly to the location case. If the incoming context timestamp t is contained in the time interval defined by t_{start} and its duration Δ_t, the edit operation cost is set to zero. Otherwise, cost is computed according to an exponential decay function, useful to smooth the transition between a perfect and a lenient temporal match. More precisely, the cost C_{time} of substituting a time context unit is defined by: (Figure 4.11)

$$C_{time}(t) = \begin{cases} e^{\frac{t-t_{start}}{\lambda_{time}}} & \text{if } t < t_{start} \\ 0 & \text{if } t_{start} < t < t_{start} + \Delta_t \\ e^{\frac{-t+t_{start}+\Delta_t}{\lambda_{time}}} & \text{if } t > t_{start} + \Delta_t \end{cases} \tag{4.2}$$

[52] http://linkedgeodata.org/, http://www.w3.org/community/geosemweb/

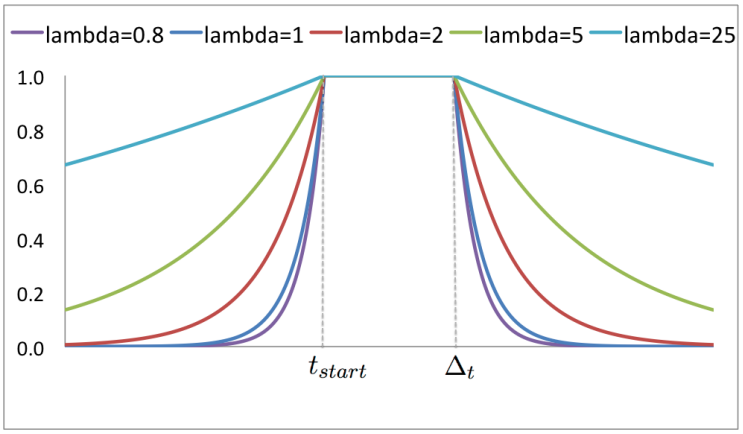


Figure 4.11: The function used to compute the substitution of a time context unit

Strings. The cost C_{string} of substituting a string literal is computed with an approximate string matching strategy, to overcome problems such as spelling variants (to date, the Prism selection algorithm focus only on this string similarity problem). Surveys such as Cohen et al. [Cohen 2003] show that the Monge-Elkan distance function outperforms other approaches when dealing with spelling variants. Hence, the Prism selection algorithm adopts Monge-Elkan string distance measure.

4.5.5.1 Algorithm Validation

The main drawback of edit distance-based subgraph matching algorithms is the need for a proper tuning campaign for cost functions parameters, as acknowledged by Messmer and Bunke [Messmer 1998]. Such parameters influence the behaviour of cost functions, thus impacting on overall matching results. What follows is the description of the parametrization procedure used to validate the Prism selection algorithm. Such task strongly depends on the notion of *context similarity*, i.e. the fact that two contexts are considered to match. Context matching usually depends on subjective decisions, on the nature of context data, and on the usage scenario. Consider the location context dimension: which is the acceptable "error-tolerance"? 50 metres, 100 meters, 1 Kilometre? The answer clearly depends on subjective human factors. The analysis presented in this section does not mean to replace a thorough and extensive campaign evaluation to assess the algorithm performance on a wider scale, for example involving PRISSMA-enabled applications users in the loop.

Four precision-recall analysis has been carried out to assess the validity of the Prism selection algorithm with different cost functions parameters, and with different similarity thresholds T. The first analysis assesses the role of $C_{missing}$, and tests the algorithm on matching contexts with missing context units. The second analysis evaluates matching results with different values of λ_{geo}. The third analysis deals with λ_{time}. The fourth precision/recall analysis compares different approxi-

mate matching string functions. Each test uses a decomposition including one Prism only. The prism is matched against groups of 20 client context graphs, half of which are supposed to match the Prism, even though they are not identical.The other half includes negative matches.

For the $C_{missing}$ evaluation, the `prissma:Context` in the Prism contains a `prissma:User` and a `prissma:Environment` dimensions. The user has five distinct `foaf:interest` properties (string literals). The environment has four distinct `prissma:nearbyEntity` properties, with entity-type context units (hence, test Prism contains 12 context units in total). Positive matches consist in copies of the decomposed Prism having at most four missing context units (either `foaf:interest` or `prissma:nearbyEntity` values). That accounts for 1/3 of the Prism context units. Figure 4.12 shows that, if threshold T is at least as high as the cost assigned to a missing unit cost, precision reaches 0.6 and the algorithm does not miss any true positive (e.g. matching input context). No match is successful if the cost of a missing entity is below T. This is because the search algorithm discards candidate subgraph isomorphisms whose cost is less than T, including subgraph isomorphisms on single context units that in the test have a deletion operation as candidate with lowest cost.

To choose the most appropriate value for the location decay constant λ_{geo}, a similar campaign has been carried out. The `prissma:Context` in the decomposed Prism contains only a `prissma:Environment` dimension. The environment includes a POI, modelled with a "Geo" context unit (latitude, longitude, and radius). Positive matches consist in context graphs with the same topology, but with different location values. Location context units are considered positive match if the distance to the declared radius boundary is at most 10% the radius itself (e.g. `prissma:radius` =1 Km, input context is a positive match up to 1.10 Km, with increasing cost C_{geo}). Not surprisingly, Figure 4.13 shows that steeper exponential decays (higher decay constant) have better precision, but lower recall, as the exponential function decreases faster. For example, with our test configuration, if $T = 0.4$ and $\lambda_{geo} = 5$, we obtain precision $P = 0.83$ and recall $R = 1$.

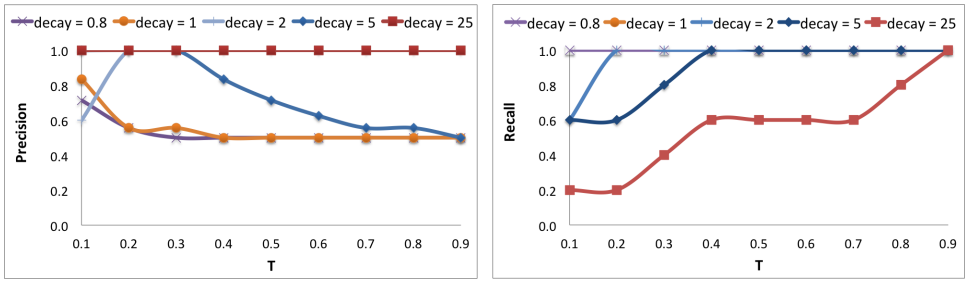
Precision and recall analysis has been carried out to choose the most appropriate value for the temporal decay constant λ_{time}. The `prissma:Context` in the decomposed Prism contains only a `prissma:Environment` dimension. The environment includes a `time:Interval` entity, i.e. a "Time" context unit (`tl:start`= "12:00", and `tl:duration`= 2H). Positive matches consist in context graphs with the same topology, but with different time values. Time context units are considered positive matches if i) are contained in the declared time interval, ii) are at most after 10% of the duration from the duration limit, or iii) occur before t_{start}, but within a 10% tolerance. Figure 4.14 shows the results of the evaluation.

Although a thorough evaluation of these techniques is out of the scope of this work (we rely on the string distance matching techniques survey by Cohen, [Cohen 2003]), in Figure 4.15 a precision-recall analysis shows the performance of four different string similarity measures (Jaro, Jaro-Winkler, Monge-Elkan, and Levenshtein) when dealing with matching string literals with compound words and



(a) (b)

Figure 4.12: Precision and recall of the search algorithm with different values of similarity threshold T, according to different values for the cost of a missing entity context type.

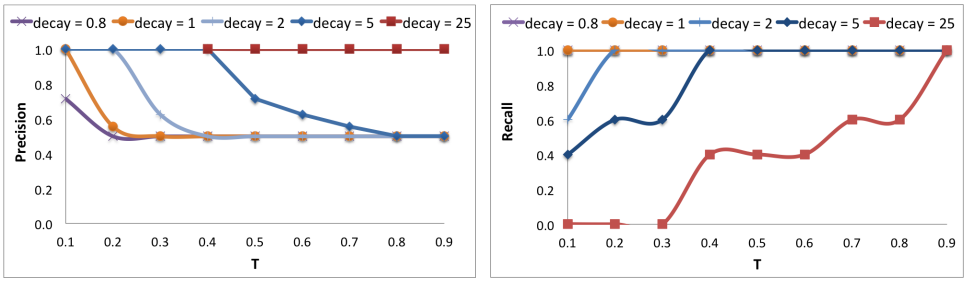


(a) (b)

Figure 4.13: Precision and recall of the search algorithm with different values of similarity threshold T and exponential decay for geographic similarity.

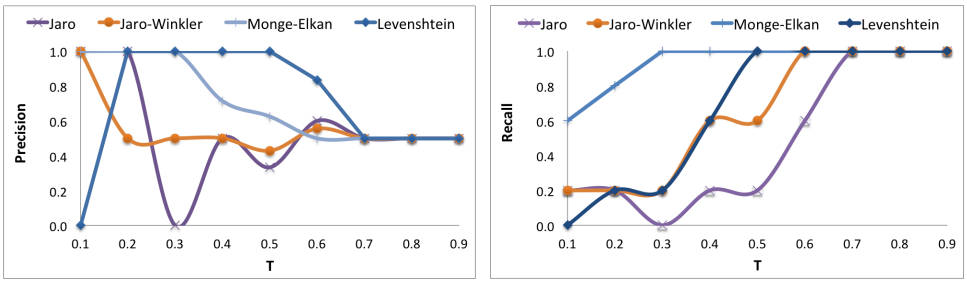
spelling variants. The `prissma:Context` in the decomposed Prism contains only a `prissma:User` dimension. The user includes a `foaf:interest` property with a string literal value "computer programming". Positive matches consist in context graphs with the same topology, but with different `foaf:interest` values. Such values are considered positive matches if the adopted string similarity measure returns a value s such that $T \leq 1-s$. For instance, the following values are considered as positive matches: "programming", "computer programming environments", "computer programming languages", "computer programs", while values such as "programs and software" or "computer software development" are considered negative matches. Results confirm surveys such as [Cohen 2003], that show that the Monge-Elkan distance function outperforms other approaches.

Instead of relying on heuristic parameter tuning, the parametrization of the Prism selection algorithm could be delegated to machine learning techniques. Such future work activity might optimize the choice of the best-fitting similarity threshold and similarity metrics parameters according to context, and relieve developers from manual tuning.



(a) (b)

Figure 4.14: Precision and recall of the search algorithm with different values of similarity threshold T and exponential decay for time similarity.



(a) (b)

Figure 4.15: Precision and recall of the search algorithm with different values of similarity threshold T and string similarity measures.

Edit Operation δ		**Cost** $C(\delta)$
$v \to v'$	$(v, v'$ "Class")	0 if URIs match, 1 otherwise
	$(v, v'$ "Entity")	0 if URIs match, $C_{topology} \in [0, 1]$ otherwise
	$(v, v'$ "String")	$C_{string} \in [0, 1]$
	$(v, v'$ "Geo")	$C_{geo} \in [0, 1]$
	$(v, v'$ "Time")	$C_{time} \in [0, 1]$
$e \to e'$	(e core property)	0 if URIs match, 1 otherwise
	(e not core)	0 if URIs match, $C_{topology} \in [0, 1]$ otherwise
$v \to \varepsilon$, $e \to \varepsilon$		1 if v is "Class","Geo","Time", or e core property. $C_{topology} \in [0, 1]$ otherwise
$\varepsilon \to e$		0

Table 4.4: Cost value ranges associated to graph edit operations presented in Definition 9. Note that the presence of additional properties between two context units is not considered to have an impact on the global cost, and is therefore assigned cost 0.

4.6 Evaluation

The PRISSMA decomposition and selection algorithms have been implemented as an Android library[53]. First, decomposition memory consumption is assessed. Second, search algorithm response time has been tested.

The first test analyses the decomposition memory consumption (Figure 4.16). The test measured the decomposition size against groups of Prisms with a variable number of identical context units. Groups included 20 Prisms, each containing 10 context units (and 3 additional shared context units shared by all Prisms, `prissma:Context`, `prissma:User`, and `prissma:Environment`) Overall, test Prisms accounted for 340 triples. The percentage of identical context units in each group of Prisms is progressively increased, ranging from 10% to 100% (where the latter means that all Prisms in the group are represented by the same decomposition item). Figure 4.16a shows evolution of the number of decomposition items: as Prisms share larger subgraphs, the overall number of context units in non-processed Prisms remain constant, while the number of decomposition context unit decreases (e.g. if Prisms contain 20% of repeated context units, the number of context units in the test decomposition is 57% lower). The number of total decomposition items (context units plus intermediate units) also decreases, as overlapping subgraphs are represented only once, thus reducing redundancy. In Figure 4.16b we estimated the memory size of the decomposition under the same test configuration. We assigned an arbitrary size of 30 Bytes to context units (we consider UTF-8 strings with an average length of 30 characters), and 42 Bytes to intermediate decomposition elements (one integer ID, two integer ancestors IDs, and a list of connecting edges. Each edge includes a triple of estimated size 90 characters). The size of PRISSMA decompositions are compared with the retained size of a group of Jena Model[54], each containing a test Prism. As expected, with higher common context units percentages, we have lower decomposition memory footprints. Nevertheless, the memory size of PRISSMA decomposition is in the same order of magnitude of the Jena models size.

A series of tests have been run to assess the computational complexity analysis of the search algorithm. The algorithm response time has been tested on a group of Android mobile devices (Google Nexus 4[55], Google Nexus 10[56], Samsung Galaxy Mega[57], and Samsung Galaxy Note[58]). All phones were running Android 4.2.2. Figure 4.17a shows the relationship between L, the number of Prisms in the decomposition, and response time. Prisms in each group are all different (thus testing the worst case decomposition configuration). Prisms contains $n = 10$ context units and the test context to be matched is made of $m = 10$ context units. Five independent

[53] Binaries and code available at `wimmics.inria.fr/projects/prissma`, along with test data used for evaluating the system.

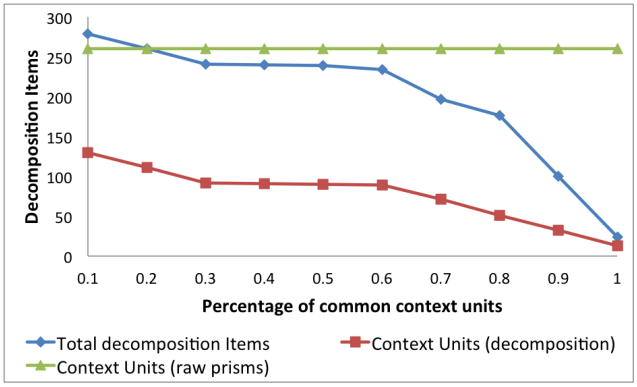
[54] `http://jena.apache.org/documentation/notes/model-factory.html`

[55] `https://play.google.com/store/devices/details?id=nexus_4_8gb`

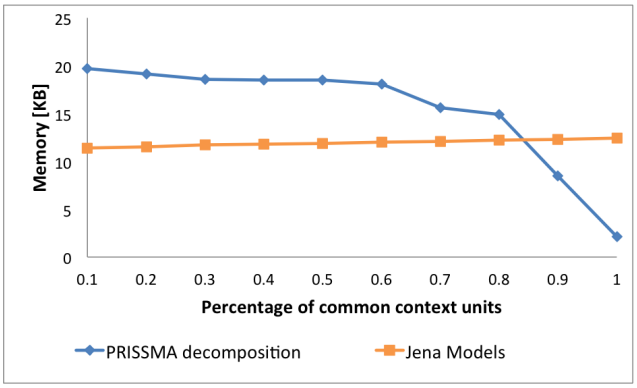
[56] `https://play.google.com/store/devices/details?id=nexus_10_16gb`

[57] `http://www.samsung.com/in/galaxymega/`

[58] `http://www.samsung.com/in/galaxynote/`



(a)



(b)

Figure 4.16: Decomposition memory consumption analysis

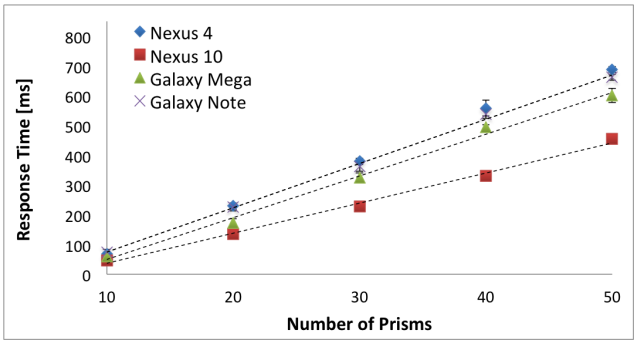
runs have been executed for each group of Prisms, thus computing average response time measurements. Results show a linear dependency, thus confirming the worst case complexity analysis of the search algorithm for what concerns the number of Prisms $O(L)$. The sublinear relationship between the number of Prisms and response time accommodates situations with a fairly large number of Prisms in the decomposition, thus fitting scenarios where a large number of context-dependent representations are needed for the same RDF entity or class. Figure 4.17b shows how the size of the incoming context graph impacts on response time. In this case, each run varied the size m of input context (ranging from 10 to 50 context units) using a fixed group of $L = 5$ Prisms each made of $n = 3$ context units. Results match computational complexity analysis, thus giving a $O(m^n)$ relationship (experimental setup shown in Figure 4.17b has $n = 3$, thus giving a $O(m^3)$ relationship). Unlike the number of Prisms, the polynomial growth associated to the size of the incoming context graph suggests that the size of the latter must be kept as small as possible, to consistently reduce response time. Finally, in Figure 4.17c the size n of each Prism has been tested. Five independent test runs assessed response time using an incoming context graph of $m = 50$ context units and a decomposition made of $L = 5$ Prisms. Results confirm the complexity analysis $O(n^2)$. As for the case of incoming context graph, the size of Prisms impacts with a quadratic growth on response time, thus it is important to avoid defining useless context conditions in Prisms to lower response time.

The test in Figure 4.18 shows the delay of the search algorithm combined with the time used by the Fresnel engine to render the desired resource. The results shown in the graph have been generated with the same test configuration of Figure 4.17a, with the inclusion of Fresnel rendering delay. The test, executed on different Android devices, consisted in choosing the Prism that better matches the incoming context, and using its Fresnel directives to render a `foaf:Person` in an HTML representation with Fresnel. The adopted Fresnel engine is the JFresnel Java library[59]. As seen in Figure 4.18, when the decomposition contains $L = 10$ Prisms, the major impact on response time is determined by Fresnel ($\sim 60\%$), while the search algorithm accounts only for $\sim 40\%$ of time. When the number of Prisms included in the decomposition grows, the predominant component of response time is determined by the search algorithm (e.g. if $L = 50$, the search algorithm accounts for $\sim 75\% - 90\%$ of response time varying from the test device).

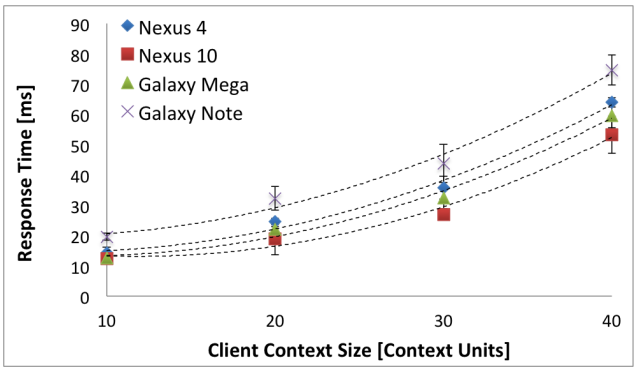
4.7 PRISSMA Browser

The PRISSMA framework has been used in a proof-of-concept mobile application, the PRISSMA Browser. PRISSMA Browser is a mobile Linked Data browser, enhanced with context-aware adaptation. Designed for Android devices, PRISSMA Browser consists in a Web browser enriched with the PRISSMA framework. Thus, the browser is capable of fetching and rendering RDF resources according to the

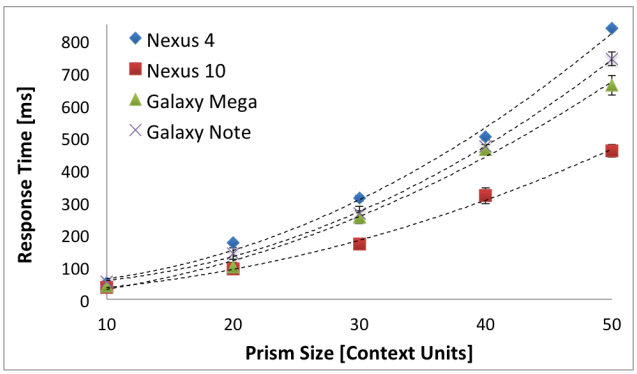
[59] http://jfresnel.gforge.inria.fr



(a)



(b)



(c)

Figure 4.17: Search algorithm response time evaluation. (a) shows the linear dependency on L, the number of Prisms in the decomposition, when all Prisms in the decomposition are different (worst case). Each Prism contains $m = 10$ context units, and the incoming context includes $n = 10$ context units. (b) shows the $O(m^n)$ dependency on n, when $n = 3$ and $L = 5$. (c) shows how the size n of each Prism impacts on response time when input context graph contains $m = 50$ context units and $L = 5$ ($O(n^2)$).

