FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

ILDA: Interoperability Framework for Linked Data-based Applications

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Information & Knowledge Management

eingereicht von

Lukáš Kavický
Matrikelnummer 00827077

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao.Univ.-Prof. Dr. Stefan Biffl
Mitwirkung: Marta Sabou, PhD

Wien, 24.10.2017

(Unterschrift Verfasser) (Unterschrift Betreuer)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

ILDA: Interoperability Framework for Linked Data-based Applications

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Information & Knowledge Management

by

Lukáš Kavický
Registration Number 00827077

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.-Prof. Dr. Stefan Biffl
Assistance: Marta Sabou, PhD

Vienna, 24.10.2017

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

_____ (Ort, Datum)

_____ (Unterschrift Verfasser)

Acknowledgements

I would like to use this opportunity to express my gratitude to those who helped me during the writing of this thesis.
I want to express my sincere thanks to my advisors, Ao. Univ.-Prof. Dr. Stefan Biffl and Marta Sabou, Ph.D. for their valuable support and advice. I also want to thank Mag. Fajar Juang Ekaputra for his insightful feedback, comments, and suggestions.
I would not be able to finish this thesis without the support of my family and especially my wife, Dorka. I am very thankful for them and their support.

Abstract

The rapid growth of the Web of Data, a global network of interlinked data sets based on Linked Data (LD) principles and technologies, along with the developing sophistication of the applications based on Linked Data increases the potential to solve complex real-world tasks using Linked Data-based applications.

The ever-changing and fast-paced nature of today's work but also personal life calls for tools adapted to the specific user's needs. The semantic nature of Linked Data enabling the applications to "understand" the content of the data can assist in such adaptation. The existing adaptation approaches in the context of LD-based applications introduce different types of adaptability, such as enabling rapid development of adapted tools, promoting in-application adaptivity, or enabling the users to create their own applications – the mashups – visually. However, the existing approaches have their limitations, mainly in the areas of reusability of existing functionality and flexibility of interaction sequences, and cannot be easily combined.

In this thesis, we introduce the interoperability of stand-alone LD-based applications – a complementary adaptation approach integrating existing LD-based applications – enabling execution of workflows spanning over multiple tools and creation of application compositions. We employ a literature study of existing adaptation approaches, a conceptual analysis resulting in a conceptual framework mapping the required and optional functionality of the interoperability solution – the ILDA (Interoperability of LD-based Applications) framework, and a meta-survey of the LD-based applications concentrated on the applications' features affecting their integrability into the ILDA framework. Based on our findings we design the ILDA framework – a blueprint of a solution for interoperability of LD-based applications. The feasibility of the ILDA framework is shown by positive results of feasibility assessment of created ILDA framework-based prototype in specific use cases.

Keywords: interoperability, Linked Data, adaptability

Kurzfassung

Das schnelle Wachstum des Webs of Data, eines globalen Netzwerks miteinander verknüpfter Datensätze auf Basis der Linked Data (LD) -Prinzipien und -Technologien, sowie der steigende Entwicklungsstand der Linked Data-basierten Applikationen erhöhen das Potenzial komplexe reale Aufgaben mittels der Linked Data-basierten Applikationen zu lösen.

Die sich ständig verändernde und hektische Natur der heutigen Berufs- sowie Privatleben erfordert Werkzeuge, die den Bedürfnissen des Benutzers angepasst sind. Die semantische Natur der Linked Data, die den Applikationen den Inhalt der Daten zu „verstehen" ermöglicht, kann bei solchen Anpassungen helfen. Bestehende Anpassungsansätze im Kontext der LD-basierten Anwendungen führen unterschiedlichen Arten der Adaptabilität ein, z. B. ermöglichen schnelle Entwicklung angepasster Tools, fördern interne Adaptivitätsmechanismen, oder ermöglichen den Benutzern eigene Applikationen – die Mashups – einfach zu erzeugen. Die bestehenden Ansätze haben jedoch ihre Beschränkungen, hauptsächlich in den Bereichen von Wiederverwendbarkeit der existierenden Funktionalität und Flexibilität der Interaktions-sequenzen, und können nicht leicht kombiniert werden.

In dieser Arbeit führen wir die Interoperabilität von eigenständigen LD-basierten Anwendungen ein – ein komplementärer Anpassungsansatz der existierende LD-basierte Applikationen integriert und Ausführung von Workflows, die sich über mehrere Applikationen erstrecken, sowie Erzeugung von Applikationskompositionen ermöglicht. Wir betreiben eine Literaturstudie konzentriert auf bestehende Anpassungsansätze, eine Konzeptanalyse führend zu einem konzeptionellen Rahmen, der erforderliche sowie optionale Funktionalität der Interoperabilitätslösung – des ILDA (Interoperabilität der LD-basierten Applikationen) Frameworks – erforscht, und eine Metaanalyse der LD-basierten Applikationen konzentriert an der Applikationseigenschaften, die die Integrierbarkeit der Applikationen in das ILDA Framework beeinflussen. Basierend auf unseren Erkenntnissen entwerfen wir das ILDA-Framework – ein Set der Richtlinien für die Entwicklung von Interoperabilitätssystemen für LD-basierten Applikationen. Die Realisierbarkeit des ILDA-Frameworks zeigt sich in einer positiven Bewertung der Anwendbarkeit des erstellten ILDA Framework-basierten Prototyps in konkreten Anwendungsfällen.

Schlüsselwörter: Interoperabilität, Linked Data, Adaptabilität

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Motivation

Almost twenty years have passed since the publication of the Semantic Web roadmap introducing the core concepts of the Semantic Web by Berners-Lee [1], and more than sixteen years since the vision of the Semantic Web was presented do broader audience by Berners-Lee et al. [2], where the authors have outlined a future with a decentralized web of interconnected self-describing data, and with applications capable of comprehending the data and performing complex sophisticated tasks using that data.

During the years after the introduction of the Semantic Web, the part of the vision concerning the semantic data has gradually become a reality. The family of semantic technologies has been standardized [3], and the principles of publishing data in the form of Linked Data (LD) formulated [4]. Especially in the recent years, we can observe a dramatic increase of the datasets published as Linked Open Data (shown in Figure 1.1). The data comes from different domains: geography, government, life sciences, linguistics, media, publications, social networking, user-generated data and others [5].