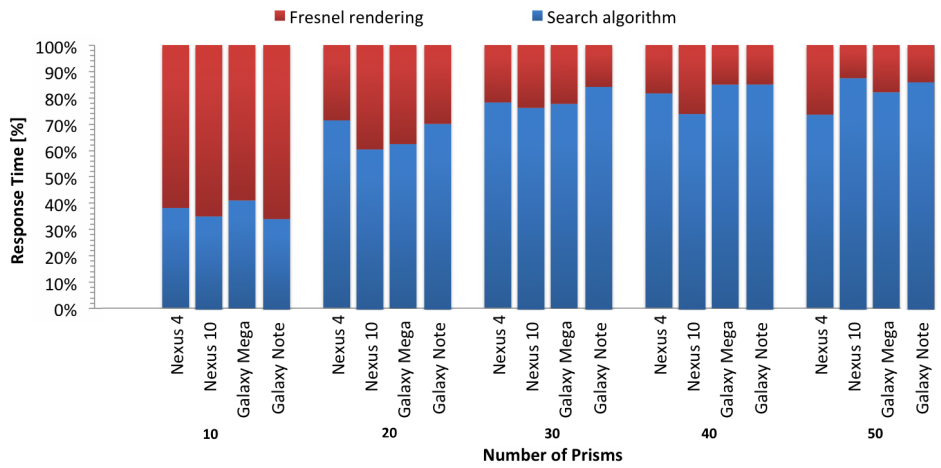


Figure 4.18: Combined delay of search algorithm and Fresnel rendering. The test has been run with a variable number of Prisms L in the decomposition, $m = 10$ and $n = 10$. The adopted Fresnel engine is JFresnel.

sensed context. PRISSMA Browser shows the interplay of PRISSMA and Fresnel, along with the Prism selection algorithm.

PRISSMA browser is aware of the current mobile context. A PRISSMA context graph is created at application startup. Such *prissma:Context* is updated every time context changes significantly, to reduce battery drain (e.g. by relying on the Android *LocationListener.onLocationChanged()* method). The browser contains a user-customizable set of Prisms, associated to a number of RDF classes or resources. Prisms are decomposed with the PRISSMA decomposition algorithm and stored in memory. When an RDF resource is requested, the browser performs an HTTP request and fetches the requested data. As soon as triples are received from the Web, current client context is used as input for the PRISSMA selection algorithm over the Prisms decomposition. Once the most suitable Prism is found, the associated Fresnel lenses and formats are extracted and used to render the desired resource (PRISSMA Browser only provides HTML representation of RDF, even though Fresnel supports multiple output paradigms such as PDF or Audio representation).

PRISSMA Browser is implemented on top of the Webkit-based, open source Lighting Browser for Android[60]. Figure 4.19 shows screenshots of the prototype (the application is running on a Nexus 4 smartphone). In the example the same `foaf:Person` resource has been requested, but different Prisms have been selected, according to device features, user profile and context information. Figure 4.19a shows the HTML page produced by a Prism that shows only name, nickname, and depiction. Figure 4.19b displays the output of a Prism that creates a view on the foaf:Person more focused on professional activities. The Prism is selected, for example, when the client request is made from a working place. In Figure 4.19c

[60] https://github.com/anthonycr/Lightning-Browser

we find the same information, but a "night mode" styling is applied, a visualization selected at night. Figure 4.19d shows the option panel of the browser. Options include the context dimensions to use when building the actual context graph. The panel also allows to edit the Prism directory path, and lets the user specify his FOAF profile URI, so that it will be used to build the `prissma:User` context dimension.

The example in Figure 4.20 shows two context-aware HTML representations of the DBpedia RDF entity that identifies the Louvre Museum[61]. The content in Figure 4.20a contains a textual description of the museum (`rdfs:comment` value), the city, the director, the curator name, and the establishment year. The layout in Figure 4.20a is optimized for the Nexus 10 tablet. On the other hand, Figure 4.20b provides a visualization of the same Louvre RDF entity optimized for people in Paris, that are currently walking, and using a smartphone with a 1280 x 768 resolution. The idea is that in such context (walking in Paris with a smartphone) users will benefit more from practical informations such as the museum address (`dbpprop:location`[62]), metro station (`dbpprop:publictransit`), and other information such as opening hours and ticket fees. Note how address and public transportation information are highlighted, since considered more important in current context.

4.8 Conclusions

The PRISSMA context-aware presentation engine answers question of context-aware adaptation for Linked Data by addressing both the problem of context modelling at presentation level, and the selection of the most proper context description at runtime. Fresnel has been enhanced with full-fledged context-awareness. Relying on Fresnel favours the sharing and reuse of Prisms across applications, and does not introduce new formalisms other than RDF. The problem of selecting the most pertinent context-based representation has been solved by reducing the error-tolerant subgraph isomorphism problem to the computation of graph edit distance. This has been done with the adoption of Messmer and Bunke optimal subgraph isomorphism algorithm. The algorithm has been adapted to RDF graphs that represent context information. Hence, it has been modified in several ways: the algorithm is now able to compare complex unit of content made of more than one node, the "context units" (e.g. location, or time information cannot be expressed by a single triple). The general support for heterogeneous dimensions has been tailored to context-attributes, thus adding a series of similarity measures in the computation of edit operations between items. For instance, location units are compared by computing the Haversine distance and applying an exponential decay on the result to account for location imprecision. Time similarity has been introduced. String units are compared with similarity measures (e.g. Monge-Elkan). The behaviour of such similarity functions for edit operations has been tested with different parameters,

[61] `http://dbpedia.org/resource/The_Louvre`
[62] the `dbprob:` prefix corresponds to `http://dbpedia.org/property/`.

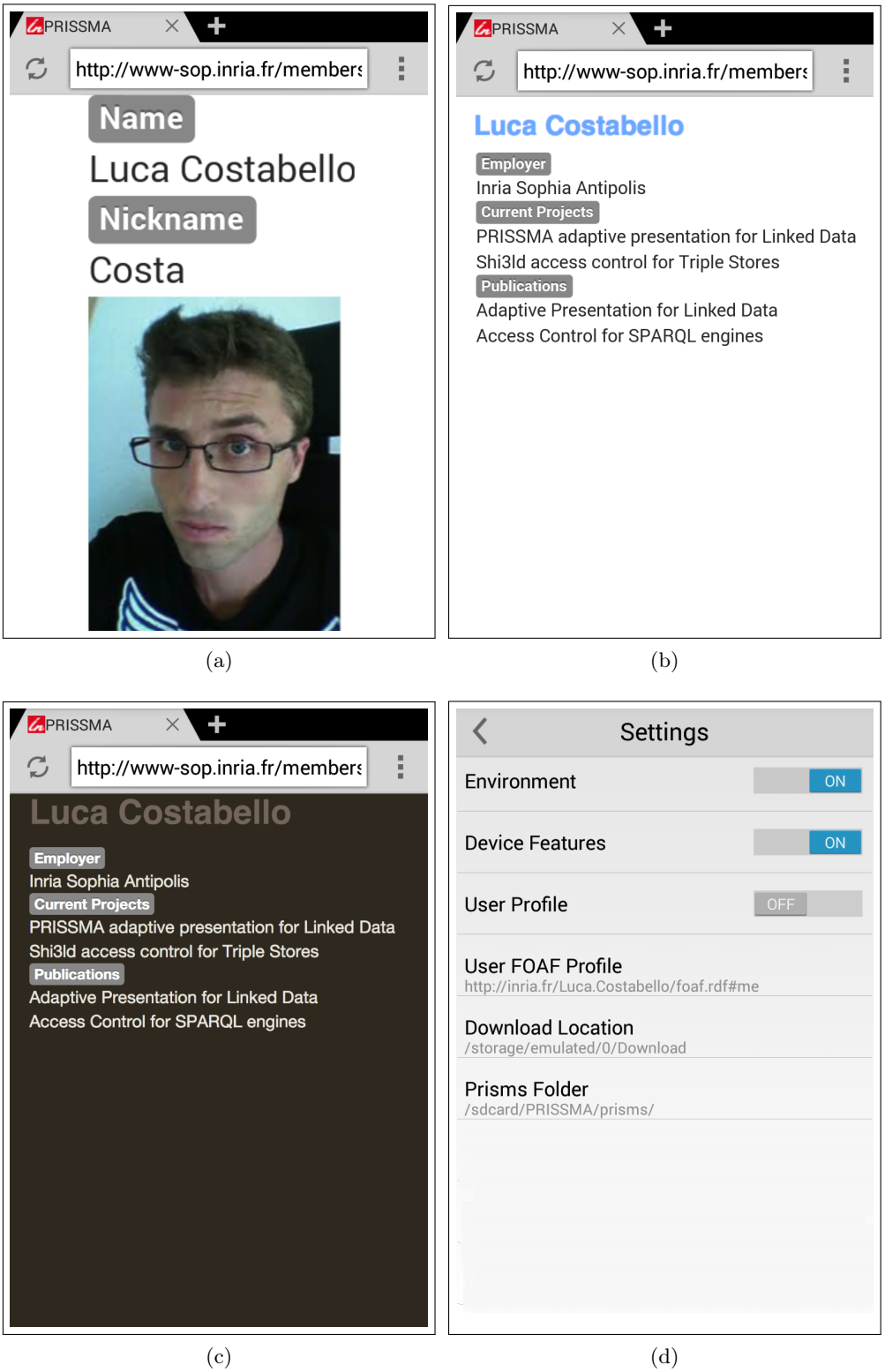
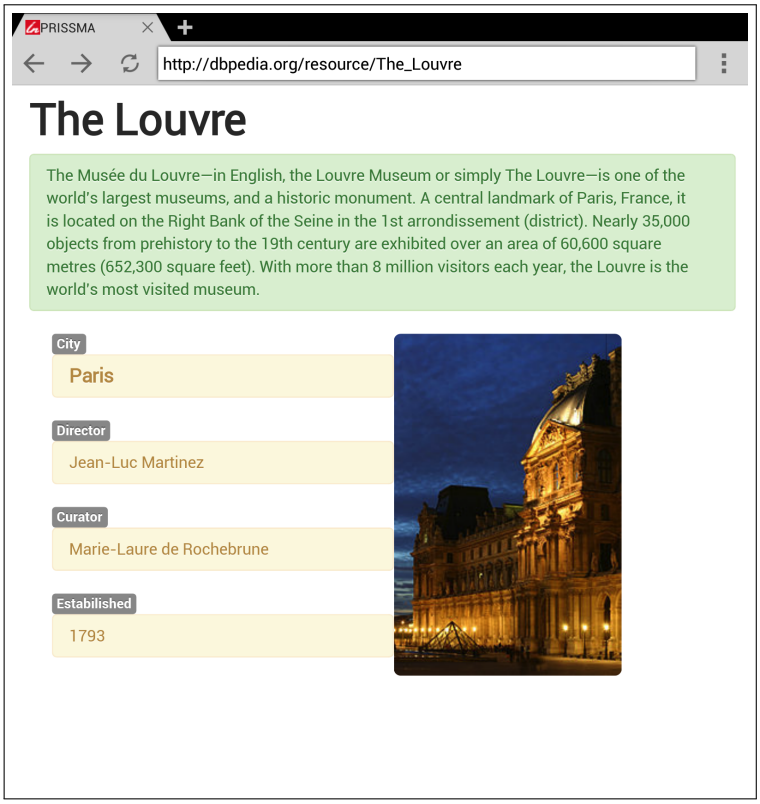


Figure 4.19: Screenshots from the PRISSMA Browser Android application. (a) Shows the result of a Prism that selects name, nickname, and depiction. (b) shows a professional profile, selected from a working place. (c) displays the same content, but it is formatted by a night-mode CSS. (d) shows the option panel.



(a)



(b)

Figure 4.20: Screenshots from the PRISSMA Browser Android application. The images show the HTML rendering the Louvre Museum (a) on a Nexus 4 smartphone when the user is walking in Paris, (b) on a Nexus 10 tablet when the user is at home.

thus showing precision/recall trade-off. Memory consumption tests show that the decomposition structure helps reducing memory usage when repeated subgraphs are present, as in the case of a set of Prisms. The response time test campaign confirms the theoretical complexity analysis of the selection algorithm, and shows the sublinear dependence on the number of Prisms in the system. Operating on the client side guarantees privacy preservation, because context data does not have to be disclosed to third-party adaptation servers. PRISSMA has been implemented as an Android library. A proof-of-concept adaptive Linked Data browser has been developed to show PRISSMA features: PRISSMA Browser is equipped with the PRISSMA library, and is therefore capable of adapting RDF instances according to the sensed context.

If on one hand, a graph edit distance-based algorithm offers a flexible and customizable solution for dealing with heterogeneous context dimensions, on the other hand such method must undertake proper parametrization of the adopted cost functions (i.e. similarity metrics parameters tuning), and choosing the most appropriate threshold for determining when graphs model the same context. This well-known issue of strategies based on graph edit distance is the current main limitation of PRISSMA.

Future work will deal with a series of activities: to date, the issue of Prisms *distribution* has not been examined. A future evolution of PRISSMA might support multiple strategies for discovery, retrieve, and consume Prisms. For instance, given that Prisms are RDF declarations, it could be possible to distribute them on the Web of Data, according to Linked Data principles, thus creating *Linked Presentation-level Metadata*. Triples stores might store Prisms with the associated entities. Otherwise, ad-hoc presentation-level repositories might be added to the Linked Data cloud. PRISSMA-powered applications might discover such context-based presentation metadata at run time, thus giving birth to applications that visualize RDF entities with "filters" created by third parties and associated to given contexts (e.g. to given locations only). Prisms are made of RDF triples, therefore, as a preliminary step before the search algorithm, they might be enriched with additional triples fetched from the Linked Data cloud, to obtain more complete context information (e.g. by dereferencing the entities included in each Prisms by n-hops). Such Prism "expansion" with Linked Data will improve the precision of the search algorithm. Future work will also deal with enhancing the selection algorithm with other cost functions, such as semantic distance between URIs. Response time comparison with cited state-of-the-art solutions is envisaged, but such task would need to adapt these frameworks to a mobile scenario, since none of the related works is designed to run on mobile devices. Moreover, although some of these works provide a response time evaluation, experimental conditions vary, making the comparison difficult. User acceptability evaluation needs to be performed with proof-of-concept applications, such as the PRISSMA Browser (Section 4.7).

Chapter 5

Context-Aware Authorization for Graph Stores

Contents

5.1 Introduction . . . 79
5.2 Access Control Frameworks for Linked Data . . . 81
5.2.1 Access Control for SPARQL . . . 82
5.2.2 Access Control for HTTP Access to Linked Data . . . 84
5.2.3 Context-Aware Access Control . . . 85
5.3 A Model for Context-Aware Access Control Policies . . . 87
5.4 Shi3ld-SPARQL . . . 93
5.4.1 Access Control Enforcement . . . 93
5.4.2 Context Attribute Privacy . . . 96
5.4.3 Evaluation . . . 98
5.5 Shi3ld-HTTP . . . 100
5.5.1 Shi3ld for SPARQL Graph Store Protocol . . . 101
5.5.2 Shi3ld-LDP with Internal SPARQL Engine . . . 101
5.5.3 SPARQL-less Shi3ld-LDP . . . 103
5.5.4 Evaluation . . . 105
5.6 Policy Manager . . . 108
5.6.1 Policy Manager Features . . . 108
5.7 Conclusions . . . 110

5.1 Introduction

Denying or allowing access to a set of resources or services is a common problem in a large number of computing fields. In ubiquitous computing, different research areas deal with access control, ranging from location-based services to personal area networks. Access control has been enriched with location awareness; other contextual dimensions such as the proximity of nearby people or objects have been considered, thus focusing on the data consumption context. The open nature of the current Web of Data may give providers the impression that their content is not safe, and may prevent further publication of datasets, at the expense of the growth of the

Web of Data itself [Heath 2011] . In scenarios such as Linked Enterprise Data, access control becomes crucial, as not all triples must be openly published. Furthermore, as ubiquitous connectivity takes off, mobile users interact with Linked Data servers under novel circumstances, e.g. in crowded areas, on public transportation systems, etc. Hence, context must be part of the access control evaluation, as uncontrolled access in given situations may be undesired by data providers.

This chapter addresses the third research question of the thesis, *how to address access control for Linked Data servers queried by context-aware, mobile devices?* Major challenges arise from such question: first, the definition of a fine-grained model for access control policies must be addressed. Such model must integrate context-aware features, rely on Semantic Web standards, provide fine-grained protection, and support different action privileges (e.g. create, read, update, delete operations). Linked Data clients access the Web of Data by means of SPARQL queries or by directly executing HTTP operations on triples. Thus, a second challenge consists in designing an access control *enforcing mechanism* for both SPARQL endpoints and HTTP operations on triple stores. Finally, dataset administrators must be assisted in creating and managing access policies in a user-friendly manner.

Access control frameworks proposed in literature protect either SPARQL endpoints or generic RDF documents. Frameworks targeting HTTP access to RDF resources rely on access control lists, thus offering limited policy expressiveness, e.g. no location-based authorization [Hollenbach 2009, Hulsebosch 2005, Muhleisen 2010, Sacco 2011b]. On the other hand, existing access control frameworks for SPARQL endpoints [Abel 2007, Flouris 2010] add complexity rooted in the query language and in the SPARQL protocol, thus not fitting the HTTP-only scenario. Furthermore, they often introduce ad-hoc policy languages.

This chapter addresses the problem of access control by presenting an authorization framework called Shi3ld[63]. Shi3ld relies on an RDFS/OWL vocabulary for defining context-based access control policies. The access control enforcement procedure of Shi3ld comes in two flavours: Shi3ld-SPARQL, designed for SPARQL endpoints, and Shi3ld-HTTP, created to protect HTTP operations on triples. Furthermore, Shi3ld includes a policy manager application designed for dataset administrators.

Shi3ld access policies are defined according to two RDF vocabularies, S4AC and PRISSMA. The S4AC ontology models access control concepts and consists in the policy backbone. The PRISSMA vocabulary models context-based conditions. Shi3ld-SPARQL protects RDF stores by changing the semantics of the incoming SPARQL queries, whose scope is restricted to triples included in accessible named graphs only. The list of accessible graphs is determined by evaluating pre-defined access policies against the actual mobile context of the requester. The Shi3ld authorization framework for HTTP derives from the SPARQL scenario and is designed to work in conjunction with the SPARQL 1.1 Graph Store HTTP Protocol, and in configurations where SPARQL is no longer present (e.g. the W3C Linked Data

[63] http://wimmics.inria.fr/projects/shi3ld/

Platform). In this case two solutions have been developed: an authorization module embedding a hidden SPARQL engine, and one where SPARQL has been scraped off. In the latter case, the Shi3ld framework adopts a SPARQL-less subgraph matcher which grants access if client attributes correspond to the declared policy graphs. Both prototypes comply to the Linked Data Platform specifications.

The Shi3ld authorization framework offers a series of benefits: first, it allows context-aware policies, thanks to the adoption of an attribute-based paradigm. Second, Shi3ld adopts exclusively Semantic Web languages, thus no new policy definition languages, parsers nor syntax validation procedures have been defined. Third, Shi3ld supports both SPARQL endpoints and HTTP operations for Linked Data. It consists in a pluggable filter for generic SPARQL endpoints, thus there is no need to modify the endpoint itself. Shi3ld provides protection up to triple level. The response time overhead introduced by the Shi3ld access control procedure is acceptable for most use scenarios. Finally, Shi3ld is compatible and complementary with the WebID authentication framework[64].

For each Shi3ld configuration, response time evaluation is provided, along with the impact of the authorization procedure on HTTP operations on RDF data.

Shi3ld focuses on *authorization* only and does not address issues related to authentication and identity on the Web. Shi3ld does not deal with mobile context fetch, nor with onboard sensors or server-side inference. For the time being, the framework assumes the trustworthiness of the information sent by the mobile consumer, including data describing context (e.g. location, device features, etc). Although this chapter discusses state-of-the-art anti-spoofing techniques for attribute data, the present work does not directly address the issue.

The remainder of this chapter is organized as follows: section 5.2 summarizes the related work, and highlights the requirements of the authorization model for the desired scenario. Section 5.3 describes the access control model adopted by Shi3ld. Section 5.4 introduces Shi3ld for SPARQL and the access control enforcement algorithm, along with response time evaluation. Section 5.5 presents three solutions to adapt the Shi3ld framework to HTTP operations on RDF. The Section includes an experimental evaluation of response time overhead. Section 5.6 describes the graphical user interface used by dataset administrators to create and manage Shi3ld access policies.