While the advances in the publication of Linked Data are clearly visible, the second part of the Berners-Lee's vision [2] concerning the emergence of applications capable of solving complex problems using the semantic data has advanced at a slower rate. Although Bernstein et al. emphasize the advance of deployment of semantic technologies in specific domains (for instance, widespread of semantic markup of web pages, knowledge graphs used by major web companies, employment of Linked Data by major libraries, museums and media, and support for semantic technologies in commercial database management systems) [6], many other authors highlight the difficulties the users are facing when using the Linked Data-based applications. Examples of such difficulties are applications expecting the users to understand technical concepts and terms (e.g., *URI*, *SPARQL endpoint*) [7],[8],[9], and providing visualizations corresponding to the underlying data structures (graph representation), not according to the user's needs [10]. This may be caused by the facts that many applications remain only in the state of a research prototype, and are targeted mainly at tech-users [9].

However, promising advances in this area could be observed recently. The researchers are concentrating more on the improved usability of the Linked Data-based applications [8],[11]. Novel visualization approaches aim to provide improved visualizations of the data

[12]. A new type of tools is dedicated to the exploratory search [13]. The tools are getting better in their ability to assist in solving real-world problems. An example of this trend is the Aemoo, an exploratory search tool outperforming traditional search approaches in some tests [14]. LD-Reactor, a framework for building Linked Data-based applications, is built using the state-of-art web technologies and has the ambition to be used as a framework for real-world applications [15]. Other promising advances come from the domain of data querying [16], semantic mashup frameworks [17], and from many more areas.

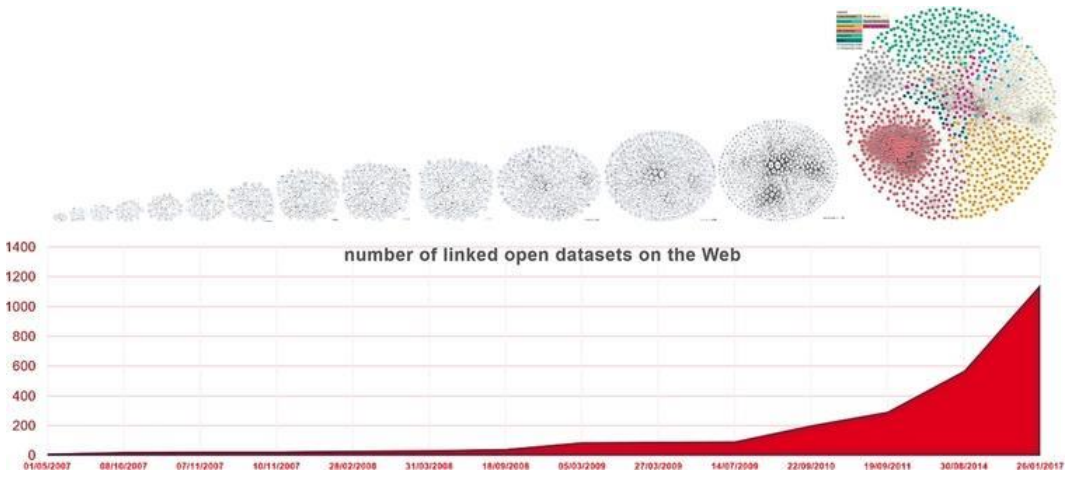

Figure 1.1: The Growth of Linked Open Datasets as Visualized in [18] (Using Data from [5])

With the gradual maturing of the technologies, more and more approaches aim to solve complex real-world problems using Linked Data: assisting the researchers in drug discovery [19], promoting better collaboration of scientists [20], applying semantic technologies in corporate intranet environments [21]. In the field of visual analytics, the use of semantic technologies can lead to significant decrease of the *analytical costs* associated with performing the analysis [22].

1.2 Problem Description

With the growing sophistication of the tools and their applicability in real usage scenarios, multiple authors have described the need to create flexible adaptation mechanisms of the systems to the specific requirements implied by the solved tasks and user preferences. Bakshi & Karger call for tools "*that can easily adapt to evolving user and task needs by working equally well with multiple, unanticipated types of information fragments as they become available*" [23]. Burkhardt et al. warn before overstraining of the users with too much information and call for approaches assisting the user during the usage of the application [24]. Dadzie and Rowe [9] note that *"visualizations have to be tailored to specific tasks, and effectively support users in the performance of these tasks."* Lebo et al.

describe the need for systems enabling flexible transitions between the LD-based tools recognized in the visual analytics field [22].

In the context of Linked Data-based applications, the adaptation need is addressed by various strategies: (a) development frameworks for Linked Data-based applications [25],[26],[27], (b) applications with adaptive UI mechanisms [15],[28],[29], (c) semantic mashup frameworks [17], and (d) applications with simple built-in interoperability mechanisms [30],[31]. While analogical approaches to the adaptation need exist outside of the Linked Data context, the semantic nature and shared vocabularies of Linked Data contribute to the ability of the LD-based adaptation strategies to adapt the applications to the changing requirements implied by solved tasks more efficiently.

The development frameworks are enabling the adaptation by providing reusable components for rapid development of task-tailored applications. In contrast to the analogous approaches (i.e., component-based software engineering [32]) targeted at increasing the development's effectivity outside the Linked Data ecosystem, the semantic nature of Linked Data enables creation of components and templates specifically tailored to the format but also content of the data (e.g., a template visualizing contact information of a person).

The applications with adaptive mechanisms are able to automatically adapt their user interfaces according to the visualized data or collected contextual information, such as user history, explicit user preferences, and device characteristics. The LD-based adaptive applications have again an advantage in comparison with their non-LD-based counterparts (mapped by Nazemi [33, pp. 147–163]) that the well-defined semantic relations of concepts in Linked Data allow applications' adaptation processes for better processing of heterogeneous data [11].

The LD-based mashup frameworks enable the users to easily create their own task-adapted applications by selecting appropriate widgets representing blocks of functionalities (such as data retrieval, processing, and visualization) and creating links between the widgets in a visual mashup application editor. Even in this type of adaptation strategy, the Linked Data contributes to the reusability of widgets by providing common syntactical and semantical model of the data the widgets can process and output, what also lowers the difficulty of creating wiring between the widgets for the users (e.g., not allowing invalid wirings) [17].

The central concept of the applications with simple interoperability mechanisms is the *interoperability* defined as "*a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal*" [34]. In the case of applications with simple interoperability mechanisms, the adaptation happens over an ecosystem of the application with interoperability mechanisms (source application) and multiple linked applications (target applications), which can be used together to create adapted workflows the user could utilize. The tools with interoperability mechanisms enable definition of actions – transitions to other applications based on the currently visualized data. The list of actions recognized by the application can be edited by the developer, and the actions are offered to the user based on criteria specified in the definition of actions (e.g., an action should be offered only in the context of a specific type of data). The transitions are

enabled by the interoperability of source and target applications based on shared data vocabularies and usage of Linked Data.

However, all of the adaptation strategies have also limitations in their abilities to adapt to the changing requirements. The development frameworks do not allow the reuse of functionality implemented in a different form than as a component of the particular framework. The adaptive applications provide only limited options for adaptation outside of the built-in set of adaptive features. While being a very flexible approach to the task-oriented adaptability, the semantic mashup frameworks require the functionality to be packaged in the form of widgets (e.g., following the standard for widgets by W3C[1]). Existing functionality not provided in the form of widgets cannot be easily used in the mashups. The applications with simple interoperability mechanisms can combine functionalities of multiple tools by enabling transitions to other tools, but they are limited to workflows with the only single transition between the tool implementing the transition functionality and linked tools.

Lukovnikov et al., the authors of the tools with interoperability features, suggested that mechanisms similar to their Triple Action Framework enabling the interoperability of Linked Data-based tools could *"become a standardized framework for human interaction with data across the Semantic Web in the future"* [31]. This thesis aims to explore, how the concept of client-side Linked Data-based interoperability of web applications can be further extended and generalized to evolve in the direction of such standardized framework and to contribute to the task-oriented adaptability of environments composed of multiple interoperable Linked Data-based applications. Such standardized framework could not just assist the users in interaction with data and in the selection of the right actions (i.e., tools), but could also enable different kinds of workflows and compositions integrating existing Linked Data-based tools and reusing their functionality in various contexts.

Based on the analysis of concepts from various research areas as well as from the other task-oriented adaptability strategies, we introduce the ILDA (Interoperability of Linked Data-based Applications) framework specifying the components and ways of their operation of interoperability systems, and providing guidance in the development of interoperability systems (ILDA systems). In simple terms, the ILDA system display the integrated application (or applications), collects outputted data from the applications, and either automatically transfer the data to other components or preconfigured other applications (e.g., enabling the coordinated views [35] between applications), or enable the user to select actions (i.e., transitions to other integrated applications and providing them with selected data) based on the selected data. This approach to the adaptation need is complementary to other adaptation strategies (except for the simpler interoperability tools), since the adapted or adaptive tools can be integrated into the ILDA systems, or even could facilitate the ILDA framework internally to overcome the limitations of the specific technological setup (e.g., potential mashup framework enabling to combine the widgets with existing LD-based applications).

[1] https://www.w3.org/TR/widgets/

1.3 Research Questions

The main **research question** we address in this thesis is:

Is it possible to introduce an interoperability framework for standalone LD-based web applications as a means contributing to the task-oriented adaptability of Linked Data-based application environments?

The concept of a framework enabling interoperability has been already proposed by multiple authors. Lukovnikov et al. see their Triple Action Framework as a basis for future extension and generalization leading to creation of a standardized framework for interaction with Linked Data [31] Lebo et al. describe the semantic descriptions of applications' capabilities as a basis for future systems enabling discovery of tools based on the specified type of data and for the reduction of costs (in terms of effort) associated with performing analysis using multiple tools in the visual analytics domain [22], Atemezing & Troncy envision the interoperability of applications visualizing Linked Data as an analogy to the cloud of Linked Open Data (LOD), and call it *Linked Open Visualizations (LOVIZ)* [36].

However, till now no such interoperability framework has been created. As we will show in Chapter 3, there are multiple challenges associated with creation of such universal interoperability framework, such as the requirement to operate within the security restrictions posed by modern web browsers (specifically the rules concerning the cross-origin documents) [37], enabling dynamic addition of new applications, supporting the various data input interfaces of existing applications, and overcoming the "chicken-egg" problem in relation to the potentially integrable applications: the framework needs a set of existing integrated applications for its operation, but without the existence of working interoperability framework the applications will not implement the functionality required for the communication with the framework.

We explore how such interoperability framework (ILDA framework) can be made, how it can integrate even existing Linked Data-based applications, and how the interoperability systems can contribute to the task-oriented adaptability of Linked Data-based multi-application environments.

To be able to answer the main research question, we have defined five sub-questions concentrating on the different aspects of the interoperability framework we need to take into account:

Research Question 1: *What are the existing strategies to address the task-oriented adaptation need in the context of Linked Data-based applications?*

To understand how the ILDA framework can contribute to the task-oriented adaptability of Linked Data-based systems alongside with existing adaptation strategies, we investigate how the existing approaches are targeting the adaptation need, how they could be supported by the complementary interoperability strategy and which of their adaptation concepts can be adopted for the ILDA framework.

Research Question 2: *What are the required and optional functionalities of the ILDA framework and ILDA systems based on the ILDA framework which enable the interoperability of Linked Data-based applications and contribute to the task-oriented adaptability?*

Since there are no similar approaches to interoperability of Linked Data-based applications, we identify the concepts and functionalities required for the operation of the ILDA framework and suggest potential advanced functionalities of the ILDA systems which can further improve the task-oriented adaptability of interoperable application environments and support users in their tasks.

Research Question 3: *Which of the existing Linked Data-based applications could be integrated into ILDA systems?*

The essential part of the interoperability system is the set of integrated applications. To find out how the applications could be integrated into the ILDA systems, and how the ILDA system can support their existing data input interfaces, we observe the existing Linked-Data-based applications in action.

Research Question 4: *How can the ILDA framework support the creation of task-adapted ILDA systems?*

We explore how the ILDA framework can support the utilization of the Linked Data-based applications' interoperability concept in various types of task-adapted systems featuring the creation of workflows over multiple applications and application compositions.

Research Question 5*: Is the ILDA framework feasible, i.e., applicable in practice?*

Our proposed interoperability strategy to task-oriented adaptation relies on the integration of existing applications and on a combination of concepts enabling transitions between applications and providing support to the user in the process. Since there are no related similar approaches to the interoperability of applications, we assess the feasibility of the approach on an implemented prototype of the ILDA system.

1.4 Methodology

To introduce the novel task-oriented adaptation strategy based on a general interoperability framework for Linked Data-based applications, we have conducted following research methods:

Literature research: We investigate the existing task-oriented adaptation approaches and categorize them based on the strategy they employ to attain the adaptability. Coming from our findings, we perform a gap analysis targeted at the identification of how the interoperability approach can contribute to the adaptability of Linked Data-based application environments. The outcomes of this method are described in Chapter 2 and answer the Research Question 1.

Conceptual Analysis: We analyze the concepts required to enable the operation of interoperability systems or contribute to their ability to support the users. The sources of the analyses are fields related to the interoperability in the context of Linked Data and alternative adaptation strategies recognized in the literature research. The analysis is contained in Chapter 3 and answers the Research Question 2.

Survey: We conduct a meta-survey of Linked Data-based visualization and exploration applications based on recent surveys. We observe the properties of the applications relevant to the integration into the interoperability system. The survey is contained in Chapter 4 and is answering the Research Question 3.