5.2 Access Control Frameworks for Linked Data

Access control is a popular topic, and spans different research communities. This Section presents three different categories of works: access control frameworks for SPARQL, access control for HTTP access to linked data, and context-aware access control solutions. The works are reviewed and compared in Table 5.1 according to the following criteria:

[64] http://www.w3.org/2005/Incubator/webid/spec/

Access Control Model. Access control frameworks might be designed according to different models. The most popular models are Role-based Access Control [Sandhu 1996] (RBAC) and Attribute-based Access Control (ABAC). RBAC systems grant or deny access according to a role assigned to the user that is performing the request. Hence, they typically rely on access control lists. ABAC frameworks, on the other hands, protect resources according to a set of attributes provided by the client. The access control enforcement procedure evaluates the attributes and grants access according to pre-defined policies.

Policy Language. Any access control solution needs policies, and therefore a language to express them.

Protection Granularity. Access control frameworks differ in terms of smallest unit of information protected (e.g. RDF documents, named graphs, RDF resources, individual triples, etc).

Permission Model. A number of frameworks support different behaviours according to the type of operation performed on the protected resource. For instance, certain access policies can be enforced for write operations only. Nevertheless, such feature is supported in different ways.

Context Awareness (CA). Context-aware capabilities are an important comparison criteria, that concerns the expressivity of access policies, thus influencing the functionalities of each framework.

Conflict Verification. A given resource might be protected by multiple, perhaps contradictory access policies. Access control frameworks must handle this situation.

Evaluation. Although a large number of access control frameworks have been proposed, only a fraction comes with proper experimental evaluation.

5.2.1 Access Control for SPARQL

A series of works tackle the issue of authorization for SPARQL endpoints, thus granting or denying the execution of SPARQL queries on RDF triples.

One of the first proposed approaches is the work by Abel et al. [Abel 2007]. They provide triple-level access control as a layer on top of SPARQL endpoints. Policies are not expressed using Semantic Web languages, instead, they are expressed using an high-level, ad-hoc syntax and mapped to existing policy languages. Context information is supported to some extent, e.g. policies including time, location, etc (context conditions are pre-evaluated before expanding the queries). They pre-evaluate the contextual conditions, then the queries are expanded, and sent to the database.

	Web Based	AC model	Policy Language	Protection Granularity	Permission Model	CA	Conflict Verif.	Eval.
WAC	•	RBAC	RDF	RDF document	R/W			
Abel et al.[Abel 2007]	•	ABAC	Custom	triples	R	•		•
Finin et al.[Finin 2008]	•	RBAC	OWL/RDF	resources				
RelBAC[Giunchiglia 2009]	•	relation	DL	resources				
Hollenbach[Hollenbach 2009]	•	RBAC	RDF	RDF document	R/W			•
Flouris et al.[Flouris 2010]	•	RBAC	Custom	triples	R		•	•
PeLDS[Muhleisen 2010]	•	RBAC	SWRL	RDF document	R/W			•
PPO[Sacco 2011b]	•	ABAC	RDF, SPARQL	RDF doc(part)	R/W			
SCBAC[Shen 2011]	•	context	SWRL	resources		•	•	
Covington[Covington 2001]		RBAC	Custom	resources	R/W	•	•	
CSAC[Hulsebosch 2005]		gen. RBAC	XML	resources	R	•		
Proteus[Toninelli 2006]		context	DL	Resources		•	•	•
OrBAC[Cuppens 2008]		organization	Datalog	resources	R/W	•	•	
UbiCOSM[Corradi 2004]		context	RDF	resources		•	•	•
MyCampus[Sadeh 2005]	•	context	RDF	Web APIs	R/W	•		
Kirrane et al.[Kirrane 2013]	•	DAC	RDF	up to subjects, objects, properties	R/W		•	•
Shi3ld SPARQL[Costabello 2012a]	•	ABAC	RDF, SPARQL	named graphs	CRUD	•		•
Shi3ld HTTP[Costabello 2013a]	•	ABAC	RDF	RDF resources	CRUD	•		•

RBAC	Role-Based Access Control
ABAC	Attribute-Based Access Control
gen. RBAC	Generalized Role-Based Access Control
RBAC	Role-Based Access Control
DAC	Discretionary Access Control
RDF	Resource Description Framework
SPARQL	SPARQL query language
OWL	Ontology Web Language
SWRL	Semantic Web Rule Language
CRUD	Create, Read, Update, Delete
DL	Description Logic
R/W	read/write
R	read-only

Table 5.1: A comparison of access control frameworks.

Flouris et al. [Flouris 2010] present a fine-grained access control framework on top of RDF repositories. The authors underline the need of a fine-grained access control framework while being repository independent. They do not support context-based access policies. They propose a high level specification language which has to be translated into a SPARQL/SerQL/SQL query to enforce the policy, and they focus on read operations only.

Based on the Oracle Relational database, the Oracle triple store protects RDF granting or revoking operations on database views. If tighter security is requested, triple-level access control can be enforced by relying on Oracle Label Security or Oracle Virtual Private Database[65].

Kirrane et al. present a general authorization framework for interacting with Linked Data using SPARQL 1.1 [Kirrane 2013]. The authors rely on the discretionary access control paradigm (DAC), and adapt such model to RDF graph patterns. Their solution features conflict resolution between policies, propagation rules, and integrity constraints. The described authorization framework must work in conjunction with SPARQL, or with other query languages for RDF. However, the support of HTTP operations on Linked Data is out of the scope of the work. Although the adopted DAC paradigm is potentially capable of handling context information, the authors do not add context-based authorisation features.

5.2.2 Access Control for HTTP Access to Linked Data

A larger number of frameworks deal with protecting access to RDF resources accessed with HTTP operations.

The definition of access control policies for the Web has been firstly addressed by the Web Access Control vocabulary (WAC)[66]. WAC relies on access control lists (ACLs), structures that define which user can access the data. ACLs are created and managed by data providers, and grant access to RDF documents. ACLs comply to the WAC vocabulary: the ontology provides four classes of access control privileges: `Read` (read the content), `Write` (delete or update the content), `Control` (set the ACL for the content), and `Append` (add information at the end of the content). and are of the form `[acl:accessTo <card.rdf>; acl:mode acl:Read, acl:Write; acl:agentClass <groups/fam#group>]`, which means that anyone in the group `<http://example.net/groups/fam#group>` may read and write `card.rdf`.

Hollenbach et al. [Hollenbach 2009] present a system where providers control access to RDF documents using WAC. Their policies expressiveness is limited by the adoption of a model based on access condition lists.

Similarly to ACLs, other approaches specify *who* can access the data, e.g., to which roles access is granted: Giunchiglia et al. [Giunchiglia 2009] propose a Relation Based Access Control model (*RelBAC*), a formal model of permissions based on description logic. They require to specify who can access the data.

[65] http://docs.oracle.com/cd/E11882_01/appdev.112/e11828/fine_grained_acc.htm
[66] http://www.w3.org/wiki/WebAccessControl

Sacco and Passant [Sacco 2011a, Sacco 2011b] present the Privacy Preference Ontology (PPO[67]). The vocabulary allows the creation of fine-grained access control policies to protect RDF documents. Although built on top of WAC, PPO embraces the attribute-based access control model (ABAC). Consumers access a target RDF file, e.g., a FOAF profile. The PPO access control manager returns only the part of the file accessible by the consumer. PPO access conditions are implemented as SPARQL `ASK` queries. The PPO vocabulary does not consider context information and, since it relies on WAC, it does not discriminate between different write actions (e.g. update, delete, create). PPO does not support conjunctive and disjunctive sets of access conditions.

Muhleisen et al. [Muhleisen 2010] present a policy-enabled server for Linked Data called PeLDS, where the access policies are expressed using the PsSF descriptive language, based on SWRL[68]. They distinguish only `Read` and `Update` actions, and they do not consider contextual information. The system is based on an ontology of the actions that can be performed on the datasets, but no further description is provided.

Finin et al. [Finin 2008] study the relationship between OWL and Role Based Access Control (RBAC) [Sandhu 1996]. Instead of relying on access control lists, they discuss possible ways of going beyond RBAC, and, in particular, they consider attribute based access control where access constraints are based on general attributes of an action. They do not specify policies in SPARQL and they do not support geo-temporal access conditions.

Giunchiglia et al. [Giunchiglia 2009] propose a Relation Based Access Control model (RelBAC), a formal model of permissions based on description logic. They require to specify who can access the data, thus relying on access control lists.

Carminati et al. [Carminati 2011] propose a fine-grained on-line social network access control model based on semantic web technologies. Their main idea is to encode social network-related information by means of an ontology. By constructing such an ontology, the authors model the Social Network Knowledge Base. They assume that a centralized reference monitor hosted by the social network manager will enforce the required policies. The access control policies are encoded as SWRL[68] rules. This approach is also based on the specification of who can access the resources. In other words, each access request is a triple (u, p, URI), where the user u requests to execute privilege p on the resource located at URI. A SWRL reasoner is requested to apply this approach.

5.2.3 Context-Aware Access Control

A number of access control models consider not only the information about the consumer who is accessing the data, but also the *context* of the request, e.g., time, location. A significant number of works in various research areas deal with context-aware access control:

[67] http://vocab.deri.ie/ppo
[68] http://www.w3.org/Submission/SWRL/

Hulsebosch et al. [Hulsebosch 2005] propose a context-sensitive access control infrastructure enriched by the verification of user-provided context information. Their work shows that context sensitive access control improves classic access control by imitating real-world authorization procedures. They provide a comprehensive overview of context verification techniques.

Bertino et al. [Bertino 2009] focus on location awareness and discuss the motivations behind enriching access control with position data. The location verification problem is presented, along with solutions (e.g. authenticator devices for physical location of users). Their approach relies on geographical role-based access control (GEO-RBAC), where users are assigned roles with given spatial validity.

Another contextual role-based approach is presented in Kulkarni and Tripathi [Kulkarni 2008]. The authors propose a context management layer in charge of authenticating the sensors that produce context data used to evaluate access.

Shen and Cheng [Shen 2011] propose a context-based access control model called SCBAC which combines Semantic Web technologies with a context-based access control mechanism. In particular, policies are expressed using SWRL [68]. They consider four types of contexts: subject contexts (our User and Device dimensions), object contexts, transaction contexts (our access privilege), and environment contexts (our Environment dimension). They do not apply their model to the Web of Data.

Covington et al. [Covington 2001] use the notion of role proposed by Role Based Access Control to capture the context of the environment in which the access requests are made. Environmental roles are defined using a prolog-like logical language for expressing policies. In a subsequent work, Covington proposes a context-aware attribute-based access control model called CABAC [Covington 2006]. They heavily rely on contextual attributes.

Cuppens and Cuppens-Boulahia [Cuppens 2008] propose an Organization Based Access Control (OrBAC) model where also context conditions are expressed. Context conditions are considered as extra statements to be satisfied to activate a security constraint and are based on Datalog rules. A context algebra is introduced. They do not entirely rely on Semantic Web languages and do not consider context data beyond temporal and spatial dimensions.

Corradi et al. [Corradi 2004] present UbiCOSM, a security middleware adopting context as a basic concept for policy specification and enforcement. The authors consider context as a first-class design principle to control access to resources. They distinguish among physical (i.e., physical spaces) and logical contexts (e.g., temporal conditions, user activities). They do not support additional context dimensions, e.g., the device. Policies are expressed at a high level of abstraction in terms of RDF metadata. Their approach is not applied to the Web of Data.

Toninelli et al. [Toninelli 2006, Toninelli 2007] adopt context-awareness and semantic technologies for access control but they do not apply their solution to the Web of Data (they focus on spontaneous coalitions of mobile agents). Their work follows two design guidelines: context-awareness to control resource access and semantic technologies for context and policy specification. They enforce access control with a rule-based approach relying on description logic. Their contextual informa-

tion does not include the device dimension and does not seem to be extensible. In a follow-up work [Toninelli 2009], the same authors present the Proteus policy framework, and discuss the role of the quality of context in access control systems.

Sadeh, Gandon, and Byung Kwon deal with context aware access control in the MyCampus experience [Sadeh 2005]. The work presents an infrastructure for context-aware service provisioning, focusing on privacy issues and usability. Their access control solution protects Web APIs, and features context-aware access rules (e.g. location and time are supported). These policies adopt a non-standard RDF syntax, and are fed to a rule engine for access enforcement. Although based on Semantic Web languages, the approach does not protect access to Linked Data resources, since it does not support neither HTTP operations protection, nor SPARQL query filtering.

Table 5.1 summarizes the main characteristics of the related work described above. None of the presented approaches satisfies all the features required for protecting HTTP operations on Linked Data, i.e., absence of ad-hoc policy languages, CRUD (Create, Read, Update, Delete) permission model, protection granularity up to triple-level, and expressive access control model to go beyond basic access control lists.

5.3 A Model for Context-Aware Access Control Policies

This section discusses the first challenge of the access control framework design, i.e. the definition of an access policy model that supports context-based conditions. The model used by Shi3ld is designed for both SPARQL queries and HTTP operations on Linked Data, and complies with the following key requirements:

Attribute-based Paradigm. Context-aware access policies cannot be expressed with an access control list paradigm or a role-based approach. Relying on attributes, instead, enables expressive access policies. That means, among all, the ability to create location-based and temporal-based access policies.

Semantic Web Languages Only. Shi3ld uses access policies defined with Semantic Web languages only, and no additional policy languages are introduced. This feature determines the adoption of a RDFS/OWL background ontology.

Granularity. The atomic resource protected by Shi3ld is a named graph in the SPARQL scenario and a Linked Data Platform Resource (LDPR) in the case of HTTP access to Linked Data. Hence, Shi3ld enables protection of whole datasets down to single triples.

CRUD Permission Model. Access policies are associated to specific permissions over the protected resource. It is therefore possible to specify rules satisfied only when the access is in *create*, *read*, *update* and *delete* mode.

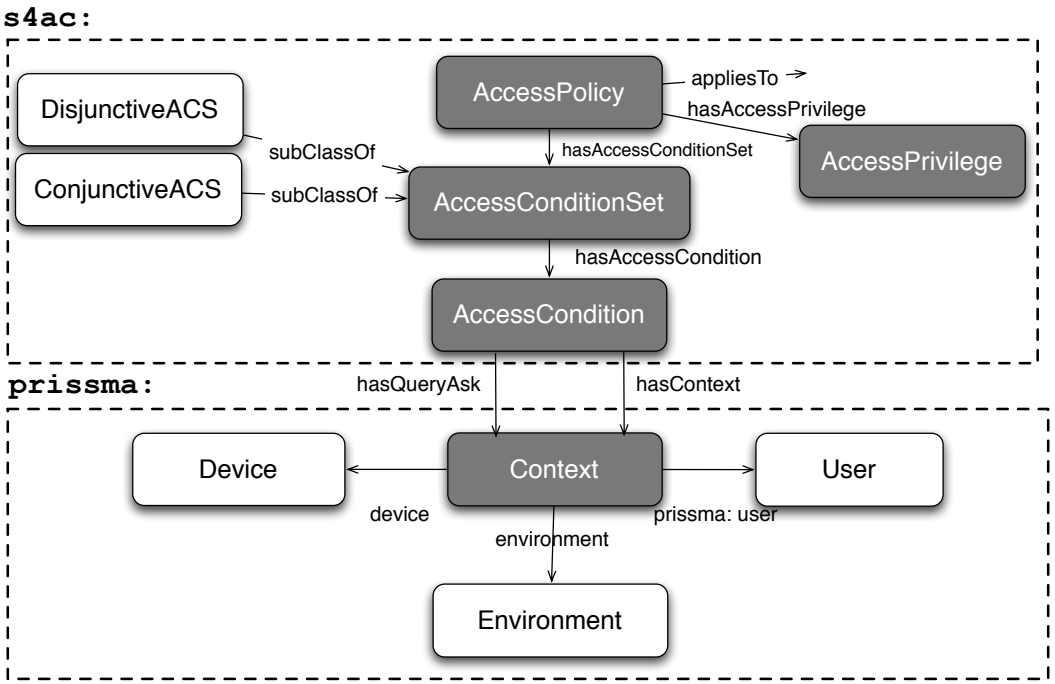


Figure 5.1: The Shi3ld model at a glance (gray boxes represent core classes).

The Shi3ld model consists in the interplay of two complementary RDFS/OWL ontologies (Figure 5.1): the S4AC[69] ontology deals with core access control concepts, and the PRISSMA vocabulary described in Chapter 3 focuses on client context[70].

Originally proposed by Villata et al. [Villata 2011], the S4AC vocabulary reuses concepts from SIOC[71], SKOS[72], WAC[66], SPIN[73], and Dublin Core[74]. The main component of the `S4AC` model is the Access Policy, as presented in Definition 17. Roughly, an Access Policy defines the constraints that must be satisfied to access a given named graph or a set of named graphs. If the Access Policy is *satisfied* the data consumer is allowed to access the data. Otherwise, access is denied. The constraints specified by Access Policies concern the data consumer, the device, the environment, or any given combination of these dimensions.

Definition 17 *(Access Policy) An Access Policy (P) is a tuple of the form* $P = \langle ACS, AP, R \rangle$ *where (i)* ACS *is a set of Access Conditions to satisfy, (ii)* AP *is an Access Privilege, and (iii)* R *is the resource protected by* P*.*

An Access Condition, as defined in Definition 18, expresses a constraint which needs to be verified to have the Access Policy satisfied.

[69] `http://ns.inria.fr/s4ac`
[70] Shi3ld can be adapted to support other definitions of context, stemming from different scenarios.
[71] `http://rdfs.org/sioc/spec`
[72] `http://www.w3.org/TR/skos-reference`
[73] `http://spinrdf.org/`
[74] `http://dublincore.org/documents/dcmi-terms`

`s4ac:AccessPolicy`	The class models the wrapper entity Access Policy.
`s4ac:AccessConditionSet`	The class represents an Access Condition Set.
`s4ac:DisjunctiveACS`	The class models a disjunctive Access Condition Set.
`s4ac:ConjunctiveACS`	The class models a conjunctive Access Condition Set.
`s4ac:AccessCondition`	The class represents an Access Condition.
`s4ac:AccessPrivilege`	The class models an Access Privilege. The class can be instantiated into `s4ac:Create`, `s4ac:Read`, `s4ac:Update`, `s4ac:Delete`
`s4ac:appliesTo`	The property associates an Access Policy to a target resource.
`s4ac:hasAccessPrivilege`	The property associates an Access Privilege to the policy.
`s4ac:hasAccessConditionSet`	The property associates a set of Access Conditions to the policy.
`s4ac:hasAccessCondition`	The property associates an Access Condition to an Access Condition Set, in a conjunctive or disjunctive manner, according to the nature of the set.
`s4ac:hasQueryAsk`	The property is used to add context conditions, expressed as SPARQL ASK queries. Such conditions are represented with the PRISSMA vocabulary. The property is adopted in SPARQL-based scenarios.
`s4ac:hasContext`	The property is used to add a `prissma:Context` entity to the Access Condition. This property is adopted in SPARQL-less scenarios, where Access Policy cannot embed SPARQL ASK queries.

Table 5.2: The S4AC vocabulary terms.

Definition 18 *(Access Condition) An Access Condition (AC) is a set of attributes that need to be satisfied to interact with a resource.*

Definition 19 *(Access Condition verification) If the triple pattern is present among the client attributes, then the Access Condition is said to be verified. If the query pattern is not found, then the Access Condition is said not to be verified.*

Access Conditions are organized in Access Condition Sets:

Definition 20 *(Access Condition Set) An Access Condition Set (ACS) is a set of access conditions of the form* $ACS = \{AC_1, AC_2, \ldots, AC_n\}$.

*ACS*s ease the reuse and combination of *AC*s to dataset administrators with shallow knowledge of SPARQL. The verification of an Access Condition Set returns a *true*/*false* answer and can be provided in a conjunctive or disjunctive fashion.

Definition 21 *(Conjunctive Access Condition Set) A Conjunctive Access Condition Set (CACS) is a logical conjunction of Access Conditions of the form* $CACS = AC_1 \wedge AC_2 \wedge \ldots \wedge AC_n$.

Definition 22 *(Conjunctive ACS evaluation) A CACS is verified if and only if every contained Access Condition is verified.*

Definition 23 *(Disjunctive Access Condition Set) A Disjunctive Access Condition Set (DACS) is a logical disjunction of Access Conditions of the form* $DACS = AC_1 \vee AC_2 \vee \ldots \vee AC_n$.

Definition 24 *(Disjunctive ACS evaluation) A DACS is verified if and only if at least one of the contained Access Conditions is verified.*

The Access Privilege (Definition 25) specifies the kind of operation the data consumer is allowed to perform on the resource(s) protected by the Access Policy.

Definition 25 *(Access Privilege) An Access Privilege (AP) is the set of allowed operations on the protected resource,* $AP = \{Create, Read, Update, Delete\}$.

We model the Access Privileges as four types of operations to keep a close relationship with CRUD-oriented access control systems, allowing a finer-grained access control beyond simple read/write privileges. We relate the four privilege classes to SPARQL 1.1 query and update language primitives through the SPIN ontology[73], which models SPARQL primitives as SPIN classes.