Design of the ILDA interoperability framework: We design the modular ILDA framework enabling the creation of various systems based on the Linked Data-based applications' interoperability concept. The description of the framework is contained in Chapter 5 and answers the Research Question 4.

Proof of Concept: We propose test targets aiming to assess the feasibility of the ILDA framework by evaluating the fulfillment of the core functionality of an ILDA system. Based on the test targets we design test scenarios and perform them on an implemented prototype. The proof of concept is described in Chapter 6 and answers the Research Question 5.

Table 1.1 summarizes the relation of research questions, research methods, and chapters where the outcomes are described.

Research Question	Method	Chapter
RQ 1	Literature research	Chapter 2
RQ 2	Conceptual Analysis	Chapter 3
RQ 3	Survey	Chapter 4
RQ 4	Design of Framework	Chapter 5
RQ 5	Proof of Concept	Chapter 6

Table 1.1: Relation between Research Questions, Research Methods, and Chapters

1.5 Thesis Outline

The thesis is organized as follows:

In Chapter 2, we discuss the evolution of Linked Data-based application and its relation to the task-oriented adaptation need described by multiple authors. We also review how this need is approached by existing adaptation strategies. Based on the results of the review we propose how the ILDA framework could complement the existing strategies in the research gap analysis.

Chapter 3 is dedicated to the analysis of the concepts applicable to create an ILDA (Interoperable Linked Data-based Applications) system enabling interoperability of Linked Data-based applications. We analyze what is required to enable this kind of interoperability and what additional functionalities such systems could offer.

Chapter 4 contains a survey of LD-based visualization and exploration applications suitable for being integrated into the ILDA system. We define which properties of the applications are relevant for the integration and map the applications contained in recent surveys.

In Chapter 5, we propose the ILDA Framework, a set of architectural and user experience guidelines enabling to create ILDA systems based on reusable components. A prototypical implementation of an interoperability system based on ILDA framework is presented in Chapter 6 alongside with the feasibility assessment of our interoperability approach. The contributions of this thesis and possible future research directions are discussed in Chapter 7.

CHAPTER 2

Related work

In this chapter, we describe current developments in the sphere of LD-based applications (Section 2.1), specifically concentrating on the emerging need for task-oriented adaptability of the applications, and explore existing approaches contributing to the task-oriented adaptability (Section 2.2). We also discuss the identified research gap in the area of integrating existing LD-based applications (Section 2.3).

2.1 Adaptation in the Context of LD-based Applications

2.1.1 Linked Data

Linked Data is a term used for a collection of interconnected datasets on the web following the *Linked Data principles* [4] and embracing the technologies from the Linked Data technology stack [3]. Linked Data is an implementation of the Berners-Lee's idea of Semantic Web [1] (in fact, Berners-Lee described Linked Data as "*the Semantic Web done right*" [38]) enabling to capture the semantic nature of the data. The combination of technologies and principles is a key element in the transition from the Web of Documents to the Web of Data [4],[39] enabling "*complex queries to be answered and traversals to be made through a diverse and semantically rich information network*" [9].

At the most basic level, the Linked Data paradigm is about publishing data in a machine-readable format (RDF – Resource Description Framework[2]) and linking this data to data from other datasets [9]. A *resource* is any concept described by the RDF. The most common types of RDF resources are the *classes* describing types of resources, specific *instances* of the classes, *literals* representing literal values (such as strings and integers), and *properties* describing relations between other resources or between a resource and a literal in form of *RDF statements* called also *triples* (*subject–predicate–object* expressions). A collection of RDF statements is called an *RDF graph* [40]. The resources are in the RDF format identified using the URIs – Uniform Resource Identifiers[3], which are also used to define links to other resources (from the same dataset or different datasets). For a dataset to be a part of the Linked Data, the URIs must be *dereferenceable* using the HTTP protocol, i.e., accessing the URI using an HTTP request (e.g., in a web browser) should provide data about

[2] https://www.w3.org/RDF/
[3] https://tools.ietf.org/html/rfc3986

the concept identified by the accessed URI [4]. As an example, the URI *dbpedia.org/resource/Spain* identifies Spain as a country in the DBpedia dataset and is accessible both using a web browser (then it is redirected to *dbpedia.org/page/Spain*) and in a machine-readable format. Other technologies in the stack enable definition of vocabularies (RDFS[4], OWL[5]), querying of the data (SPARQL[6]), and multiple other supporting concepts. These technologies provide "*a common framework that allows data to be shared and reused across application, enterprise, and community boundaries*"[41].

The growth of the Web of Data is accelerating every year (as visualized in Chapter 1 on Figure 1.1) and its potential to provide value to the users is proliferating too (as we will describe in the next sections), assuming the existence of the right tools.

2.1.2 Evolution of LD-based Applications

The advantages of the Linked Data can fully show up only through sophisticated applications based on Linked Data assisting the users in achieving their targets. In this section, we briefly describe the evolution of such LD-based applications focusing on the application enabling visualization and exploration of Linked Data. These types of applications are well described throughout the last ten years by numerous surveys [42],[9],[43],[13],[12],[11] enabling us to capture the ongoing evolution of the tools. We can also assume that they represent the majority of the LD-based tools since as of 2014 the significant majority (91%) of tools mapped by a study by Hayes et al. [44] contained visualizations of the data (but we were not able to find more recent sources to confirm this trend).

The visualization methods surveyed by Katifori et al. [42] in 2007 could be characterized as "early research approaches." At the time the Linked Data was just born (the term is not mentioned in the survey), so the tools employ no features related to the Linked Data technological stack (e.g., querying, using data from multiple sources). Many of the reviewed tools were employing the graph visualizations tightly coupled to the underlying data structure, what has been criticized as confusing for users already at that time [10]. Other approaches were exploring alternative visualizations techniques (e.g., 3D visualizations) which are not represented in the later surveys anymore.

The contemporary Linked Data-based textual and visual browsers were mapped by Dadzie & Rowe in 2011 [9]. Employing the already standardized part of Linked Data technologies, the applications are capable of providing functionalities such browsing, querying, editing and even publishing of Linked Data. There is an observable growth in the sophistication of the tools. Some of them allow advanced features like finding relations between concepts and adapting the visualizations based on the content of the data. However, the survey points to two central issues of the contemporary state of the applications' landscape: (a) the limited number of existing Linked Data browsers (b) the almost complete absence of support for lay-user (those not having expertise in Linked Data domain). In this context, the authors express the conclusion that "*the uptake of Linked Data by a mainstream*

[4] https://www.w3.org/TR/rdf-schema/
[5] https://www.w3.org/OWL/
[66] https://www.w3.org/TR/sparql11-query/

audience is dependent on its utility to those outside the Semantic Web and Linked Data communities, therefore this lack of support for non-tech-savvy users could inhibit its adoption" [9].

A similar survey of Linked Data-based browsers by Alahmari et al. in 2012 [43] reviewed the same set of applications as Dadzie et al. [9] but concentrated more on the querying and link-creation features of the tools. The authors praised sophisticated functions of some of the tools, but also called for improvements in the areas of survey's focus, and described the need for flexible and easy tools presenting data in an intuitive way [43].

The focus of a survey by Marie & Gandon [13] lies in the newly emerged type of tools – the LD-based exploration systems. The exploration systems aim to help the users in their investigation and learning tasks. The authors described the recent exploration systems (view-based and algorithm based), categorize the tools and describe their task-specific functionalities in relation to the desired effects of exploratory systems and the characteristics of exploratory tasks (as shown in Figure 2.1). Marie and Gandon concluded the survey with the opinion that the "*exploratory search functionalities and systems will constitute a decisive improvement for the future of web search experience and its outcomes*" [13].

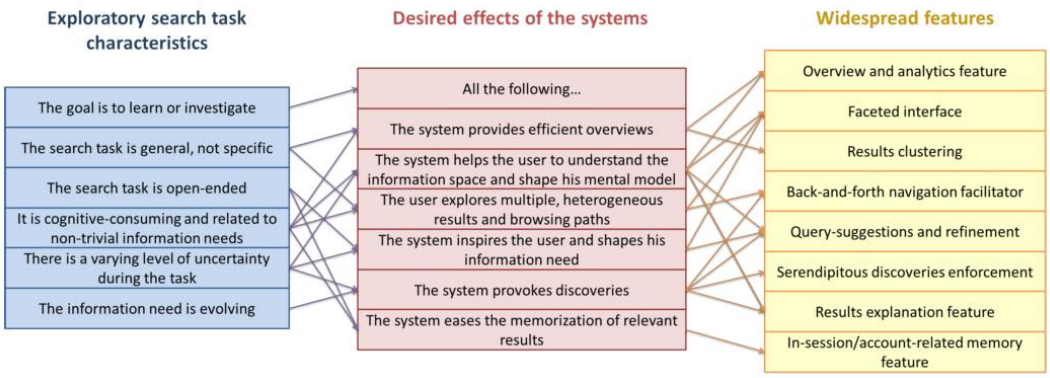

Figure 2.1: Exploratory search – Tasks Characteristics, Desired effects, and Widespread features [13]

In a recent survey of both exploration and visualization applications from 2016 [12], Bikakis and Sellis defined the major challenges faced by the applications as (a) large size and the dynamic nature of data, and (b) variety of tasks solved using the applications and of users' preferences and skill levels. The reviewed tools employed multiple approaches to handle the variety of tasks and users including customization of the user interface and experience according to the type of data, user preferences, environment settings (e.g., screen resolution) and also provided recommendations of visualizations but were not able to handle large datasets. The authors expect future tools to "provide more sophisticated mechanisms that capture users' preferences and assist them throughout large data exploration and analysis tasks" [12].

The very recent review (2017) of state of the art visualization solutions of Linked Data by Dadzie & Pietriga [11] described the latest advances in the field. All of the reviewed tools

provide a rich set of functionalities, and with one exception all are targeted at lay-users or domain-experts (i.e., not the experts on Linked Data). The authors expect further improvements of the visual experience and better orientation at the communities of practice and cooperation with them in the development of the tools.

The ten years of the Linked-Data based visualization and exploration systems' evolution can be summarized followingly: There are still many issues and obstacles to overcome (e.g., the technical barrier for non-tech users is still high [11], the tools are having issues with the scalability [12]), but the sophistication and functionality of the tools grew rapidly over the years, and the tools are getting better in their ability to assist in solving real-world problems. An example of this trend is the Aemoo exploratory search tool, which has outperformed traditional search approaches in some user tests [14].

2.1.3 Task-oriented Adaptation of LD-based Applications

With the growing sophistication of the tools and their applicability in real usage scenarios, multiple authors have described the need to introduce flexible adaptation mechanisms of applications and systems to the specific requirements implied by the tasks and user preferences. Bakshi & Karger [23] describe the need for a solution "*that can easily adapt to evolving user and task needs by working equally well with multiple, unanticipated types of information fragments as they become available.*" Burkhardt et al. [24] describe the need to avoid overstraining the users and call for approaches to support users during the usage of the application. Dadzie and Rowe [9] note that *"visualizations have to be tailored to specific tasks, and effectively support users in the performance of these tasks."* Lebo et al. describe the need for systems enabling flexible transitions between the LD-based tools in the context of visual analytics [22]. Bikakis & Sellis call for tools' internal adaptation mechanisms capable of capturing users' preferences [12]. The calls for adaptation mechanisms share the common goal of providing assistance the users to reach their targets while solving various tasks. They also share the conviction that Linked Data and its ability to describe the semantics of the data play an essential role in the adaptations [45]. However, the authors differ in their emphasizes on the various means of how to achieve the adaptation goal – internal mechanisms of the tools (automatic or controlled by the users) [12],[23],[24], flexible transitions between multiple tools [22], effective creation and modification of adapted tools by application developers [9],[27] or even by the users [17].

Similarly to the different accents of the calls for the Linked Data-based applications' and systems' adaptability, a spectrum of different emphasizes can be observed in the various definitions of adaptation in the context of software systems. The definitions of the *adaptation* and related terms *adaptivity* and *adaptability* are unclear and inconsistent in the literature. Subramanian and Chung offer an overview of the various definitions in the context of software engineering generally [46], Nazemi discusses the definitions in the context of the user interface design [33, pp. 124–126]. In the sphere of software development, the term *adaptability* is used to express the "*ease with which a system or parts of the system may be adapted to the changing requirements"* [47] and is tightly related to the concept of reusable software components of component-based software engineering [32]. When discussing the user interfaces, the term *adaptivity* is commonly used to describe the ability

of a system to automatically modify the user interface based on its internal model of the user or the usage context, while the term *adaptability* is either used for the ability to enable the user to change the interface manually, or connected to the change and tailoring processes before the usage of the application [33, pp. 124–126]. In this thesis, to describe the application's adaptivity (as the automated modification ability) and adaptability (as the ability of the user interface to be manually adapted by the users) we will use the terms *runtime adaptivity* or just *adaptivity* (since there is no confusion in this term) and *runtime adaptability.*

Because of our interest in the broad range of approaches enabling LD-based systems to adapt to the various tasks the users need to solve (including visual analytics, exploratory search, but also other tasks, e.g., the ones described in Section 1.1), for the purposes of this thesis we will use a broad definition of the adaptability and adaptation derived from [47] and including the different emphasizes presented by Subramanian and Chung [46] and Nazemi [33, pp. 124–126]:

> *The **adaptability** expresses the ease with which a system or parts of the system may be adapted to the changing requirements implied by solved tasks and user preferences.*
>
> *The **adaptation** of a system is a manual or automated process providing changes in the user interface of the system, its functionality, and sequences of possible operations targeted at fulfilling the changing requirements implied by solved tasks and user preferences.*

As we have already indicated above, such adaptations can happen during the application's usage – either automatically or be done manually by the user or can target the development or configuration phase of the application's lifecycle. In Section 2.2, we will explore the adaptation capabilities of LD-based solutions in the context of this definition of adaptability.