Dataset administrators protect resources using the `s4ac:appliesTo` property. Protected resources vary according to the current Shi3ld configuration: in the case of Shi3ld-SPARQL they consist in named graphs (Section 5.4), while Shi3ld-HTTP protects Linked Data Platform Resources and Containers (Section 5.5).

The Access Policy is associated to a Context, a class included in the PRISSMA vocabulary (`prissma:Context`). The class adopts the context definition provided and discussed in Chapter 1.

A complete list of S4AC vocabulary terms is shown in Table 5.2.

Example 1 *Figure 5.2 presents two sample access policies, expressed with and without SPARQL. The policy visualized in Figure 5.2a allows read-only access to the protected resource exclusively by a specific user and from a given location. The policy in Figure 5.2b authorizes the update of the resource by the given user, only if he is currently near Alice.*

```
@prefix prissma: <http://ns.inria.fr/prissma/v2#> .
@prefix s4ac: <http://ns.inria.fr/s4ac/v2#>.
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix : <http://example.org/shi3ld/>.

                                          PROTECTED
                                          RESOURCE
:policy1 a s4ac:AccessPolicy;
           s4ac:appliesTo :protected_res;          ACCESS
           s4ac:hasAccessPrivilege s4ac:Read;      PRIVILEGE
           s4ac:hasAccessConditionSet :acs1.

:acs1 a s4ac:AccessConditionSet;
        s4ac:ConjunctiveAccessConditionSet;
        s4ac:hasAccessCondition :ac1.
                                                ACCESS CONDITION
                                                  TO VERIFY
:ac1 a s4ac:AccessCondition;
     s4ac:hasQueryAsk
"""ASK
{?ctx a prissma:Context.
?ctx prissma:environment ?env.
?ctx prissma:user <http://johndoe.org/foaf.rdf#me>.
?env prissma:currentPOI ?poi.
?poi prissma:based_near ?p.
?p geo:lat ?lat; geo:lon ?lon.
FILTER(((?lat-45.8483) > 0 && (?lat-45.8483) < 0.5
|| (?lat-45.8483) < 0 && (?lat-45.8483) > -0.5)
&& ((?lon-7.3263) > 0 && (?lon-7.3263) < 0.5
|| (?lon-7.3263) < 0 && (?lon-7.3263) > -0.5 ))""".
```

(a) SPARQL-based

```
@prefix prissma: <http://ns.inria.fr/prissma/v2#> .
@prefix s4ac: <http://ns.inria.fr/s4ac/v2#>.
@prefix : <http://example.org/shi3ld/>.
                                          PROTECTED
                                          RESOURCE
:policy1 a s4ac:AccessPolicy;
           s4ac:appliesTo :protected_res;          ACCESS
           s4ac:hasAccessPrivilege s4ac:Update;    PRIVILEGE
           s4ac:hasAccessConditionSet :acs1.

:acs1 a s4ac:AccessConditionSet;
        s4ac:ConjunctiveAccessConditionSet;
        s4ac:hasAccessCondition :ac1.
                                                ACCESS CONDITION
                                                  TO VERIFY
:ac1 a s4ac:AccessCondition;
     s4ac:hasContext :ctx1.

:ctx1 a prissma:Context;
    prissma:user <http://johndoe.org/foaf.rdf#me>.
    prissma:environment :env1

:env1 a prissma:Environment;
      prissma:nearbyEntity <http://alice.org#me>.
```

(b) SPARQL-less

Figure 5.2: Shi3ld access policies: A policy embedding a SPARQL-based access condition in Figure 5.2a, and a policy expressed without SPARQL (5.2b).

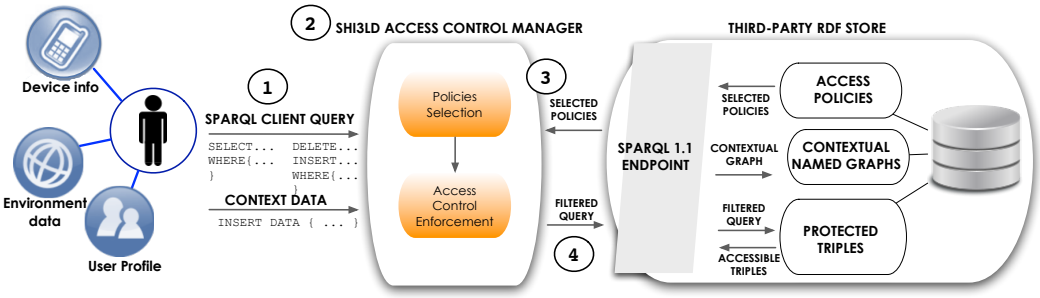


Figure 5.3: Shi3ld-SPARQL system architecture and access control enforcement flow.

In SPARQL-based Shi3ld policies, conflicts among policies might occur if the data provider uses Access Conditions with contrasting `FILTER` clauses. For instance, it is possible to define positive and negative statements such as `ASK{FILTER(?u=<http://example#bob>)}` and `ASK{FILTER(!(?u=<http://example#bob>))}`. If these two Access Conditions are applied to the same data, a logical conflict arises. This issue is handled in the framework by evaluating policies applied to a resource in a disjunctive way.

5.4 Shi3ld-SPARQL

This section answers the challenge of enforcing authorization on SPARQL endpoints, i.e. it describes Shi3ld-SPARQL and illustrates how it restricts the execution of SPARQL queries.

5.4.1 Access Control Enforcement

Shi3ld for SPARQL is built over the notion of Named Graph [Carroll 2005], thus supporting fine-grained access control policies, including the triple level. Enforcing permission models is an envisioned use case for RDF named graphs[75]. Shi3ld for SPARQL relies on named graphs to avoid depending on documents (one document can serialize several named graphs, one named graph can be split over several documents, and not all graphs come from documents[76]). At conceptual level, Shi3ld policies for SPARQL can be considered as access control conditions over g-boxes[77] (according to W3C RDF graph terminology), with semantics mirrored in the SPARQL language.

As seen in Figure 5.3, Shi3ld for SPARQL is conceived as a pluggable component for SPARQL endpoints. The access control enforcement flow is described below:

[75] http://www.w3.org/2011/rdf-wg/wiki/TF-Graphs-UC#.28_C_priority.29_Permissions

[76] The discussion about the use of named graphs in RDF 1.1 can be found at http://www.w3.org/TR/rdf11-concepts

[77] http://www.w3.org/2011/rdf-wg/wiki/GraphConceptTerminology

1. The mobile consumer queries the SPARQL endpoint to access the content. Context data is sent with the query and cached as a named graph using SPARQL 1.1 update language statements. Each time a context element is added we use an `INSERT DATA`, while we rely on a `DELETE/INSERT` when the contextual information is already stored and has to be updated. Summarizing, the mobile client sends two SPARQL queries: the first is the client query to the datastore (e.g. Figure 5.5a), the second provides contextual information.

2. The client query is filtered by the Shi3ld Access Control Manager instead of being directly executed on the SPARQL endpoint.

3. The Access Control Manager selects the set of policies affecting the client query, i.e. those with a matching Access Privilege. This is achieved by mapping the client query to one of the four Access Privileges defined by S4AC. The Access Conditions (SPARQL `ASK` queries) included in the selected policies are executed. According to the type of Access Condition Set (i.e., conjunctive or disjunctive), for each verified policy, the associated named graph is added to the set of accessible named graphs.

4. The client query is sent to the SPARQL endpoint with the addition of the following clauses:

   `FROM/FROM NAMED` clauses for `SELECT` queries, to execute the query only on the accessible named graphs, given the contextual information associated to the consumer. Adding the `FROM` clause is not enough because, in case the client query includes a `GRAPH` clause, we need to specify the set of named graphs to be queried in a `FROM NAMED` clause, otherwise the query will be executed on all the named graphs of the store;

   `USING/USING NAMED` clauses for `DELETE/INSERT`, `DELETE` and `INSERT` queries. The clauses describe a dataset in the same way as `FROM` and `FROM NAMED`. The keyword `USING` instead of `FROM` in update requests has been chosen to avoid possible ambiguities which could arise from writing `DELETE FROM`[78].

   Query execution is therefore performed only on the accessible named graphs, given the consumer contextual information. If the list of accessible named graphs is empty, the query is not executed.

Algorithm 6 is the main procedure for the execution of a query with access enforcement. The input of the algorithm is the client query Q and the RDF graph G_{ctx} modeling the client mobile context. It assumes the existence of a repository of access policies APS. The algorithm starts by saving the contextual graph in a local cache (line 1). At the beginning, the set of accessible named graph NGS is empty (line 3). The selection of the Access Policies is addressed by the sub-routine Access Policies Selection (line 4), which returns the set of Access Policies the query

[78] http://www.w3.org/TR/sparql11-update/#deleteInsert

```
PREFIX bobCtx: <http://example/contextgraphs/bobCtx>
PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX prissma: <http://ns.inria.fr/prissma/v2#>

ASK{?context a prissma:Context.
    ?context prissma:user ?u.
    ?u foaf:knows ex:alice#me.}
   VALUES ?context {(bobCtx:ctx)}

ASK {?context a prissma:Context.
     ?context prissma:environment ?env.
     ?env prissma:based_near ?p.
     FILTER (!(?p=ex:ACME boss#me))}
   VALUES ?context {(bobCtx:ctx)}
```

THE CONSUMER'S CONTEXT

Figure 5.4: SPARQL-based Shi3ld Access Conditions bound to incoming client context attributes.

```
DELETE {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Concert_tours>. }
INSERT {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Music_performance>. }
WHERE {ex:article a bibo:Article}
```

(a)

```
DELETE {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Concert_tours>. }
INSERT {ex:article dcterms:subject
        <http://dbpedia.org/page/Category: Music_performance>. }

USING :peter_data
USING NAMED :peter_data

WHERE {ex:article a bibo:Article}
```

THE NAMED GRAPH ACCESSIBLE BY THE CONSUMER

(b)

Figure 5.5: The SPARQL query issued by Bob's mobile client (a) and the *filtered* version (b).

is concerned by. Then, the algorithm runs all the Access Conditions composing the selected policies (lines 7-10). According to the type of Access Condition Set (i.e., conjunctive or disjunctive), for each verified policy, the associated named graph is added to the set of accessible named graphs (lines 11-12). Finally, after the execution of all Access Conditions, the client query is sent to the SPARQL endpoint with the addition of the `FROM` and `FROM NAMED` clauses (lines 16-17). Query execution is therefore performed only on the accessible named graphs, given the consumer contextual information. Line 19 outputs the triples resulting from Q.

Algorithm 7 is the Access Policies Selection routine. It selects the Access Policies concerned by the client query. The input of the algorithm is the query Q and the repository of the policies APS. We do not want to verify all the Access Policies every time a query is run. Thus, we adopt a selection mechanism to obtain only a subset of Access Policies to execute. In particular, the algorithm maps the client query to one of the four access privileges `S4AC` defines (line 1). Then, the algorithm selects all the Access Policies which have the identified Access Privilege (lines 3-5). The selected policies are returned to the main Access Enforcement algorithm (Algorithm 6).

Example. An example of client query is shown in Figure 5.5a, where Bob wants to access and modify the datastore (including Alice data `:alice_data`, protected by the policies in Figure 5.4) in such a way that all triples having `dcterms:subject Concert_tours` are changed into `dcterms:subject Music_performance`. Bob wants to perform such operation on the datastore from a given context. When the query is received by Shi3ld, the latter selects the Access Policies concerning this query The Access Conditions included in the policies are then coupled with a `VALUES` clause, as shown in Figure 5.4, where the `?context` variable is bound to Bob's actual context. The identification of the named graph(s) accessible by Bob returns, for example, only the graph `:peter_data`. Alice data is forbidden because Access Conditions evaluation leads to a `false` answer with Bob's context (Bob is near Alice's boss). The Access Control Manager adds the `USING`, `USING NAMED` clauses to constrain the execution of the client query only on the allowed named graph(s), i.e., `:peter_data`. The *filtered* client query is shown in Figure 5b.

5.4.2 Context Attribute Privacy

Privacy concerns arise when dealing with sensible mobile user context information such as current location. These data must be handled with a privacy-preserving mechanism. Recent surveys describe strategies to introduce privacy mainly in location-based services [Duckham 2010, Krumm 2009]. These works focus on two classes of solutions, *anonymity-based* and *obfuscation-based*. The goal of anonymity-based techniques is to use pseudonyms instead of real user IDs or adding ambiguity by grouping users. Obfuscation techniques reduces the quality of location data, e.g. using spatial degradation techniques. For example, the myCampus experience [Sadeh 2005], deals with access control and obfuscation rules for tracking mobile users. Shi3ld adopts an *anonymity-based* solution and delegates attribute anonymi-

Algorithm 6 Query Execution with Access Enforcement

Input: a SPARQL query Q, an RDF graph G_{ctx}, Access Policy Set APS
Output: the SPARQL query result R

save G_{ctx} in local contextual cache
if G_{ctx} *has changed* **then**
    $NGS = \emptyset$
    $APS \leftarrow$ `APSelection`(Q, APS)
    forall the $AP_i \in APS$ **do**
        $ACcount_{false} = 0$
        forall the $AC_j \in ACS_i$ **do**
            append G_{ctx} to AC_j as `VALUES` clause
            if `ASK`$_{AC_j}$ *execution returns false* **then**
                $ACcount_{false}++$
        if (ACS_{AP_i} *is DACS and* $ACcount_{false} < |ACS_{AP_i}|$)||($ACS_{AP_i}$ *is CACS and* $ACcount_{false} = 0$) **then**
            $NGS \leftarrow NGS \cup NG_{AP_i}$
else
    $NGS \leftarrow NGS_{cached}$
forall the $NG_i \in NGS$ **do**
    append `FROM` <NG_i> to Q
    append `FROM NAMED`<NG_i> to Q
$R \leftarrow$ run Q
return R

Algorithm 7 Access Policies Selection

Input: SPARQL client query Q, APS
Output: a reduced set of Access Policies APS_r

$AccPrv_Q \leftarrow$ map Q type to CRUD operation
$APS_r = \emptyset$
forall the $AP_i \in APS$ **do**
    if $AccPrv_{AP_i} \equiv AccPrv_Q$ **then**
        $APS_r \leftarrow APS_r \cup AP_i$
return APS_r

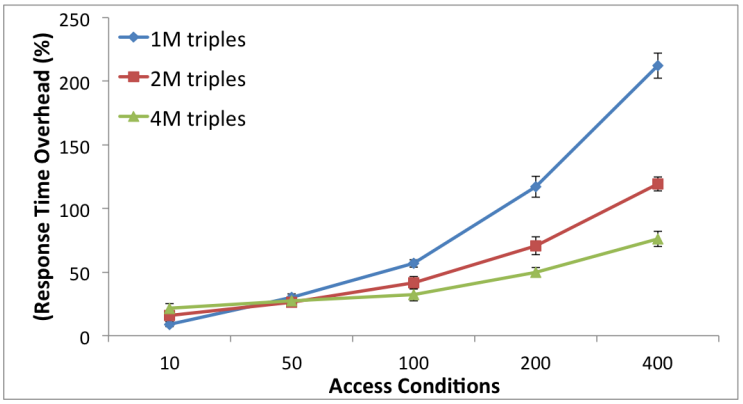
Figure 5.6: SPARQL Query Execution Procedure

sation to the client side, thus sensitive information is not disclosed to the server. We rely on partially encrypted RDF graphs, as proposed by Giereth [Giereth 2005]. Before building the RDF attribute graph and sending it to the Shi3ld-protected repository, a partial RDF encryption is performed, producing RDF-compliant results, i.e., the encrypted graph is still RDF (we use SHA-1 cryptographic hash function to encrypt RDF literals). On the server-side, every time a new policy is added to the system, the same operation is performed on the attributes included in access policies. As long as literals included in access conditions are hashed with the same function used on the client side, the Shi3ld authorization procedure still holds. The adopted technique does not guarantee full anonymity, as explained by Krumm et al. [Krumm 2009]. Nevertheless, the problem is mitigated by the short persistence of client-related data inside Shi3ld cache, since client attributes are deleted after each authorization evaluation. Note that encryption is not applied to location coordinates and timestamps, as this operation prevents geo-temporal filtering.

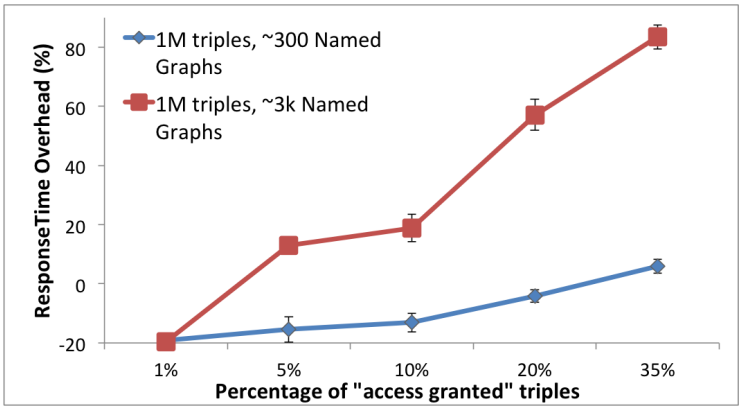
5.4.3 Evaluation

To assess the impact on response time, the Access Control Manager has been implemented as a Java EE component and plugged to the Corese-KGRAM RDF store and SPARQL 1.1 query engine[79] [Corby 2010]. Prototype evaluation is performed on an Intel Xeon E5540, Quad Core 2.53 GHz machine with 48GB of memory, using the Berlin SPARQL Benchmark (BSBM) dataset 3.1[80].
Figure 5.7a shows the execution of 10 independent runs of a test query batch consisting in 50 identical queries of a simple `SELECT` over `bsbm:Review` instances (tests are preceded by a warmup run). Response time (with and without access control) is measured. When executed against the Access Control Manager, the test SPARQL query is associated to the client context. Each Access Policy contains exactly one Access Condition. In Figure 5.7a, to simulate a worst-case scenario, access is granted to all named graphs defined in the base (i.e. all Access Conditions return true), so that query execution does not benefit from cardinality reduction. Larger datasets are less affected by the delay introduced by the prototype, as datastore size plays a predominant role in query execution time (e.g. for 4M triples and 100 always-true Access Policies response time delay accounts for 32.6%). The proposed solution is independent from the complexity of the incoming SPARQL query, as the only change is adding a list of FROM/FROM NAMED clauses (USING/USING NAMED for updates). Since there is no need to rewrite the query, the overhead is independent from query complexity.
In a typical scenario, the Access Control Manager restricts the results of a query. Figure 5.7b assesses the impact on performance for various levels of cardinality reduction, using modified versions of the BSBM dataset featuring a larger amount of named graphs (a higher number of `bsbm:RatingSite`s have been defined, thus obtaining more named graphs). When access is granted to a small fraction of named

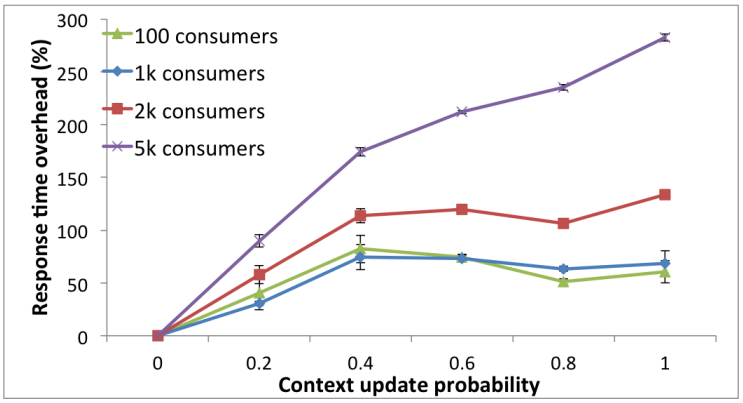
[79] http://tinyurl.com/corese-engine
[80] http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/Dataset/



(a)



(b)



(c)

Figure 5.7: Shi3ld-SPARQL Response time overhead

graphs, the query is executed faster than the case without access control (e.g. if access is granted to only 1% of named graphs, the query is executed ∼19% faster on the 1M triple test dataset). As more named graphs and triples are accessible, performance decreases. In particular, response time is affected by the construction of the active graph, determined by the merge of graphs in `FROM` clauses. As shown in Figure 5.7b, the cost of this operation grows with the number of named graphs returned by the evaluation of the Access Policies.
Figure 5.7c analyses the overhead introduced on response time by queries executed in dynamic mobile environments. Independent runs of 100 identical `SELECT` queries are executed, dealing with a range of context change probabilities. In case of a context update, the query is coupled with a SPARQL 1.1 DELETE INSERT query. Not surprisingly, with higher chances of updating context, the response time of the query grows, since more SPARQL queries need to be executed. The delay of `INSERT DATA` or `DELETE/INSERT` operations depends on the size of the triple store and on the number of named graphs (e.g. after a `DELETE` query, the adopted triple store refreshes internal structures to satisfy RDFS entailment). Performance is therefore affected by the number of active mobile users, since each of them is associated to a mobile context graph.