This thesis aims (described more in Section 2.3) to explore how to support the emergence of multi-application adaptable systems by introducing the interoperability of stand-alone LD-based applications. As we will see in Section 2.3, this interoperability adaptation approach while having multiple advantages is limited in how it can achieve the adaptation. Such approach cannot directly influence internal processes of the integrated applications; it can only facilitate communication between the applications or transitions between them. For the specific sub-type of adaptability achieved by the interoperability adaptation approach, we use the term *adaptability of interaction flows.* The term *interaction flow* is derived for our purposes from the *site flow,* a term commonly used in the user experience domain to describe the tree of possible sequences of user interaction with a web page [48]. We define the interaction flow and its adaptability as follow:

> *The **interaction flow** is a tree of all the possible transitions and interactions between* the visualizations *and other interaction elements of one or multiple LD-based applications*.

*The **adaptability of interaction flows** describes the ease with which a system can change the possible sequences of interaction elements based on the changing requirements implied by the tasks and user preferences.*

The interaction flow's adaptability is a concept related to the *process-oriented adaptation* described by Nazemi [33, p. 397] which sees "*process as repeated sequences of activities and activities as repeated sequences of* interactions" and enables to adapt such sequences.

2.2 Adaptation Approaches in LD-based Applications

In Section 2.1, we have described the evolution of the Linked Data-based application sphere coming to the point where the tools are considered for deployment or are already deployed in real environments. We have discussed the adaptation need described by multiple authors: Linked Data can contribute to the potential of the applications to be efficiently adapted or equipped with internal adaptation mechanisms and by that be tailored to the specific tasks solved with the applications and according to the users' specifics. We have also defined the task-oriented adaptability and related terms.

This section aims to map the existing approaches to the task-oriented adaptability (as defined in Section 2.1) in the context of LD-based applications and application ecosystems. To do that, we have reviewed the adaptability features of tools presented in the recent surveys discussed in Section 2.1.2 [9],[49],[13],[12],[11], with a few additions from our search. We have divided the identified tools into four categories according to the strategy of how they achieve the adaptation: (a*) development-oriented approaches* supporting the developers of Linked Data-based applications to create adapted tools efficiently, (b) tools with *built-in adaptive mechanisms* automatically adapting their user interfaces according to the collected information about the data and usage context, (c) *semantic mashups* enabling the users to create their own applications, and (d) applications with *built-in interoperability mechanisms* enabling to define and execute simple workflows spanning over multiple tools.

We also discuss the Seamlessness theory by Lebo et al. [22] related to the approaches with built-in interoperability features and the concept of interaction flow's adaptation.

2.2.1 Development-oriented Approaches

The development-oriented approaches aim to provide the developers with effective toolkits for rapid development of LD-based applications. The idea of this approach and its importance was described by authors of one of the approaches of this category as follow: "The uptake of semantic technology depends on the availability of simple tools that enable Web developers to build complete applications" [27]. Hayes et al. identified this approach called by them software factories as a major factor in the future development of LD-based applications [44]. The approaches aim to increase the productivity of developers and the quality of the applications by providing reusable components similarly to the standard component-based software engineering approaches [32]. Employing of Linked Data enables

the creation of components sharing data vocabularies, what can improve their reuse potential (e.g., a component showing list of persons usable in various contexts) [45].

The Semantic Web Framework [50] was one of the first attempts to enable the creation of reusable components for LD-based applications. The authors have identified 32 types of component, and categorized them into 7 groups (e.g., Data & Metadata Management, Querying and reasoning), and defined the interfaces between them with the intention to help the developers identify and reuse already implemented components. Having a similar aim, Atemezing & Troncy [36] created classifications of existing LD-based visualization tools and technologies the tools employ. The aim of these classifications is to enable the developers to orientate in the existing approaches and by that foster reuse of existing tools or components. The probably most advanced development framework enabling the definition and reuse of components for LD-based applications is the LD-Reactor [15],[45]. Building on the state-of-art approaches to client-side web application development, the LD-Reactor intends to attract the attention of web developer community and increase the usability of Linked Data applications [45].

Other sub-categories of development-oriented approaches are the templating engines. These approaches enable definitions of templates describing how the data should be visualized. One of the first attempts in this direction was Lena [26], an RDF browser employing Fresnel – an RDF-based templating language [51] to define how the data should be presented. Callimachus [27] is another templating engine based on the idea of defining the templates using standard web technologies (HTML, CSS, JavaScript) in combination with a "reversed use" the RDFa[7] technology. RDFa is usually used to add semantic metadata to web documents, but in Callimachus, it is utilized to define relationships between data and template. Uduvudu [25] is a template engine enabling the application developers to create flexible and even adaptive user interfaces based on a combination of matchers defining which data to select and associated renderers visualizing the data based on the defined templates and context variables (language, user, device).

While enabling reuse of components and templates within various applications, the development-oriented approaches do not allow reuse of components and templates defined in alternative frameworks. Fresnel display vocabulary [51] used in Lena [26] was an attempt to define a universal templating standard but was not employed in later templating engines.

The generalized scheme of how the development-oriented frameworks and template engines approach the task-oriented adaptability is shown in Figure 2.2. The authors of the framework (or template engine) or developers of the framework components provide functionality in the form of components, which can be utilized by the application developers. The user uses the application tailored to the currently solved task. New functionality and the flow of interactions have to be implemented by either the framework developer, component developer or the application developer.