5.5 Shi3ld-HTTP

This section answers the challenge of enforcing authorization for HTTP operations on triple stores. The Semantic Web community is recently emphasizing the need for a substantially *"Web-like"* interaction paradigm with Linked Data. For instance, the W3C Linked Data Platform initiative[81] (LDP) promotes the use of read/write HTTP operations on triples, thus providing a *basic profile* for Linked Data servers and clients. Another example is the SPARQL 1.1 Graph Store Protocol[82], a set of guidelines to interact with RDF graphs with HTTP operations. Defining an access control model for these scenarios is still an open issue[83].

Protecting HTTP operations on Linked Data requires modifications to the Shi3ld architecture presented so far. Although still based on the attribute-based paradigm and the CRUD permission model, Shi3ld for HTTP (Shi3ld-HTTP) satisfies also the following requirements:

Protection of HTTP Access to Resources. Protected resources are retrieved and modified by clients using HTTP methods only, without SPARQL querying[84].

RDF-only Policies. If a SPARQL-less scenario is adopted, access conditions are defined with RDF triples only, i.e. with no embedded SPARQL.

[81] `http://www.w3.org/TR/ldp/`
[82] `http://www.w3.org/TR/sparql11-http-rdf-update/`
[83] `http://www.w3.org/2012/ldp/wiki/AccessControl`
[84] This is in compliance with the LDP specifications.

Granularity. The atomic element protected by Shi3ld is an HTTP resource represented in RDF.

Shi3ld-HTTP relies on the definition of resource provided by the W3C Linked Data Platform Working Group: LDP resources are HTTP resources queried, created, modified and deleted via HTTP requests processed by LDP servers[85].

The first configuration of Shi3ld for HTTP operations is the Shi3ld authorization framework for the SPARQL 1.1 Graph Store Protocol (Section 5.5.1). In Sections 5.5.2 and 5.5.3 we describe two scenarios tailored to the Linked Data Platform specifications, the second being completely SPARQL-less. The work is grounded on the analogies between SPARQL 1.1 functions and the HTTP protocol semantics, as suggested by the SPARQL Graph Store Protocol specification[82].

5.5.1 Shi3ld for SPARQL Graph Store Protocol

The SPARQL 1.1 HTTP Graph Store Protocol[82] provides an alternative interface to access RDF stored in SPARQL-equipped triple stores. The recommendation describes a mapping between HTTP methods and SPARQL queries, thus enabling HTTP operations on triples (Table 5.3). The Graph Store Protocol can be considered as an intermediate step towards an HTTP-only access to RDF datastores, since it still needs a SPARQL endpoint.

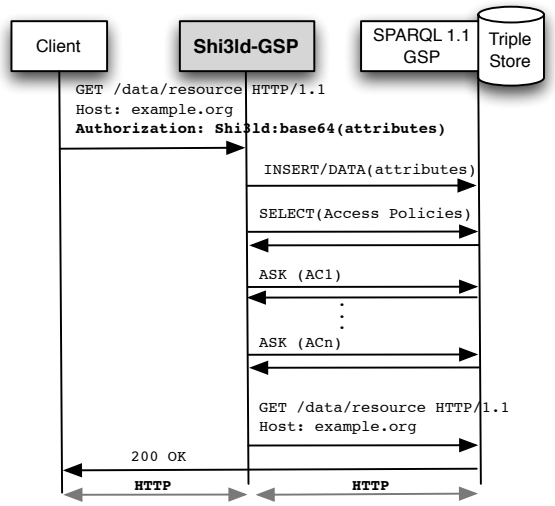
Figure 5.8a shows the architecture of the authorization procedure of Shi3ld for GSP-compliant SPARQL endpoints (Shi3ld-GSP). Shi3ld-GSP acts as a module protecting a stand-alone SPARQL 1.1 endpoint, equipped with a Graph Store Protocol module. First, the client performs an HTTP operation on a resource. This means that an RDF attribute graph is built on the client, serialized and sent with the request in the HTTP `Authorization` header[86]. Attributes are saved into the triple store with a SPARQL 1.1 query. Second, Shi3ld selects the access policies that protect the resource. The access conditions (SPARQL `ASK` queries, as in Figure 5.2a) included in the policies are then executed against the client attribute graph. Finally, the results are logically combined according to the type of access condition set (disjunctive or conjunctive) defined by each policy. If the result returns *true*, the HTTP query is forwarded to the GSP SPARQL engine, which in turns translates it into a SPARQL query. If the access is not granted, a HTTP 401 message is delivered to the client.

5.5.2 Shi3ld-LDP with Internal SPARQL Engine

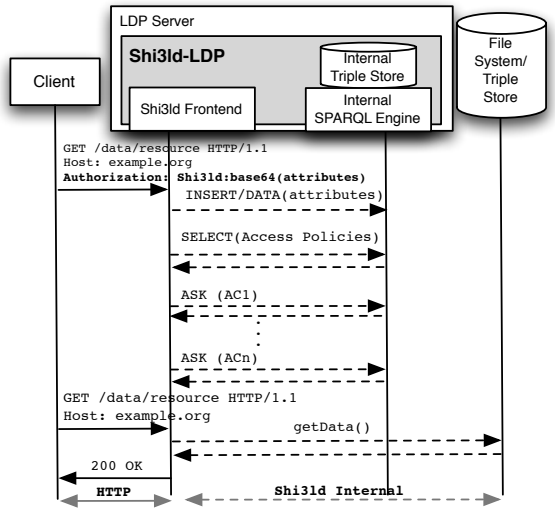
The Linked Data Platform initiative proposes a simplified configuration for Linked Data servers and Web-like interaction with RDF resources. Compared to the GSP

[85] An LDP server is an "*application program that accepts connections in order to service requests by sending back responses*" as specified by HTTP 1.1 definition, online at `http://www.ietf.org/rfc/rfc2616.txt`

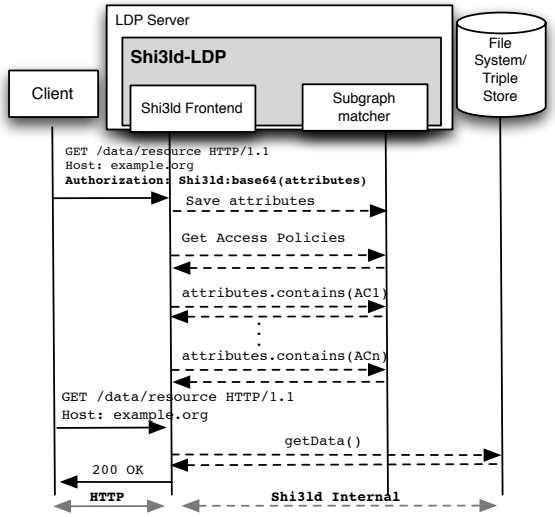
[86] Shi3ld for HTTP extends the header with the ad-hoc `Shi3ld` option. Other well-known proposals on the web re-use this field, e.g. the OAuth authorization protocol.



(a) Shi3ld-GSP



(b) Shil3d-LDP (internal SPARQL engine)



(c) Shi3ld-LDP (SPARQL-less)

Figure 5.8: Shi3ld for HTTP: Configurations

Operation	SPARQL 1.1	HTTP
Create	INSERT DATA, LOAD	POST
Read	SELECT, CONSTRUCT, ASK	GET
Update	DELETE/INSERT, INSERT	PUT
Delete	DELETE DATA, DELETE, CLEAR	DELETE

Table 5.3: Mapping of SPARQL 1.1 operations on HTTP methods.

case, authorization frameworks in this scenario must deal with a certain number of changes, notably the absence of SPARQL and potentially the lack of a graph store.

Shi3ld for the Linked Data Platform (Shi3ld-LDP) is adapted to work under these restrictions. The framework architecture is shown in Figure 5.8b. Shi3ld-LDP protects HTTP operations, but it does not communicate with an external SPARQL endpoint, i.e. there are no intermediaries between the RDF repository (the filesystem or a triple store) and Shi3ld. To re-use the authorization procedure previously described, an internal SPARQL engine is embedded into Shi3ld, along with an internal triple store. Although SPARQL is still present, this is perfectly legitimate in a Linked Data Platform scenario, since the use of the query language is limited to Shi3ld internals and is not exposed to the outside world[87]. Despite the architectural changes, the Shi3ld model remains unchanged. Few modifications occur to the authorization procedure as described in Figure 5.8a: clients send HTTP requests to the desired resource. HTTP headers contain the attribute graph, serialized as previously described in Section 5.5.1. Instead of relying on an external SPARQL endpoint, attributes are now saved internally, using an `INSERT DATA` query. The access policies selection and the access conditions execution remain substantially unchanged, but the whole process is transparent to the platform administrator, as the target SPARQL endpoint is embedded in Shi3ld.

5.5.3 SPARQL-less Shi3ld-LDP

To fulfill the Linked Data Platform recommendations, thus achieving a full-fledged *basic profile* for authorization frameworks, SPARQL is discontinued from the Shi3ld-LDP framework described in Section 5.5.2. Ditching SPARQL allows RDF-only access policies definition, and a leaner authorization procedure. To obtain a SPARQL-less framework, the access policy model and the logical steps of the previously described authorization procedure are re-used, although conveniently adapted (Figure 5.8c). First, Shi3ld-LDP policies adopt RDF only, as shown in Figure 5.2b: attribute conditions previously expressed with SPARQL ASK queries (Figure 5.2a) are expressed now as RDF graphs. Second, the embedded SPARQL engine used in Section 5.5.2 has been replaced: its task was testing whether client attributes verify the conditions defined in each access policy. This operation boils down to a *subgraph matching problem*. In other words, it must be checked if the access conditions (expressed in RDF) are contained into the attribute graph sent with the HTTP

[87] SPARQL is still visible in access policies (Figure 5.2a).

client query. Such subgraph matching procedure can be performed without introducing SPARQL in the loop. To steer clear of SPARQL, without re-inventing yet another subgraph matching procedure, the SPARQL interpreter is scrapped from the SPARQL engine [Corby 2010] used in Section 5.5.2, keeping only the underlying subgraph matching algorithm[88].

To understand the SPARQL-less policy verification procedure and the complexity hidden by the SPARQL layer, this section provides a description of the adopted subgraph matching algorithm, along with an overview of the RDF indexes used by the procedure. The algorithm checks whether a query graph Q (the access condition) is contained in the reference graph R (the client attributes sent with the query).

The reference graph R is stored in two key-value indexes (see example in Figure 5.9): index I_s stores the associations between property types and property subjects, and index I_o stores the associations between property types and property objects. Each RDF property type of R is therefore associated to a list of property subjects S_p and a list of property objects O_p. S_p contains URIs or blank nodes, O_p contains URIs, typed literals and blank nodes. Blank nodes are represented as anonymous elements, and their IDs are ignored.

The query graph Q, i.e., the access condition attributes, is serialized in a list L of subject-property-object elements $\{s_i, p_i, o_i\}$[89]. Blank nodes are added to the serialization as anonymous s_i or o_i elements.

The matching algorithm works as follows: for each subject-property-object $\{s_i, p_i, o_i\}$ in L, it looks up the indexes I_s and I_o using p_i as key. It then retrieves the list of property subjects S_p and the list of property objects O_p associated to p_i. Then, it searches for a subject in S_p matching with s_i, and an object in O_p matching with o_i. If both matches are found, $\{s_i, p_i, o_i\}$ is matched and the procedure moves to the next elements in L. If no match is found in either I_s or I_o, the procedure stops. Subgraph matching is successful if all L items are matched in the R index. Blank nodes act as wildcards: if a blank node is found in $\{s_i, p_i, o_i\}$ as object o_i or subject s_i, and O_p or S_p contains one or more blank nodes, the algorithm continues the matching procedure recursively, backtracking in case of mismatch and therefore testing all possible matchings. The example in Figure 5.9 shows a matching step of the algorithm, i.e., the successful matching of the triple "`_:b2 p:nearbyEntity http://alice.org/me`" against the client attributes indexes I_s and I_o. The highlighted triple is successfully matched against the client attributes R.

Note that policies might contain location and temporal constraints: Shi3ld-GSP (Section 5.5.1) and Shi3ld-LDP with internal SPARQL endpoint (Section 5.5.2) handle these conditions by translating RDF attributes into SPARQL `FILTER` clauses. The subgraph matching algorithm adopted by SPARQL-less Shi3ld-LDP does not support geo-temporal authorization evaluation yet.

[88] Third-party SPARQL-less Shi3ld-LDP implementations might adopt other off-the-shelf subgraph matching algorithms.

[89] A preliminary step replaces the query graph Q intermediate nodes into blank nodes. Blank nodes substitute SPARQL variables in the matching procedure.

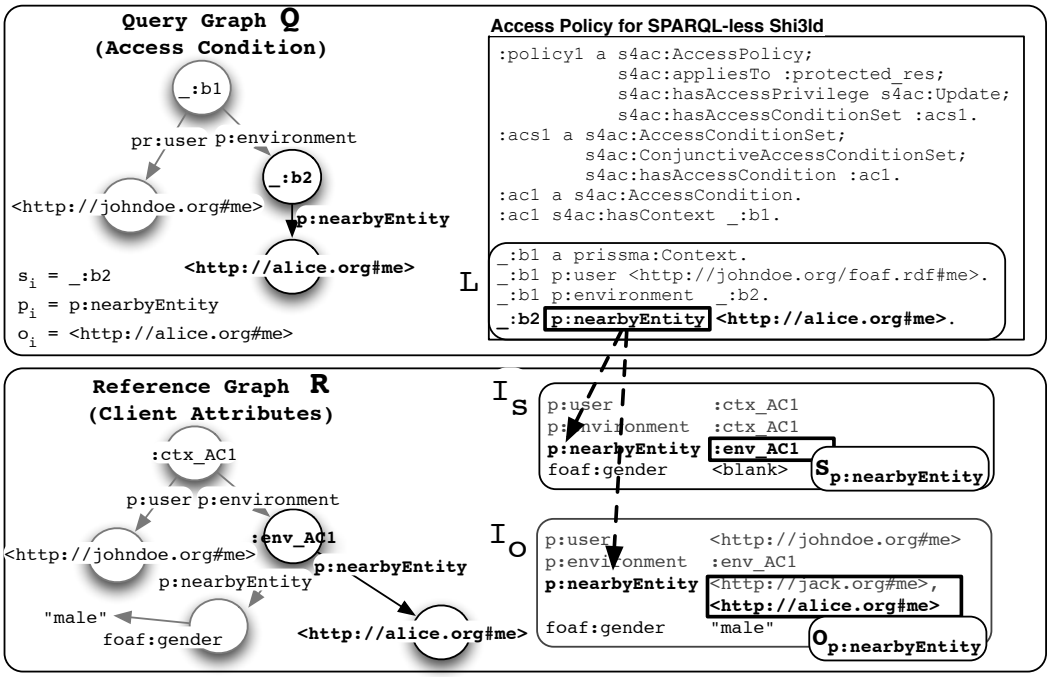


Figure 5.9: Example of subgraph matching used in the SPARQL-less Shi3ld-LDP

The three Shi3ld configurations described in this Section use the `Authorization` header to send client attributes. Even if there is no limit to the size of each header value, it is good practice to limit the size of HTTP requests, to minimize latency. Ideally, HTTP requests should not exceed the size of a TCP packet (1500 bytes), but in real world, finding requests that exceed 2KB is not uncommon, as a consequence of cookies, browser-set fields, and URL with long query strings[90]. To keep size as small as possible, before base-64 encoding, client attributes are serialized in turtle (less verbose that N-triples and RDF/XML). We plan to test the effectiveness of common lossless compression techniques to reduce the size of client attributes as future work. Furthermore, instead of sending the complete attribute graph in all requests, a server-side caching mechanism would enable the transmission of attribute graph deltas (i.e. only newly updated attributes will be sent to the server). Sending differences of RDF graphs is an open research topic[91], and it is out of the scope of the thesis.

5.5.4 Evaluation

The three scenarios presented in Section 5.5.1, 5.5.2 and 5.5.3 have been implemented as Java standalone web services[92]. The Shi3ld-GSP prototype works with

[90] https://developers.google.com/speed/docs/best-practices/request

[91] http://www.w3.org/2001/sw/wiki/How_to_diff_RDF

[92] Binaries, code and complete evaluation results are available at:
http://wimmics.inria.fr/projects/shi3ld-ldp

the Fuseki GSP-compliant SPARQL endpoint[93]. The Shi3ld-LDP prototype with internal SPARQL endpoint embeds the KGRAM/Corese[94] engine [Corby 2010]. The test campaign assesses the impact of Shi3ld on HTTP query response time[92]. Prototypes have been evaluated on an Intel Xeon E5540, Quad Core 2.53 GHz machine with 48GB of memory. In the test configuration, Shi3ld-GSP protects a Fuseki SPARQL server, while Shil3d-LDP scenarios secure RDF resources saved on the filesystem. First, the relationship between response time and the number of access conditions to verify is investigated. Second, the impact on response time of the complexity of access conditions is tested. The third test studies response time with regard to different HTTP methods. Five independent runs of a test query batch consisting in 50 HTTP operations have been executed (tests are preceded by a warmup run). Each query contains client attributes serialized in turtle (20 triples). The base-64 turtle serialization of the client attributes used in tests[92] is 1855 bytes long (including prefixes). Tests do not consider client-side literal anonymization (Section 5.4.2).

The first test shows the impact of the access conditions number on HTTP GET response time (Figure 5.10a and 5.10b). Each policy contains one access condition, each including 5 triples. The number of access conditions protecting the target RDF resource is progressively increased. Not surprisingly, the number of access conditions defined on the protected resource impacts on response time. Figure 5.10a shows the results for Shi3ld-LDP scenarios: data show a linear relationship between response time and access conditions number. The system has been tested up to 100 access conditions, although common usage scenarios have a smaller number of conditions defined for each resource. For example, the 5 access condition case is approximately 3 times slower than unprotected access. Nevertheless, ditching SPARQL improved performance: Figure 5.10a shows that the SPARQL-less configuration is in average 25% faster than its SPARQL-based counterpart, due to the absence of the SPARQL interpreter. As predicted, the delay introduced by Shi3ld-GSP is higher, e.g., 7 times slower for resources protected by 5 access policies (Figure 5.10b). This is mainly due to the HTTP communication between the Shi3ld-GSP module and Fuseki. Further delay is introduced by the Fuseki GSP module, that translates HTTP operations into SPARQL queries. Moreover, unlike Shi3ld-LDP scenarios, Shi3ld-GSP uses a shared RDF store for protected resources and access control-related data (client attributes and access policies). This increases the execution time of SPARQL queries, thus determining higher response time: Figure 5.10b, shows the behaviour of Shi3ld-GSP with two Fuseki server configurations: empty and with approximately 10M triples, stored in 17k graphs ("4-hop expansion Timbl crawl" part of the Billion Triple Challenge 2012 Dataset[95]). Results show an average response time difference of 14%, with a 27% variation for the 5 access condition case (Figure 5.10b). The number and the distribution of triples in the RDF store influence Shi3ld-GSP response time. Results might vary when Shi3ld-GSP is coupled with SPARQL endpoints adopting

[93] http://jena.apache.org/documentation/serving_data
[94] http://tinyurl.com/corese-engine
[95] http://km.aifb.kit.edu/projects/btc-2012/

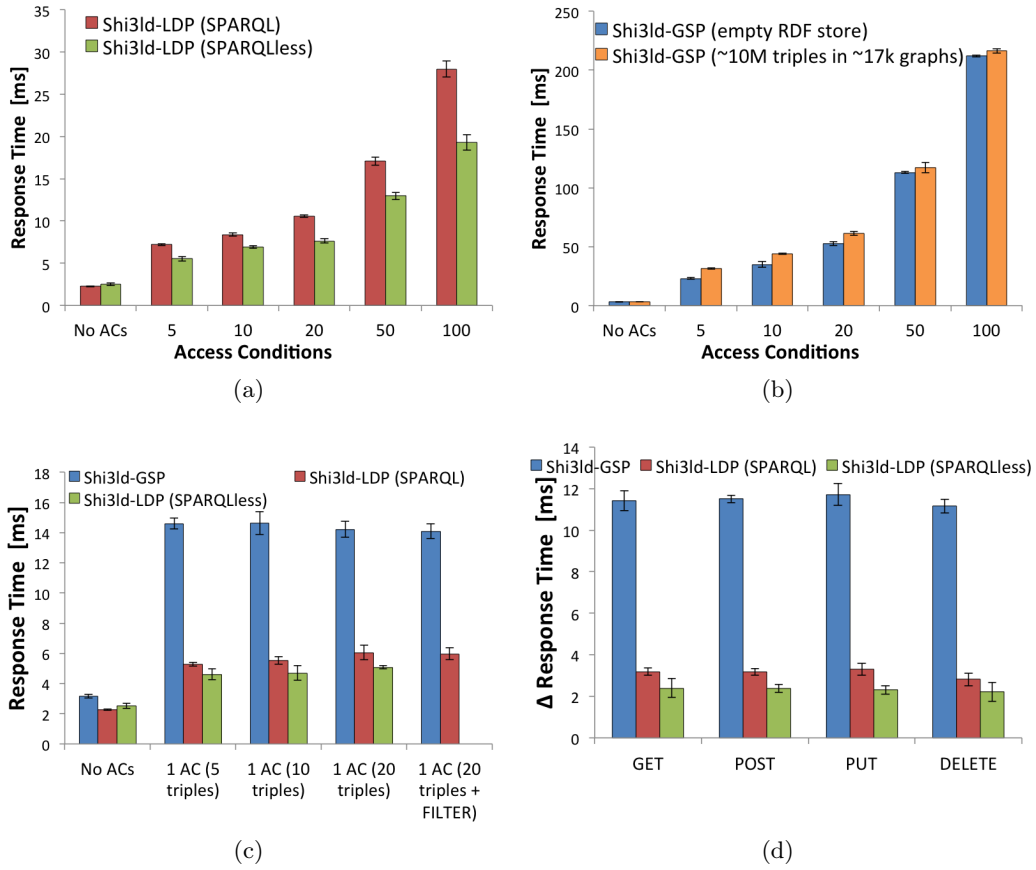


Figure 5.10: Shi3ld response time evaluation. The "No ACs" column shows performance without access control.

different indexing strategies or with different triple numbers and graph partitioning.