[7] https://rdfa.info/

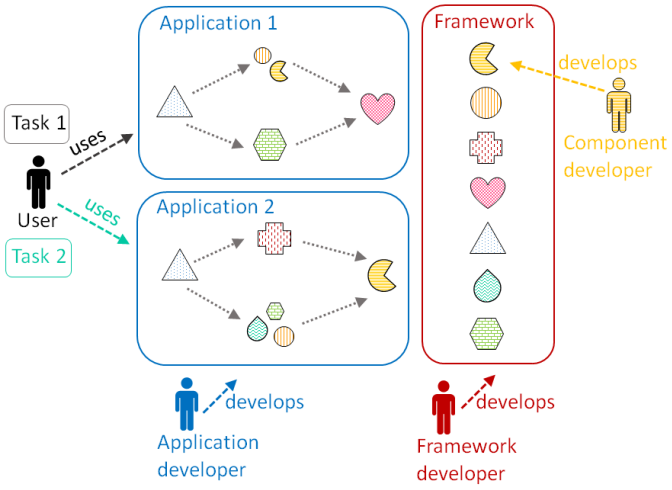

Figure 2.2: Generalized Scheme of Adaptation in Development-oriented Approaches

2.2.2 Built-in Adaptive Mechanisms

Linked Data-based applications with built-in adaptive mechanisms are utilizing adaptive interfaces trying to present the content "*in a manner that best suits individual users' needs*" [52]. The adaptation process can consist of two subprocesses: (a) *content adaptation* deciding what content is the most relevant and (b) *content presentation* deciding how to adapt the presentation of content effectively [52].

Existing LD-based application employ different means to enable the adaptivity of their user interface. The majority of the adaptive tools updates the visualizations according to the type of currently presented data (also called data-aware user interfaces [53]), some of the adaptive tools adjust their interfaces also according to contextual information about the user, the used device, and the language or explicit configuration.

Rhizomer [54] is an example of the simpler adaptive tools capable of generating navigation menus and facets based on the visualized data. Vizboard [28] is an adaptive tool employing the collaborative filtering method to provide the user with the selected visualization based on the collected implicit (usage history) and explicit (user's ratings of the visualizations) user information. SynopsVis [29] is a visualization tool helping the users in the exploration of large datasets by aggregating the data using a technology called hierarchical exploration tree, which enables dynamic adaptation of the hierarchy based on user's preferences. Uduvudu [25], one of the templating engine described in Section 2.21 enables the renderers responsible for the visualization of data to take into account the usage context (language, user, device) when rendering the visualizations. LD-Reactor, another development-oriented framework employs its Adaptation Engine to enable adaptive features in the reusable components [53]. Much more elaborate review of the adaptive approaches (both general and Linked Data-based) is provided by Nazemi [33].

The adaptation in the form of adaptivity mechanisms is not directly task-oriented; the applications adapt their user interfaces based on the presented data and contextual information. Except for the development-oriented approaches with adaptive features [25],[53], this type of tools usually does not explicitly support further adaptations of the tools outside of their built-in adaptive mechanisms, and their functionality cannot be easily reused in other contexts (e.g., combined with the functionality of other applications).

The generalized scheme of the adaptation by applications with adaptive features is shown in Figure 2.3. Based on the content of the presented data and the collected information about the user's context (e.g., history, preferences, and device characteristics) the application adapts its user interface. Different shapes in the image represent various functionalities of the application such as types of visualizations

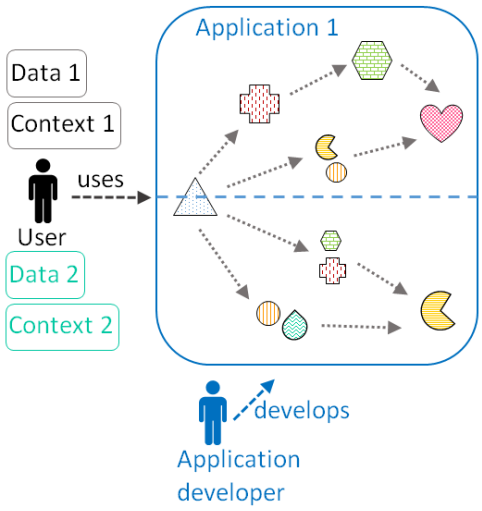

Figure 2.3: Generalized Scheme of Adaptation in Approaches with Built-In Adaptive Mechanisms

2.2.3 Semantic Mashups

Fichter [55] defines a mashup as "a *web application that uses content from more than one source to create a single new service displayed in a single graphical interface."* In the context of Linked Data, several mashup frameworks have been introduced to enable the users to explore Linked Data actively. MashQL [56] and DERI Pipes [57] are the earlier semantic mashups frameworks enabling the creation of mashup applications with a limited set of functionality in a visual editor.

Linked Widgets platform [17] provides a robust solution for the creation of semantic mashups using a graphical editor utilizing the *widget* standard[8] proposed by the World Wide Web Consortium (W3C). A W3C widget is an "*interactive single purpose application for displaying and updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device*" [58]. Open linked Widgets

[8] http://www.w3.org/TR/widgets/

provide three types of widgets: *data widgets* dedicated to retrieving the data from data sources and providing the data to other widgets, *process widgets* taking input data from other widgets and providing processed data to other widgets, and *presentation widgets* presenting the provided data in various ways. The platform provides a set of built-in widgets, and additional ones packaged following the W3C standard could be easily added. The platform aims to "*help users overcome technological barriers of adoption and get in touch with Open Data*" [17].

Mashup platforms based on widget technology are a very flexible type of tool enabling the users to define their own applications depending on the currently solved tasks. They enable reuse of widgets and even whole mashup application in different contexts, but the functionality needs to be deployed in the form of W3C widgets. The generalized scheme of this approach is shown in Figure 2.4. Based on the specifics of tasks the user creates his own mashup application using the visual editor of the mashup platform by connecting the various widgets created by widget developers.

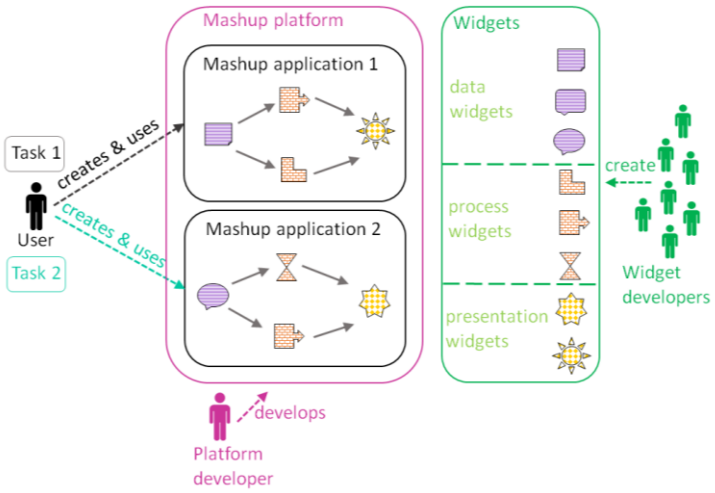

Figure 2.4: Generalized Scheme of Adaptation in Semantic Mashup Frameworks

2.2.4 Built-in Interoperability Mechanisms

Applications with built-in interoperability mechanisms are another task-oriented adaptation strategy introducing interoperability of Linked Data-based applications. Interoperability is defined as "*a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal*" [34]. In the context of stand-alone Linked Data-based web applications, we use the term *interoperability* to describe the ability of an *interoperability* system to facilitate communication between applications (applications used in parallel) or transitions between the applications (applications used sequentially). In both cases, the data outputs of one or multiple applications can be used as inputs to other application or applications.