Figure 5.10c, shows the impact of access conditions complexity on HTTP GET response time. The requested resource is protected by a single access condition, with growing complexity: up to 20 triples are added, and an access condition containing a FILTER clause (for SPARQL-based scenarios only) is assessed. Results show no relevant impact on response time: this is because of the small size of the client attributes graph, over which access conditions are evaluated (client attributes in tests include 20 triples). Although attribute graph varies according to the application domain, it is reasonable that size will not exceed tens of triples.

The third test (Figure 5.10d) shows the delay introduced by Shi3ld for each HTTP operation. The figure displays the difference between response time with and without access control. HTTP GET, POST, PUT and DELETE methods are executed. Each HTTP method is associated to a 5-triple access condition. As predicted, the delay introduced by Shi3ld is independent from the HTTP method.

Section 5.2 contains a qualitative comparison with respect to the related work. On the other hand, addressing a quantitative comparison is a tricky task: among the list in Table 5.1, only few works explicitly designed for the Web come with an evaluation campaign [Abel 2007, Costabello 2012a, Flouris 2010, Hollenbach 2009, Muhleisen 2010]. Moreover, although some of these works provide a response time evaluation, the experimental conditions vary, making comparison difficult.

5.6 Policy Manager

A main drawback of Shi3ld is the assumption that dataset administrators *know* RDF and SPARQL, thus including a certain knowledge of Linked Data vocabularies and the ability of defining new named graphs. An effective back-end user interface to define Access Policies has to be designed as user interaction issues should not be underestimated. This section addresses this open issue by presenting a Web application that simplifies Shi3ld policies management by hiding the complexity of RDF and SPARQL to non-expert dataset administrators. The Shi3ld policy manager allows the definition of context-aware access conditions featuring user, environment (time and location above all), and device attributes. Moreover, such application allows a simpler definition of new named graphs over a set of existing triples. The work presented in this section can be classified among the works trying to hide the complexity of SPARQL and the Semantic Web to end users [Sonntag 2007, Lopez 2011, Ngomo 2013]. Such proposals mainly consist in GUIs to query, search, visualize, browse, and edit triples published on the Web of Data. The Shi3ld Policy Manager deals with querying issues and tackles the problem of providing a user-friendly interface for the creation of context-aware access control policies for RDF stores.

5.6.1 Policy Manager Features

The Shi3ld policy management GUI[96] is designed to support the interaction with two kinds of dataset administrators: *non-experts*, which are assumed not to know the SPARQL query language and RDF, and *experts*, which are able to edit access policies source code. In particular, it has the following functionalities:

- **Policies visualization and modification**: the application shows the list of policies stored in the triple store through a grid view. Each policy is an expandable row that, if selected, shows the main features of the selected policy, i.e., the plicy target (i.e. the named graphs protected by the policy), the privilege granted by the policy (Create, Update, Read, Delete), and the access conditions which specify the requirements that need to be satisfied to access the target resource. Users can edit all these elements, e.g., they associate the policy to another named graph, add or remove privileges, or modify the defined access conditions. Two different views are proposed to the user: i) a graphical view where operations are performed without the need to write policies using SPARQL and RDF, and ii) a textual editor which allows to directly write policies using SPARQL and RDF.

- **Policies creation**: the creation of a new context-aware policy is managed by a wizard which supports non-expert users. In particular, the wizard proposes the following views: *i)* the definition of the policy name (which is then "translated" into an `rdfs:label`), the target named graph (it is possible to select one of the already defined named graphs included in the triple store, or to define a new one, as detailed later), and privilege(s) to associate to the policy; *ii)* the view concerning User dimension, that consists in a text box where the administrator inserts the features that must be satisfied by the user accessing the target resource, e.g., `foaf:knows :ACME_boss`. The text box provides autocompletion and it suggests a list of properties and shows the associated vocabulary (to date, the Shi3ld Policy Manager uses the `foaf`[97] and `relationship`[98] vocabularies, but any vocabulary can be added); *iii)* the view concerning the Environment dimension, that consists in two parts: the first one defines temporal conditions, and the second one deals with geographical conditions. Temporal conditions are expressed with a time picker, to select the desired time interval in which access is granted. The definition of the geographical condition is done with a map interface[99], enriched with a movable marker and a resizeable radius; *iv)* the view concerning the Device dimension, similar to the User view, that suggests the access properties related to the device used to access the target resource (the Shi3ld Policy Manager uses the Delivery context vocabulary[100] but further vocabularies can be added). At the end of the wizard, the SPARQL/RDF access policy is automatically generated

[96] Video demonstration available at http://wimmics.inria.fr/projects/shi3ld/.
[97] http://xmlns.com/foaf/spec/
[98] http://purl.org/vocab/relationship/
[99] http://developers.google.com/maps/
[100] http://www.w3.org/TR/dcontology/

and stored in the triple store.

- **Named graphs creation**: the administrator is assisted in the definition of a new named graph. Shi3ld access policies must be associated to named graphs, and this leads to a number of difficult tasks for non-expert users, since it involves the use of non-trivial SPARQL features. The Shi3ld Policy Manager masks such complexity, by letting administrators define a new named graph starting from the set of triples they want to "insert" in such newly defined named graph. The application asks for the label of the named graph to be created and it presents the template of a `SELECT` query, to be completed with the desired triple pattern. A preview of the selected triples is shown, thus showing the administrator some sample triples that will be added to the named graph. If results are satisfying, the new named graph is created and it can be used as the access policy target.

Figure 5.13 shows how user actions are translated into SPARQL and RDF by the Shi3ld access policy manager. The application supports the administrator in creating, editing and deleting both policies and target named graphs. SPARQL queries are completely masked to end users, unless the embedded SPARQL textual editor is opened.

The Shi3ld Policy Manager is a Web application developed in JavaScript and backed by a Fuseki SPARQL 1.1 triple store[101]. The server-side relies on the Node.js platform[102], and the front-end is built over jQuery, the Twitter Bootstrap framework[103], and Backbone.js[104] as structure. The SPARQL editor is provided by Flint[105].

5.7 Conclusions

This chapter described the Shi3ld context-aware access control framework for Linked Data. Shi3ld is designed as a pluggable filter for SPARQL endpoints (Shi3ld-SPARQL), and RDF stores that support HTTP operations on Linked Data (Shi3ld-HTTP). Whenever a SPARQL query or an HTTP operation is performed on a target (named graph, or generic HTTP RDF resource), Shi3ld runs the authorization algorithm to check if the policies that protect the resource are satisfied or not. This is achieved by matching client context attributes sent with the query to the access policies. Such policies are defined with a lightweight RDF vocabulary. Furthermore, the Shi3ld Policy Manager Web application helps dealing with policies lifecyle.

Shi3ld policy model reaches triple-level granularity, relies only on Semantic Web languages, and supports CRUD privileges. Above all, Shi3ld model adds client

[101] http://jena.apache.org/documentation/serving_data/
[102] http://nodejs.org/
[103] http://twitter.github.io/bootstrap/
[104] http://backbonejs.org/
[105] http://openuplabs.tso.co.uk/demos/sparqleditor

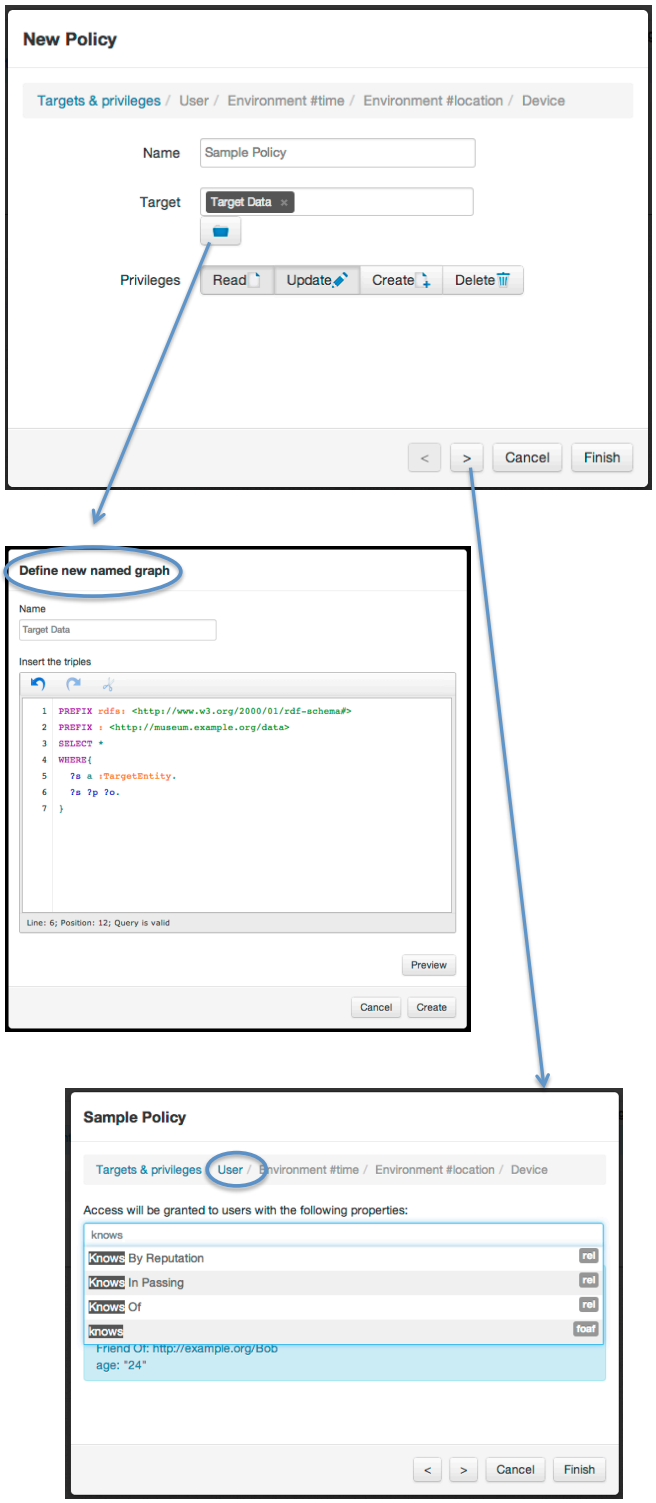


Figure 5.11: The Shi3ld Policy Creation Wizard. The new policy must be assigned a name, a target and a set of privileges. The wizard features the assisted creation of new targets, i.e. new named graphs. The triples contained in the new named graph are selected with a custom SPARQL query. Once the target is set, context conditions can be added, such as adding a `foaf:knows` triple with assisted typing support.



Figure 5.12: The Shi3ld Policy Creation Wizard. A time interval of several days is added, along with a location restriction on the Louvre museum, in Paris. Finally, a condition on the device operating system is added.

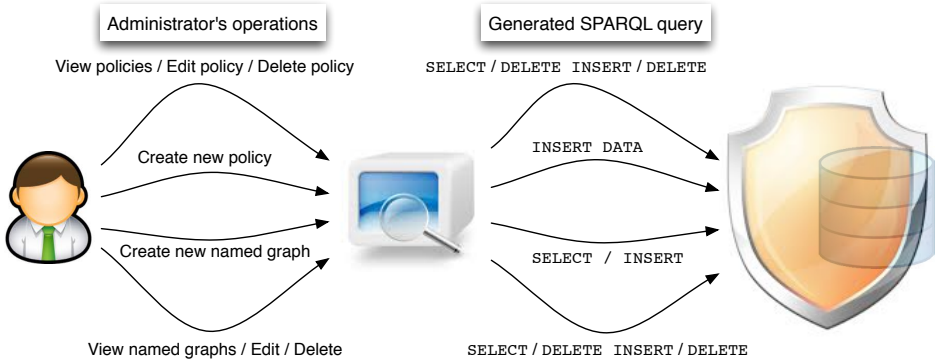


Figure 5.13: How the Shi3ld Access Policy Manager translates user interactions in RDF/SPARQL code.

context in control enforcement by embedding the PRISSMA vocabulary, thus enabling context-aware access policies. Shi3ld for SPARQL protects triples by relying on named graphs, and by changing the semantics of incoming SPARQL queries, whose scope is restricted to triples included in accessible named graphs only. Such mechanism delivers fine-grained protection, up to triple level. The list of accessible graphs is determined by evaluating pre-defined access policies against the actual mobile context of the requester. Shi3ld-HTTP comes in three distinct configurations: Shi3ld-GSP, for the SPARQL 1.1 Graph Store Protocol, and Shi3ld-LDP for the Linked Data Platform (with and without the internal SPARQL endpoint). The Shi3ld Policy Manager offers a wizard-like, simplified administration interface for dataset administrators not familiar with Semantic Web languages. More experienced users can edit triples manually, thus creating more complex policies.

Shi3ld-SPARQL prototype evaluation shows that when access is granted to a small fraction of named graphs, queries are executed faster than the case without access control. The delay introduced by Shi3ld-SPARQL grows with the number of Access Conditions in the system but has less impact on larger datasets while depending on the number of requesters. Shi3ld-HTTP evaluation confirms that Shi3ld-GSP is slower than the Shi3ld-LDP counterparts, due to the HTTP communication with the protected RDF store. Shi3ld-LDP with internal SPARQL endpoint introduces a 3x delay in response time (when resources are protected by 5 access conditions). Nevertheless, under the same conditions, the SPARQL-less solution exhibits a 25% faster response time. Tests show that response time grows linearly with the number of access conditions, and the complexity of each access condition does not increase the delay. When resources are protected by five access conditions, response time grows with a factor that is linear with the number of access conditions, and remains acceptable in each Shi3ld-HTTP configuration.

Future Shi3ld developments include a series of activities. To date, Shi3ld policies are only syntactically validated. A deeper, semantic validation procedure would help detecting intra and inter policy inconsistencies, and would report conflicts to the dataset administrator. The work presented in this chapter assumes that context

data is fetched and pre-processed beforehand. A library must be developed to deal with issue. Such solution must consider that the PRISSMA vocabulary (adopted by Shi3ld to model context) supports both raw context data (fetched directly from mobile sensors, e.g. GPS location, mobile features), and refined information (processed on board or by third-party, server-side services, e.g. POI resolution or user activity detection). Shi3ld deals with authorization only. Nevertheless, authentication issues cannot be ignored as the trustworthiness of client attributes is critical for a reliable access control framework. Shi3ld supports heterogeneous authentication strategies, since the attributes attached to each client request include heterogeneous data, ranging from user identity to environment information fetched by device sensors (e.g. location). The trustworthiness of user identity is achieved thanks to the WebID[64] compatibility: in Shi3ld, user-related attributes are modelled with the `foaf` vocabulary[106], thus easing the adoption of WebID. To date, no tamper-proof strategy is implemented in Shi3ld. Authenticating attributes fetched by client sensors is crucial to prevent tampering: Hulsebosch et al. [Hulsebosch 2005] provide a survey of verification techniques, such as heuristics relying on location history and collaborative authenticity checks. A promising approach is mentioned in Kulkarni and Tripathi [Kulkarni 2008], where client sensors are authenticated beforehand by a trusted party. Privacy concerns arise while dealing with mobile user context. Current Shi3ld privacy-preserving mechanism strategy must be improved (e.g. by supporting location data) and thoroughly evaluated. Furthermore, a caching mechanism for client attributes must be introduced, to speed up the authorization procedure. The caching mechanism must be coupled with an efficient strategy to send attributes updates, to reduce the average size of HTTP requests. Future developments of the Shi3ld Policy Manager will feature integration with the Linked Open Vocabulary catalogue (LOV)[107]. Hence, administrators will be supported in including new terms in access conditions from third-party vocabularies. Another Policy Manager extension will enable easier policies reuse by generating policy templates and publish them on a web of data datastore. Finally, a sandbox to test the access policy effectiveness on protected triples will be added to the interface.

[106] `http://xmlns.com/foaf/spec/`
[107] `http://lov.okfn.org/dataset/lov/`

Chapter 6

Conclusions and Perspectives

Contents

6.1 Summary of Contributions 115
6.2 Limitations and Open Issues 117
6.3 Publications 118
6.4 Perspectives 119

6.1 Summary of Contributions

The work presented in this thesis aims at enhancing Linked Data access with the notion of context-awareness. The contribution is threefold: first, the thesis presents the PRISSMA lightweight context vocabulary. The ontology is a mandatory step in enabling the other contributions of this work. The second main contribution is the PRISSMA adaptive presentation-level framework for Linked Data. Finally, the third work consists in the Shi3ld access control module for accessing RDF stores from context-aware devices.

First of all, context-aware features for enhancing access to the Web of Data need a proper context representation, i.e modelling context is a mandatory step required by the other contributions of the thesis. Unlike existing context ontologies, the PRISSMA vocabulary (presented in Chapter 3) reuses terms from third party Linked Data ontologies and provides the necessary extendibility to meet the requirement of the open world assumption, a key aspect of the Web and hence of Linked Data. Moreover, by modelling context as an information space defined as the sum of the mobile User model, device features, and environment, the PRISSMA vocabulary complies with mainstream context definitions. The ontology is published on the Web according to Linked Data principles and, in the light of Web of Data best practices, it relies on RDFS and on basic OWL features.

The second main contribution of the thesis addresses the adaptation of Linked Data presentation to context (Chapter 4). The PRISSMA presentation engine for Linked Data enables context-dependent visualization of RDF resources. PRISSMA introduces the concept of Prisms, i.e. RDF declarations containing formatting directives associated to a given context. When embedded in mobile applications, PRISSMA compares the current, sensed context to a series of Prisms. The most

similar Prism to the actual context is selected, and the embedded formatting directives are used to visualize the desired resource. The thesis explains how PRISSMA extends Fresnel engines with context awareness, thus enabling developers to declare context-dependent visualizations for Linked Data resources. Fresnel support for basic media-based presentation has been increased with full-fledged context-awareness. Relying on Fresnel favours the sharing and reuse of Prisms across applications, and does not introduce new formalisms other than RDF. The problem of selecting the most pertinent context-based representation has been solved by reducing the subgraph isomorphism problem to the computation of graph edit distance. This has been done with the adoption of Messmer and Bunke optimal subgraph isomorphism algorithm. The algorithm has been chosen because of a series of characteristics: first, it features a compact index structure (called decomposition) that does not contain subgraphs duplicates, thus fitting limited memory consumption requirements of mobile scenarios, as shown by quantitative tests. Moreover, the decomposition structure supports incremental updates, thus allowing on-the-fly addition of Prisms. Second, worst case computational complexity analysis shows that the algorithm is sublinear in the number of Prisms included in the decomposition. Hence, performance are not degraded by the (potentially high) number of context declarations. The algorithm has been adapted to RDF, and to graphs representing context information. Hence, it has been modified in several ways: the algorithm is now able to compare complex unit of content made of more than one node, the "context units" (e.g. location, or time information cannot be expressed by a single triple). The global cost function has been adapted to support heterogeneous context-attributes, thus adding a series of similarity measures in the computation of edit operations between items. For instance, location units are compared by computing the Haversine distance and applying an exponential decay on the result to account for location imprecision. Time similarity has been introduced. String units are compared with similarity measures (e.g. Monge-Elkan). The behaviour of such similarity functions for edit operations has been tested with different parameters, thus showing precision/recall trade-off. Operating on the client side guarantees privacy preservation, because context data does not have to be disclosed to third-party adaptation servers. A Response time test campaign proves the theoretical complexity analysis of the selection algorithm, and shows the sublinear dependence on the number of Prisms in the system. PRISSMA has been implemented as an Android library. A proof-of-concept adaptive, mobile Linked Data browser has been developed. The "PRISSMA Browser" is equipped with the PRISSMA library, and is therefore capable of adapting RDF instances according to the sensed context.

Besides adapting Web of Data entities to current context, the thesis deals with another access-related theme, the issue of access control for RDF stores queried in pervasive environments. Chapter 5 describes the third main contribution of the thesis, the Shi3ld access control manager. Shi3ld features a number of contributions. First of all, unlike other proposals, Shi3ld enables full-fledged context-dependent authorization for triples accessed from mobile scenarios. Since Shi3ld policies use RDFS/OWL ontologies (including the PRISSMA vocabulary, for expressing con-

text conditions), no additional languages or formalisms have been added: indeed, attribute-based access control policies are expressed with Semantic Web languages only (RDF and SPARQL). Shi3ld policies provide fine-grained protection, up to triple level, and support both conjunctive and disjunctive access conditions evaluation. Shi3ld includes a policy administration interface, to help dataset administrators manage the lifecycle of policies. Shi3ld is designed as a pluggable module for RDF stores, i.e. it is compatible with current triple stores and SPARQL 1.1 engines. Shi3ld protects Linked Data in case of different access strategies, i.e. both when triples are retrieved with SPARQL queries, and in case of read/write HTTP operations. Such feature determined two different Shi3ld configurations, Shi3ld-SPARQL and Shi3ld-HTTP. Shi3ld-SPARQL protects triples by relying on named graphs and changing the semantics of incoming SPARQL queries, whose scope is restricted to triples included only in the accessible named graphs. Shi3ld-HTTP comes in three distinct configurations: Shi3ld-GSP, for the SPARQL 1.1 Graph Store Protocol, and Shi3ld-LDP for the Linked Data Platform (with and without the internal SPARQL endpoint). Response time evaluation campaign for Shi3ld-SPARQL shows that when access is granted to a small fraction of named graphs, queries are executed faster than the case without access control. The delay introduced by Shi3ld-SPARQL grows with the number of Access Conditions in the system but has less impact on larger datasets while depending on the number of requesters. An equivalent test campaign has been run for Shi3ld-HTTP: evaluation confirms that the Shi3ld-GSP configuration is slower than the Shi3ld-LDP counterparts, due to the HTTP communication with the protected RDF store. Nevertheless, under the same conditions, the SPARQL-less solution exhibits a 25% faster response time. Tests show that response time grows linearly with the number of access conditions, and the complexity of each access condition does not impact on the delay.

6.2 Limitations and Open Issues

The PRISSMA framework uses Prisms declarations. To date, the issue of Prisms *distribution* has not been examined yet. Future evolutions of PRISSMA might support multiple strategies for discovering, retrieving, and consuming Prisms. For instance, given that Prisms are RDF declarations, it could be possible to distribute them on the Web of Data, according to Linked Data principles, thus creating *Linked Presentation-level Metadata*. Triple stores might store Prisms along with the associated entities. Otherwise, ad-hoc presentation-level repositories might be added to the Linked Data cloud. PRISSMA-powered applications might discover such context-based presentation metadata at run time, thus giving birth to applications that visualize RDF entities with "filters" created by third parties and associated to given contexts (e.g. to given locations only). Since Prisms are made of RDF triples, as a preliminary step before the search algorithm, they might be enriched with additional triples fetched from the Linked Data cloud, to obtain more complete context information (e.g. by dereferencing the entities included in each Prisms by

n-hops). Such Prism "expansion" with Linked Data will improve the precision of the search algorithm. The main limitation of our prism selection algorithm is the need for a proper parametrization of the cost function, and the choice of the most appropriate threshold for determining when graphs model the same context. This is a well-known issue of strategies based on graph edit distance. Instead of relying on heuristic parameter tuning, machine learning techniques might optimize the choice of the best-fitting similarity threshold values. Furthermore, future work will deal with enhancing the selection algorithm with other cost functions, such as semantic distance between URIs. Response time comparison with cited state-of-the-art solutions is envisaged, but such task would need to adapt these frameworks to a mobile scenario, since none of the related works is designed to run on mobile devices. Moreover, although some of these works provide a response time evaluation, experimental conditions vary, making comparison difficult. PRISSMA opens a series of questions: from a human-computer interaction point of view, it is legit to ask if context adaptation improves end-users experience in browsing the Web of Data. In other words, does it make Linked Data consumption "easier"? User acceptability evaluation needs to be performed with proof-of-concept applications, such as the PRISSMA Browser (Chapter 4.7). Another interesting point would be to assess the overhead that PRISSMA introduces on application developers.

There are several open issues related to Shi3ld. First of all, Shi3ld is built on the assumption that user context information is trustworthy. Nevertheless, the trustworthiness of the information sent by clients, including data describing context (e.g. location, device features, etc.) should not be taken for granted: future work needs to investigate this issue. Privacy concerns arise while dealing with mobile user context. Although Shi3ld already adopts a simple anonymization mechanism, it is necessary to improve such feature by adding a deeper privacy-preserving mechanism. Furthermore, a caching mechanism for client attributes must be introduced, to speed up the authorization procedure. The caching mechanism must be coupled with an efficient strategy to send attributes updates, to reduce the average size of client requests. An explanation mechanism for Shi3ld "access denied" responses should be deployed, if needed by the current use scenario.

6.3 Publications

The PRISSMA vocabulary and the PRISSMA presentation engine have been introduced in a doctoral consortium paper at the International Semantic Web Conference (ISWC) in 2011 [Costabello 2011]. An early version of a context-aware access control system for Linked Data has been published at the Linked Data on the Web Workshop (LDOW), co-located with WWW 2012 [Costabello 2012b]. Shi3ld for SPARQL, as presented in this thesis, appears in a 2012 paper presented at the European Conference on Artificial Intelligence (ECAI), [Costabello 2012a]. Shi3ld-HTTP has been presented at the Extended Semantic Web Conference (ESWC) 2013, [Costabello 2013a]. The Shi3ld administrator interface is demoed at ISWC

2013 [Costabello 2013b]. Extended versions of Shi3ld papers have been published in the Journal of Data Semantics (JoDS) [Villata 2013a] and as book chapter in "Security and Privacy Preserving in Social Networks" [Villata 2013b].

6.4 Perspectives

Linked Data is a promising trend in recent Web evolution. Mobile access to such information will grow, as happened for the classic Web. Context awareness, one of the peculiar features of mobile access, has already opened challenging scenarios for classic Web consumption, and is posed to play a crucial role in enhancing mobile consumption of Linked Data.

Adapting the presentation of RDF resources paves the way for services and applications tailored to the actual context. As context knowledge becomes deeper and more refined (e.g. by detecting the user needs and intentions), the PRISSMA adapting framework can deliver more powerful services, thus enhancing mobile applications with more precise and focused information. Access control will favour the publication of RDF datasets containing data outside the public domain. Hence, being able to control access according to context will enlarge the size of data available to Linked Data client applications.

The thesis contributions, context adaptation and access control, are only two aspects of the multifaceted enhancements of bringing context-awareness to Linked Data consumption. Context-aware access to the Web of Data inspires a series of open research questions. For instance, information retrieval might be improved, thus enabling context-based search engines on interlinked datasets. Linked Data discovery is another promising field of application for context-awareness: the notion of context might guide the finding of relevant triples in datasets unknown a priori, thus obtaining *context-guided* browsing patterns. In other words, it could be possible to envision context-aided follow-your-nose browsing sessions, where for each dereferenced resource, only the most pertinent properties are considered, and the remaining data is filtered out according to current context relevance. Moreover, instead of relying on classic "pull-like" interaction with triples, mobile devices might receive pertinent RDF resources in a push-like fashion. These resources could be cached on-board and used by Linked Data applications for further processing. Context might also influence the *creation* of Linked Data: for instance, context-awareness might foster Web of Data interlinking: the presence of real-world users in a given mobile context could determine the creation of temporary links between RDF resources published on the web of data. This will enhance Linked Data interlinking with non-factual, transient information, thus turning users in "interlinking hubs". A series of application scenarios will benefit from such research directions. Examples range from smart cities services to multi-modal interaction with the Web from rural areas.

Appendix A

PRISSMA Vocabulary

Listing A.1: The complete PRISSMA vocabulary

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix prissma: <http://ns.inria.fr/prissma/v2#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vann: <http://purl.org/vocab/vann/> .
@prefix vs: <http://www.w3.org/2003/06/sw-vocab-status/ns#> .
@prefix cc: <http://creativecommons.org/ns#> .
@prefix fresnel: <http://www.w3.org/2004/09/fresnel#> .


prissma: a owl:Ontology ;
       cc:license <http://creativecommons.org/licenses/by/3.0/> ;
    dc:creator <http://www-sop.inria.fr/members/Luca.Costabello/foaf.rdf#me> ;
    dc:description "A vocabulary to model context-aware presentation knowledge for RDF User
        Interfaces."@en ;
    dc:issued "2012-03-20"^^<http://www.w3.org/2001/XMLSchema#date> ;
    dc:modified "2012-03-20"^^<http://www.w3.org/2001/XMLSchema#date> ;
    dc:publisher <http://dbpedia.org/resource/
        National_Institute_for_Research_in_Computer_Science_and_Control> ;
    dc:title "PRISSMA: Presentation of Resources for Interoperable Semantic and Shareable
        Mobile Adaptability" ;
    vann:preferredNamespacePrefix "prissma" ;
    vann:preferredNamespaceUri "http://ns.inria.fr/prissma/v2#" ;
    foaf:page <http://ns.inria.fr/prissma/v2/prissma_v2.html> .

# ============== PRISSMA Classes ===========
prissma:Prism
    a rdfs:Class, owl:Class ;
    rdfs:comment "Wrapper class for describing the contextual conditions under which a given
        RDF presentation must be activated." ;
    rdfs:label "Prism" ;
    owl:equivalentClass fresnel:Group ;
    vs:term_status "stable"@en .

prissma:Context
    a rdfs:Class, owl:Class ;
    rdfs:comment "The Context class represents the mobile context and is equivalent to a
        fresnel:Purpose" ;
    rdfs:label "Context" ;
    owl:equivalentClass fresnel:Purpose ;
    vs:term_status "stable"@en .

prissma:User
    a rdfs:Class, owl:Class ;
    rdfs:comment "Represents the target mobile user associated to a prissma:Context. To
        provide more flexibility, the class can be used to model both user stereotypes and
        specific users, according to the designer needs." ;
```

```
44      rdfs:label "User" ;
45      owl:equivalentClass foaf:Person ;
46      vs:term_status "stable"@en .
47
48  prissma:Device
49      a rdfs:Class, owl:Class ;
50      rdfs:comment "The Device represents the mobile device on which Web of Data resource
            consumption takes place. It enables device-specific data representation. It is
            equivalent to the Device class of the delivery context ontology" ;
51      rdfs:label "Device" ;
52      owl:equivalentClass <http://www.w3.org/2007/uwa/context/deliveryContext.owl#Device> ;
53      vs:term_status "stable"@en .
54
55  prissma:Environment
56      a rdfs:Class, owl:Class ;
57      rdfs:comment "The class Environment models the user context in which the resource
            consumption takes place, therefore enabling customized resource presentation
            according to specific situations. " ;
58      rdfs:label "Environment" ;
59      vs:term_status "stable"@en .
60
61
62  prissma:Activity
63      a rdfs:Class, owl:Class ;
64      rdfs:comment "The Activity class consists in a placemark aimed at modeling a high-level
            representation of an user action, such as 'running', 'driving', 'working', 'shopping
            ', etc." ;
65      rdfs:label "Activity" ;
66      vs:term_status "testing"@en .
67
68  prissma:POI
69      a rdfs:Class, owl:Class ;
70      rdfs:comment "The class models a Point of Interest (POI) and consists in a simplified
            version of W3C Point of Interest Core specifications. POIs are defined as entities
            that \"describe information about locations such as name, category, unique identifier
            , or civic address\"." ;
71      rdfs:label "POI" ;
72      rdfs:subClassOf <http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing> ;
73      vs:term_status "stable"@en .
74
75
76
77  # ============== PRISSMA Properties ===========
78
79  prissma:user
80      a rdf:Property ;
81      rdfs:comment "The property associates a User to a Context" ;
82      rdfs:domain :Context> ;
83      rdfs:label "user" ;
84      rdfs:range prissma:User ;
85      vs:term_status "stable"@en .
86
87  prissma:device
88      a rdf:Property ;
89      rdfs:comment "The property associates a Device to a Context" ;
90      rdfs:domain prissma:Context ;
91      rdfs:label "device" ;
92      rdfs:range :Device> ;
93      vs:term_status "stable"@en .
94
95  prissma:environment
96      a rdf:Property ;
```

```
97      rdfs:comment "The property associates an Environment to a Context" ;
98      rdfs:domain prissma:Context ;
99      rdfs:label "environment" ;
100     rdfs:range prissma:Environment ;
101     vs:term_status "stable"@en .
102
103 prissma:poi
104     a rdf:Property ;
105     rdfs:comment "The property associates a POI to a prissma:Environment" ;
106     rdfs:domain prissma:Environment ;
107     rdfs:label "poi" ;
108     rdfs:range :POI> ;
109     vs:term_status "stable"@en .
110
111 prissma:poiCategory
112     a rdf:Property ;
113     rdfs:comment "Associates a category to a POI (e.g. monument, restaurant, etc.)" ;
114     rdfs:domain prissma:POI;
115     rdfs:label "poiCategory" ;
116     vs:term_status "stable"@en .
117
118 prissma:poiLabel
119     a rdf:Property ;
120     rdfs:comment "Associates an identifying resource to a POI (e.g. a given monument, a
            specific restaurant, etc.)" ;
121     rdfs:domain prissma:POI ;
122     rdfs:label "poiLabel" ;
123     vs:term_status "stable"@en .
124
125 prissma:radius
126     a rdf:Property, owl:DatatypeProperty ;
127     rdfs:comment "Specifies the geographic extension of a POI. Value is expressed in metres."
            ;
128     rdfs:domain prissma:POI ;
129     rdfs:label "radius" ;
130     rdfs:range <http://www.w3.org/2001/XMLSchema#nonNegativeInteger> ;
131     vs:term_status "stable"@en .
132
133 prissma:string
134     a rdf:Property ;
135     rdfs:comment "Associates any given high-level representation of motion to a prissma:
            Environment" ;
136     rdfs:domain prissma:Environment ;
137     rdfs:label "motion" ;
138     vs:term_status "testing"@en .
139
140 prissma:nearbyEntity
141     a rdf:Property ;
142     rdfs:comment "The environmental proximity of a generic real-world entity can trigger
            different resource representations. The property is therefore used to associate
            nearby objects to the Environment model." ;
143     rdfs:domain prissma:Environment ;
144     rdfs:label "nearbyObject" ;
145     rdfs:range owl:Thing ;
146     vs:term_status "testing"@en .
```

Bibliography

[Abel 2007] Fabian Abel, Juri Luca De Coi, Nicola Henze, Arne Wolf Koesling, Daniel Krause and Daniel Olmedilla. *Enabling Advanced and Context-Dependent Access Control in RDF Stores.* In Karl Aberer, Natasha Fridman, Lyndon Nixon, Peter Mika, Diana Maynard, Riichiro Mizoguchi and Guus Schreiber, editors, ISWC, LNCS, vol. 4825, pages 1–14. Springer, 2007. (Cited on pages 80, 82, 83 and 108.)

[Adipat 2011] Boonlit Adipat, Dongsong Zhang and Lina Zhou. *The Effects of Tree-View Based Presentation Adaptation on Model Web Browsing.* MIS Quarterly, vol. 35, no. 1, pages 99–121, 2011. (Cited on pages 18, 39 and 40.)

[Auer 2010] Sören Auer, Raphael Doehring and Sebastian Dietzold. *LESS - Template-Based Syndication and Presentation of Linked Data.* In In Procs of ESWC, pages 211–224, 2010. (Cited on page 41.)

[Baldauf 2007] Matthias Baldauf, Schahram Dustdar and Florian Rosenberg. *A survey on context-aware systems.* Int. J. of Ad Hoc and Ubiquitous Computing, vol. 2, no. 4, 2007. (Cited on pages 17 and 18.)

[Becker 2009] Christian Becker and Christian Bizer. *Exploring the Geospatial Semantic Web with DBpedia Mobile.* J. Web Sem., vol. 7, no. 4, pages 278–286, 2009. (Cited on page 12.)

[Berners-Lee 2006a] Tim Berners-Lee. *Linked Data - Design Issues.* `http://www.w3.org/DesignIssues/LinkedData.html`, 2006. (Cited on pages 8 and 9.)

[Berners-Lee 2006b] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer and David Sheets. *Tabulator: Exploring and analyzing linked data on the semantic web.* In Proceedings of the 3rd International Semantic Web User Interaction Workshop, volume 2006, 2006. (Cited on page 12.)

[Bertino 2009] Elisa Bertino and Michael S. Kirkpatrick. *Location-Aware Authentication and Access Control.* In Procs of the IEEE 23rd Int. Conf. on Advanced Information Networking and Applications (AINA-2009), pages 10–15, 2009. (Cited on page 86.)

[Bolchini 2007] Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber and Letizia Tanca. *A data-oriented survey of context models.* SIGMOD Rec., vol. 36, no. 4, pages 19–26, December 2007. (Cited on pages 22 and 24.)

[Butter 2007] Thomas Butter, Markus Aleksy, Philipp Bostan and Martin Schader. *Context-aware User Interface Framework for Mobile Applications.* In ICDCS Workshops, page 39, 2007. (Cited on pages 37 and 40.)

[Carminati 2011] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu and Bhavani M. Thuraisingham. *Semantic web-based social network access control*. Computers & Security, vol. 30, no. 2-3, pages 108–115, 2011. (Cited on page 85.)

[Carroll 2002] Jeremy J. Carroll. *Matching RDF Graphs*. In Procs of ISWC, pages 5–15, 2002. (Cited on page 45.)

[Carroll 2005] Jeremy J. Carroll, Christian Bizer, Patrick J. Hayes and Patrick Stickler. *Named graphs*. J. Web Sem., vol. 3, no. 4, pages 247–267, 2005. (Cited on page 93.)

[Castano 2011] Silvana Castano, Alfio Ferrara, Stefano Montanelli and Gaia Varese. *Ontology and Instance Matching*. In Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, volume 6050 of *LNCS*, pages 167–195. Springer, 2011. (Cited on page 45.)

[Champin 2009] Pierre-Antoine Champin. *T4R: Lightweight presentation for the Semantic Web*. In Scripting for the Semantic Web, workshop at ESWC, 2009. (Cited on page 41.)

[Chen 2004] Harry Chen, Filip Perich, Timothy W. Finin and Anupam Joshi. *SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*. In Procs of the 1st Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous-2004), 2004. (Cited on pages 24 and 25.)

[Chen 2005] Yu Chen, Xing Xie, Wei-Ying Ma and Hong-Jiang Zhang. *Adapting Web Pages for Small-Screen Devices*. IEEE Internet Computing, vol. 9, no. 1, pages 50–56, January 2005. (Cited on pages 37 and 40.)

[Cohen 2003] William W. Cohen, Pradeep D. Ravikumar and Stephen E. Fienberg. *A Comparison of String Distance Metrics for Name-Matching Tasks*. In IIWeb, pages 73–78, 2003. (Cited on pages 66, 67 and 68.)

[Conte 2004] Donatello Conte, Pasquale Foggia, Carlo Sansone and Mario Vento. *Thirty Years Of Graph Matching In Pattern Recognition*. IJPRAI, pages 265–298, 2004. (Cited on pages 45 and 52.)

[Corby 2010] Olivier Corby and Catherine Faron-Zucker. *The KGRAM Abstract Machine for Knowledge Graph Querying*. In Procs of WI, pages 338–341. IEEE, 2010. (Cited on pages 98, 104 and 106.)

[Corradi 2004] Antonio Corradi, Rebecca Montanari and Daniela Tibaldi. *Context-Based Access Control Management in Ubiquitous Environments*. In Procs of NCA, pages 253–260. IEEE, 2004. (Cited on pages 83 and 86.)

[Costabello 2011] Luca Costabello. *DC Proposal: PRISSMA, Towards Mobile Adaptive Presentation of the Web of Data*. In Procs of ISWC, pages 269–276, 2011. (Cited on pages 48 and 118.)

[Costabello 2012a] L. Costabello, S. Villata and F. Gandon. *Context-Aware Access Control for RDF Graph Stores*. In Procs of ECAI, pages 282–287, 2012. (Cited on pages 83, 108 and 118.)

[Costabello 2012b] Luca Costabello, Serena Villata, Nicolas Delaforge and Fabien Gandon. *Linked Data Access Goes Mobile: Context-Aware Authorization for Graph Stores*. In LDOW, 2012. (Cited on page 118.)

[Costabello 2013a] Luca Costabello, Serena Villata, Oscar Rodriguez Rocha and Fabien Gandon. *Access Control for HTTP Operations on Linked Data*. In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink and Sebastian Rudolph, editors, The Semantic Web: Semantics and Big Data, volume 7882 of *Lecture Notes in Computer Science*, pages 185–199. Springer Berlin Heidelberg, 2013. (Cited on pages 83 and 118.)

[Costabello 2013b] Luca Costabello, Serena Villata, Iacopo Vagliano and Fabien Gandon. *Assisted Policy Management for SPARQL Endpoints Access Control*. In International Semantic Web Conference (Posters & Demos), pages 33–36, 2013. (Cited on page 119.)

[Covington 2001] Michael J. Covington, Wende Long, Srividhya Srinivasan, Anind K. Dey, Mustaque Ahamad and Gregory D. Abowd. *Securing Context-aware Applications using Environment Roles*. In Procs of SACMAT, pages 10–20. ACM, 2001. (Cited on pages 83 and 86.)

[Covington 2006] Michael J. Covington and Manoj R. Sastry. *A Contextual Attribute-Based Access Control Model*. In Procs of the Workshops On the Move to Meaningful Internet Systems (OTM-2006), LNCS 4278, pages 1996–2006, 2006. (Cited on page 86.)

[Cui 2010] Yanqing Cui, Mikko Honkala, Kari Pihkala, Kimmo Kinnunen and Guido Grassel. *Linked internet UI: a mobile user interface optimized for social networking*. In Mobile HCI, pages 45–54, 2010. (Cited on page 18.)

[Cuppens 2008] Frédéric Cuppens and Nora Cuppens-Boulahia. *Modeling Contextual Security Policies*. Int. J. Inf. Sec., vol. 7, no. 4, pages 285–305, 2008. (Cited on pages 83 and 86.)

[Dadzie 2011] Aba-Sah Dadzie, Matthew Rowe and Daniela Petrelli. *Hide the Stack: Toward Usable Linked Data*. In Procs of ESWC, volume 6643 of *LNCS*, pages 93–107. Springer, 2011. (Cited on page 41.)

[David 2010] Jérôme David, Jérôme Euzenat*et al.* *Linked data from your pocket: The Android RDFContentProvider*. In Proc. 9th demonstration track on international semantic web conference (ISWC), pages 129–132, 2010. (Cited on page 12.)

[Dey 2001] Anind K. Dey. *Understanding and Using Context*. Personal Ubiquitous Comput., vol. 5, no. 1, pages 4–7, 2001. (Cited on pages 3, 16, 21, 26, 33 and 35.)

[Duckham 2010] Matt Duckham. *Moving Forward: Location Privacy and Location Awareness*. In Procs of SPRINGL, pages 1–3. ACM, 2010. (Cited on pages 17 and 96.)

[Euzenat 2007] Jérôme Euzenat and Pavel Shvaiko. Ontology matching. Springer, 2007. (Cited on pages 15 and 45.)

[Euzenat 2008] JÉrÔme Euzenat, JÉrÔme Pierson and Fano Ramparany. *Dynamic context management for pervasive applications*. Knowl. Eng. Rev., vol. 23, no. 1, pages 21–49, March 2008. (Cited on page 18.)

[Fernéandez 2012] Josep Maria Brunetti Fernéandez, Sören Auer and Roberto Garcia. *The Linked Data Visualization Model*. In ISWC (Posters & Demos), 2012. (Cited on page 41.)

[Finin 2008] Timothy W. Finin, Anupam Joshi, Lalana Kagal, Jianwei Niu, Ravi S. Sandhu, William H. Winsborough and Bhavani M. Thuraisingham. *ROWLBAC: representing role based access control in OWL*. In Procs of SACMAT, pages 73–82. ACM, 2008. (Cited on pages 83 and 85.)

[Flouris 2010] Giorgos Flouris, Irini Fundulaki, Maria Michou and Grigoris Antoniou. *Controlling Access to RDF Graphs*. In Arne-Jørgen Berre, Asunción Gómez-Pérez, Kurt Tutschku and Dieter Fensel, editors, FIS, LNCS, vol. 6369, pages 107–117. Springer, 2010. (Cited on pages 80, 83, 84 and 108.)

[Gandon 2005] Fabien L. Gandon. *Generating Surrogates to Make the Semantic Web Intelligible to End-Users*. In Web Intelligence, pages 352–358, 2005. (Cited on pages 40 and 41.)

[Gao 2010] Xinbo Gao, Bing Xiao, Dacheng Tao and Xuelong Li. *A survey of graph edit distance*. Pattern Analysis & Applications, vol. 13, no. 1, pages 113–129, 2010. (Cited on pages 45 and 46.)

[Gasimov 2010] Anar Gasimov, Fabio Magagna and Juliana Sutanto. *CAMB: context-aware mobile browser*. In Procs of the 9th International Conference on Mobile and Ubiquitous Multimedia (MUM-2010), 2010. (Cited on pages 38 and 40.)

[Giereth 2005] Mark Giereth. *On Partial Encryption of RDF-Graphs*. In Yolanda Gil, Enrico Motta and V. Richard Benjamins, editors, ISWC, LNCS, vol. 3729, pages 308–322. Springer, 2005. (Cited on page 98.)

[Giunchiglia 2009] Fausto Giunchiglia, Rui Zhang and Bruno Crispo. *Ontology Driven Community Access Control*. In Procs of SPOT, 2009. (Cited on pages 83, 84 and 85.)

[Hartig 2010] Olaf Hartig and Andreas Langegger. *A Database Perspective on Consuming Linked Data on the Web*. Datenbank-Spektrum, vol. 10, no. 2, pages 57–66, 2010. (Cited on page 14.)

[Hasnain 2012] Ali Hasnain, Mustafa Al-Bakri, Luca Costabello, Zijie Cong, Ian Davis, Tom Heath*et al.* *Spamming in Linked Data*. In Third International Workshop on Consuming Linked Data (COLD2012), 2012. (Cited on page 15.)

[Heath 2008] Tom Heath. *How Will We Interact with the Web of Data?* IEEE Internet Computing, vol. 12, no. 5, pages 88–91, 2008. (Cited on page 33.)

[Heath 2011] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space (1st edition). Morgan & Claypool, 2011. (Cited on pages 8, 9, 12, 14, 23, 47 and 80.)

[Henricksen 2004] Karen Henricksen and Jadwiga Indulska. *Modelling and Using Imperfect Context Information*. In PerCom Workshops, pages 33–37, 2004. (Cited on page 50.)

[Hervás 2011] Ramón Hervás and José Bravo. *Towards the ubiquitous visualization: Adaptive user-interfaces based on the Semantic Web*. Interacting with Computers, vol. 23, no. 1, pages 40–56, 2011. (Cited on pages 24 and 25.)

[Hollenbach 2009] James Hollenbach, Joe Presbrey and Tim Berners-Lee. *Using RDF Metadata To Enable Access Control on the Social Semantic Web*. In Procs of CK, 2009. (Cited on pages 80, 83, 84 and 108.)

[Hong 2009] Jong-yi Hong, Eui-ho Suh and Sung-Jin Kim. *Context-aware systems: A literature review and classification*. Expert Systems with Applications, vol. 36, no. 4, pages 8509–8522, 2009. (Cited on pages 16, 18 and 19.)

[Hulsebosch 2005] R. Hulsebosch, Alfons Salden, Mortaza Bargh, P. Ebben and J. Reitsma. *Context Sensitive Access Control*. In Procs of SACMAT, pages 111–119. ACM, 2005. (Cited on pages 80, 83, 86 and 114.)

[Huynh 2002] David Huynh, David R. Karger and Dennis Quan. *Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF*. In Semantic Web Workshop, 2002. (Cited on pages 40 and 41.)

[Kiefer 2007] Christoph Kiefer, Abraham Bernstein and Markus Stocker. *The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks*. In Procs of ISWC/ASWC, volume 4825 of *LNCS*, pages 295–309. Springer, 2007. (Cited on pages 44 and 45.)

[Kirrane 2013] Sabrina Kirrane, Ahmed Abdelrahman, Alessandra Mileo and Stefan Decker. *Secure Manipulation of Linked Data*. In International Semantic Web Conference (1), pages 248–263, 2013. (Cited on pages 83 and 84.)

[Korpipää 2003] Panu Korpipää and Jani Mäntyjärvi. *An Ontology for Mobile Device Sensor-Based Context Awareness.* In Procs of the 4th International and Interdisciplinary Conference Modeling and Using Context (CONTEXT-2003), LNCS 2680, 2003. (Cited on pages 22, 24 and 25.)

[Krumm 2009] John Krumm. *A Survey of Computational Location Privacy.* Personal Ubiquitous Comput., vol. 13, no. 6, pages 391–399, August 2009. (Cited on pages 17, 96 and 98.)

[Krummenacher 2007] Reto Krummenacher and Thomas Strang. *Ontology-based context modeling.* In Proceedings Third Workshop on Context-Aware Proactive Systems (CAPS 2007)(June 2007), 2007. (Cited on pages 22 and 24.)

[Kulkarni 2008] Devdatta Kulkarni and Anand Tripathi. *Context-aware Role-based Access Control in Pervasive Computing Systems.* In Procs of SACMAT, pages 113–122. ACM, 2008. (Cited on pages 86 and 114.)

[Laakko 2005] Timo Laakko and Tapio Hiltunen. *Adapting Web Content to Mobile User Agents.* IEEE Internet Computing, vol. 9, no. 2, pages 46–53, 2005. (Cited on pages 38 and 40.)

[Laakko 2008] Timo Laakko. *Context-Aware Web Content Adaptation for Mobile User Agents.* In Richi Nayak, Nikhi Ichalkaranje and LakhmiC. Jain, editors, Evolution of the Web in Artificial Intelligence Environments, volume 130 of *Studies in Computational Intelligence*, pages 69–99. Springer Berlin Heidelberg, 2008. (Cited on page 38.)

[Le Phuoc 2010] Danh Le Phuoc, Josiane Xavier Parreira, Vinny Reynolds and Manfred Hauswirth. *RDF On the Go: RDF Storage and Query Processor for Mobile Devices.* In ISWC Posters&Demos, 2010. (Cited on page 12.)

[Lemlouma 2004] Tayeb Lemlouma and Nabil Layaida. *Context-Aware Adaptation for Mobile Devices.* IEEE International Conference on Mobile Data Management, 2004. (Cited on pages 37 and 40.)

[Lopez 2011] Vanessa Lopez, Victoria S. Uren, Marta Sabou and Enrico Motta. *Is Question Answering fit for the Semantic Web?: A survey.* Semantic Web, vol. 2, no. 2, pages 125–155, 2011. (Cited on page 108.)

[Lovett 2012] Tom Lovett and Eamonn O'Neill. Mobile context awareness, volume 10. Springer, 2012. (Cited on pages 16 and 19.)

[m. c. schraefel 2010] m. c. schraefel and Lloyd Rutledge. *User interaction in semantic web research.* J. Web Sem., vol. 8, no. 4, pages 375–376, 2010. (Cited on page 34.)

[Malandrino 2010] Delfina Malandrino, Francesca Mazzoni, Daniele Riboni, Claudio Bettini, Michele Colajanni and Vittorio Scarano. *MIMOSA: context-aware*

adaptation for ubiquitous web access. Personal and Ubiquitous Computing, vol. 14, no. 4, pages 301–320, 2010. (Cited on pages 39 and 40.)

[Messmer 1998] B.T. Messmer and H. Bunke. *A new algorithm for error-tolerant subgraph isomorphism detection*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 20, no. 5, pages 493–504, 1998. (Cited on pages 44, 46, 52, 53, 54, 59, 60, 62 and 66.)

[Mitrovic 2002] Nikola Mitrovic and Eduardo Mena. *Adaptive User Interface for Mobile Devices*. In DSV-IS, pages 29–43, 2002. (Cited on page 37.)

[Muhleisen 2010] Hannes Muhleisen, Martin Kost and Johann-Christoph Freytag. *SWRL-based Access Policies for Linked Data*. In Procs of SPOT, 2010. (Cited on pages 80, 83, 85 and 108.)

[Nathanail 2007] Spyros Nathanail, Vassileios Tsetsos and Stathes Hadjiefthymiades. *Sensor-Driven Adaptation of Web Document Presentation*. In Universal Access in Human-Computer Interaction. Applications and Services, volume 4556 of *LNCS*. 2007. (Cited on pages 37 and 40.)

[Ngomo 2013] Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann and Daniel Gerber. *Sorry, i don't speak SPARQL: translating SPARQL queries into natural language*. In Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates and Sue Moon, editors, WWW, pages 977–988. International World Wide Web Conferences Steering Committee / ACM, 2013. (Cited on page 108.)

[Nigay 1993] Laurence Nigay and Joëlle Coutaz. *A design space for multimodal systems: concurrent processing and data fusion*. In Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems, CHI '93, pages 172–178, New York, NY, USA, 1993. ACM. (Cited on page 35.)

[Paternò 2010] Fabio Paternò and Giuseppe Zichittella. *Desktop-to-Mobile Web Adaptation through Customizable Two-Dimensional Semantic Redesign*. In HCSE, pages 79–94, 2010. (Cited on pages 39 and 40.)

[Perera 2013] C. Perera, A. Zaslavsky, P. Christen and D. Georgakopoulos. *Context Aware Computing for The Internet of Things: A Survey*. Communications Surveys Tutorials, IEEE, vol. PP, no. 99, pages 1–41, 2013. (Cited on page 17.)

[Pietriga 2006] Emmanuel Pietriga, Christian Bizer, David Karger and Ryan Lee. *Fresnel: A Browser-Independent Presentation Vocabulary for RDF*. In Procs of ISWC, volume 4273 of *LNCS*, pages 158–171. Springer, 2006. (Cited on pages 4, 34, 41, 47 and 48.)

[Preuveneers 2004] Davy Preuveneers, Jan Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers and Koen Bosschere. *Towards an Extensible Context Ontology for Ambient Intelligence*. In Panos Markopoulos, Berry Eggen, Emile Aarts and JamesL. Crowley, editors, Ambient Intelligence, volume 3295 of *Lecture Notes in Computer Science*, pages 148–159. Springer Berlin Heidelberg, 2004. (Cited on pages 24 and 25.)

[Quan 2003] Dennis Quan, David Huynh and David R Karger. *Haystack: A platform for authoring end user semantic web applications*. In The semantic web-ISWC 2003, pages 738–753. Springer, 2003. (Cited on page 12.)

[Quan 2005] Dennis Quan and David R. Karger. *Xenon: An RDF Stylesheet Ontology*. In Procs of WWW, 2005. (Cited on pages 40 and 41.)

[Reynolds 2010] Vinny Reynolds, Michael Hausenblas, Axel Polleres, Manfred Hauswirth and Vinod Hegde. *Exploiting linked open data for mobile augmented reality*. In W3C Workshop: Augmented Reality on the Web. June, 2010. (Cited on page 12.)

[Riesen 2010] Kaspar Riesen, Xiaoyi Jiang and Horst Bunke. *Exact and Inexact Graph Matching: Methodology and Applications*. In Managing and Mining Graph Data, volume 40, pages 217–247. Springer, 2010. (Cited on page 54.)

[Rutledge 2005] Lloyd Rutledge, Jacco van Ossenbruggen and Lynda Hardman. *Making RDF presentable: integrated global and local semantic Web browsing*. In Procs of WWW, pages 199–206, 2005. (Cited on page 41.)

[Sacco 2011a] Owen Sacco and Alexandre Passant. *A Privacy Preference Manager for the Social Semantic Web*. In Procs of the 2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation (SPIM-2011), 2011. (Cited on page 85.)

[Sacco 2011b] Owen Sacco, Alexandre Passant and Stefan Decker. *An Access Control Framework for the Web of Data*. In Proc. of TrustCom, pages 456–463. IEEE, 2011. (Cited on pages 80, 83 and 85.)

[Sadeh 2005] Norman M Sadeh, Fabien L Gandon and Oh B Kwon. *Ambient intelligence: The mycampus experience*. Rapport technique, DTIC Document, 2005. (Cited on pages 83, 87 and 96.)

[Sandhu 1996] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein and Charles E. Youman. *Role-Based Access Control Models*. IEEE Computer, vol. 29, no. 2, pages 38–47, 1996. (Cited on pages 82 and 85.)

[Schilit 1994] B. Schilit, N. Adams and R. Want. *Context-aware computing applications*. In Mobile Computing Systems and Applications, 1994. WMCSA

1994. First Workshop on, pages 85–90. IEEE, 1994. (Cited on pages 15, 19, 36 and 40.)

[Schmidt 2000] Albrecht Schmidt, Antti Takaluoma and Jani Mäntyjärvi. *Context-Aware Telephony Over WAP*. Personal and Ubiquitous Computing, vol. 4, no. 4, pages 225–229, 2000. (Cited on page 18.)

[Shen 2011] Haibo Shen and Yu Cheng. *A Semantic Context-Based Model for Mobile Web Services Access Control*. I. J. Computer Network and Information Security, vol. 1, pages 18–25, 2011. (Cited on pages 83 and 86.)

[Sonntag 2007] Daniel Sonntag and Philipp Heim. *A Constraint-Based Graph Visualisation Architecture for Mobile Semantic Web Interfaces*. In Bianca Falcidieno, Michela Spagnuolo, Yannis S. Avrithis, Ioannis Kompatsiaris and Paul Buitelaar, editors, SAMT, volume 4816 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2007. (Cited on page 108.)

[Strang 2003] Thomas Strang, Claudia Linnhoff-Popien and Korbinian Frank. *CoOL: A Context Ontology Language to Enable Contextual Interoperability*. In Procs of the 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS-2003), 2003. (Cited on pages 24 and 25.)

[Strang 2004] T. Strang and C. Linnhoff-Popien. *A context modeling survey*. In Workshop Proceedings, 2004. (Cited on page 17.)

[Toninelli 2006] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal and Ora Lassila. *A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments*. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold and Lora Aroyo, editors, ISWC, LNCS, vol. 4273, pages 473–486. Springer, 2006. (Cited on pages 83 and 86.)

[Toninelli 2007] Alessandra Toninelli, Rebecca Montanari, Lalana Kagal and Ora Lassila. *Proteus: A Semantic Context-Aware Adaptive Policy Model*. In Procs of the 8th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY-2007), pages 129–140, 2007. (Cited on page 86.)

[Toninelli 2009] Alessandra Toninelli, Antonio Corradi and Rebecca Montanari. *A Quality of Context-Aware Approach to Access Control in Pervasive Environments*. In Proceedings of the 2nd International Conference on Mobile Wireless Middleware, Operating Systems, and Applications (MOBILWARE-2009), volume 7 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 236–251. Springer, 2009. (Cited on page 87.)

[Ullmann 1976] J. R. Ullmann. *An Algorithm for Subgraph Isomorphism*. J. ACM, vol. 23, no. 1, pages 31–42, 1976. (Cited on page 58.)

[Villata 2011] Serena Villata, Nicolas Delaforge, Fabien Gandon and Amelie Gyrard. *An Access Control Model for Linked Data*. In Proceedings of the 7th International IFIP Workshop on Semantic Web & Web Semantics (SWWS-2011), volume 7046, pages 454–463. Springer, LNCS, 2011. (Cited on page 88.)

[Villata 2013a] Serena Villata, Luca Costabello, Nicolas Delaforge and Fabien Gandon. *A Social Semantic Web Access Control Model*. J. Data Semantics, vol. 2, no. 1, pages 21–36, 2013. (Cited on page 119.)

[Villata 2013b] Serena Villata, Luca Costabello, Fabien Gandon, Cathrine Faron-Zucker and Michel Buffa. *Social Semantic Network-based Access Control*. In Richard Chbeir and Bechara Al Bouna, editors, Security and Privacy Preserving in Social Networks. Springer, 2013. (Cited on page 119.)

[Volz 2009] Julius Volz, Christian Bizer, Martin Gaedke and Georgi Kobilarov. *Discovering and Maintaining Links on the Web of Data*. In Procs of ISWC, pages 650–665, 2009. (Cited on pages 44 and 45.)

[Wang 2004] X.H. Wang, Da Qing Zhang, Tao Gu and H.K. Pung. *Ontology based context modeling and reasoning using OWL*. In Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, pages 18–22, 2004. (Cited on pages 24 and 25.)

[Weber 2012] Markus Weber, Marcus Liwicki and Andreas Dengel. *Faster subgraph isomorphism detection by well-founded total order indexing*. Pattern Recognition Letters, vol. 33, no. 15, 2012. (Cited on page 58.)

[Weiser 1991] Mark Weiser. *The computer for the 21st century*. Scientific american, vol. 265, no. 3, pages 94–104, 1991. (Cited on page 15.)

[Woensel 2011] William Van Woensel, Sven Casteleyn and Olga De Troyer. *A generic approach for on-the-fly adding of context-aware features to existing websites*. In HT, pages 143–152, 2011. (Cited on pages 38 and 40.)

[Zhang 2007] Dongsong Zhang. *Web content adaptation for mobile handheld devices*. Commun. ACM, vol. 50, no. 2, pages 75–79, February 2007. (Cited on pages 37 and 40.)

[Zimmermann 1980] H. Zimmermann. *OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection*. Communications, IEEE Transactions on, vol. 28, no. 4, pages 425–432, 1980. (Cited on page 35.)

[Zou 2012] Lei Zou, Lei Chen, M. Tamer Özsu and Dongyan Zhao. *Answering pattern match queries in large graph databases via graph embedding*. VLDB J., vol. 21, no. 1, pages 97–120, 2012. (Cited on pages 44 and 45.)