The adaptation to changing requirements happens over an ecosystem of multiple applications, where applications are offered as actions or "next steps" in the task-solving

process according to data or other contextual information (similarly to the applications with adaptive mechanisms).

Current approaches offer only very limited interoperability features. Some of the Linked Data-based applications enable a transition to another tool, providing the currently visualized resource as an input to the tool. The default DBpedia view used when dereferencing the URIs of DBpedia resources (e.g., *http://dbpedia.org/resource/Spain*, which is when accessed by a web browser redirected to *http://dbpedia.org/page/Spain*) enables to open the currently viewed resource using a predefined set of applications, not adapted in relation to the visualized data. The feature is provided by a component of the DBpedia layout and enables just a one-way single-step transition between DBpedia and the selected tool.

So far the most sophisticated approaches introducing interoperability features are the tools developed by Lukovnikov et al. [30] [31]. The DBpedia Viewer[9] and LinkedData Viewer applications incorporate the Triple Action Framework enabling to define available actions (i.e., transitions to other applications) over specified types of data and their contexts (triples) programmatically (i.e., the definition of an action is a plug-in of the tool). The bind method provided by every action is called for every triple to find out if the action is applicable in the context of that particular triple enabling to provide selections of possible transactions adapted to the visualized data and possibly also to the contextual information. The available actions are shown as small icons on the right side of every data triple.

Existing interoperability approaches have strong limitations of their interoperability features: (a) they enable only single-step transitions – the interoperability must be enabled internally by the tool providing interoperability features, and (b) the set of integrated tools cannot be changed dynamically – they have to be defined programmatically. The interoperability approaches can contribute to the task-oriented adaptation by offering the tools capable of processing the data of selected type (data-aware adaptation mechanism related to the approaches with built-in adaptive features discussed in Section 2.2.2), and by enabling the reuse of stand-alone applications in various or changing workflows. Such workflows are not genuinely *enabled* by the interoperability, manual transitions between the tools (copy & paste or download and upload of data) are possible without interoperability mechanisms, but they require more effort from the users, who also receive no support in the selection of tools in the case of manual transitions.

The generalized scheme of the adaptation by interoperability approaches is shown in Figure 2.5. Application 1 provides the interoperability features (e.g., links to other applications), which could be configured by the developer. Based on the presented data the application shows actions (i.e., links to the other applications) applicable to the data, and enables the user to execute these actions (i.e., make transitions to the other application).

[9] http://ldv.dbpedia.org/

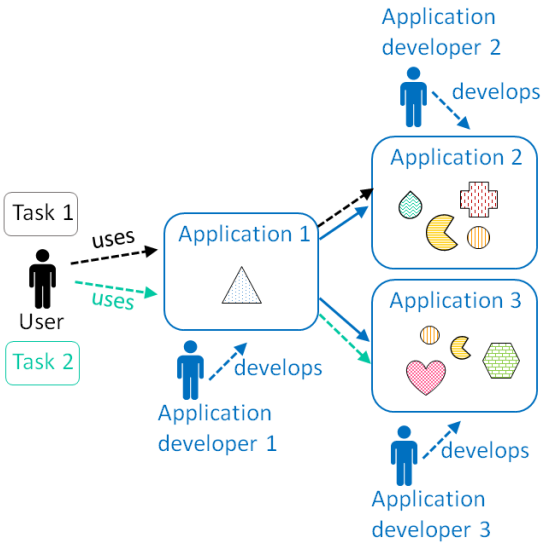

Figure 2.5: Generalized Scheme of the Adaptation in Approaches with Built-in Interoperability

2.2.5 Seamlessness Theory

Seamlessness theory by Lebo et al. [22] discusses in the context of visual analytics domain how the *analytical costs* (in terms of person-hours, lines of code, and commit frequencies) associated with performing analysis using multiple analytical tools can be reduced when the involved applications follow a set of guidelines proposed by the authors. The majority of the guidelines are based on the introduction of Linked Data-based representations of the inputs, outputs, and descriptions of the tools.

The authors describe different kinds of *munging activities* – operations required to enable transfer of data between tools and perform a theoretical cost analysis of example analytical scenarios based on the defined *seamlessness metric*. The analysis shows how the costs associated with performing analysis could be reduced when the involved analytical tools score higher in the proposed *five-star rating* assessing the *application seamlessness* (we discuss the rating and its relation to our proposed solution in Section 3.4.3).

The paper also discusses how the utilization of Linked Data for the creation of semantic descriptions of applications' capabilities and as a format for the inputs and outputs of the applications could help to guide the munging by enabling automatic discovery of tools capable of processing specified types of data.

The authors provide no implementation of the discussed concepts, but the contribution of the paper lies in the strong argumentation for the need of supporting the creation of analytical toolchains based on Linked Data technologies.

2.3 Research Gap

All of the approaches discussed in Section 2.2 share the goal of supporting the adaptation of LD-based application to changing requirements of tasks their users solve, but they differ in the means they employ to achieve the adaptation. In this research gap analysis, we look at the approaches from perspective of two concepts associated with adaptation (cf. Section 2.1.3): (a) adaptability as the "ease of change" – how difficult is it to change the existing functionality of the applications or to add new functionality, and (b) reusability – how difficult it is to reuse the implemented functionality in other contexts (e.g., new applications).

The development-oriented approaches offer reusable components, and by that simplify the development of LD-based applications. The components can be configured or extended and utilized to create adapted applications by the application developers. Some of the frameworks and template engines offer support for the adaptive behavior of the components (such as the LD-Reactor [15],[45] and Uduvudu [25]). The components are reusable only in the context of the particular framework, even though there are attempts to enable the creation of more general components or templates, such as the Fresnel display vocabulary [51] or planned utilization of web components[10] in LD-Reactor [45].

The adaptive LD-based applications provide adaptation of their interfaces according to the visualized data and contextual information. Their advantage is that the adjustments are happening automatically, without the need of manual intervention (although in some of the adaptive tools the user is allowed to intervene in the adaptation). Except for the development-oriented approaches with adaptive features [25],[53], the tools with adaptive mechanisms usually do not explicitly support other adaptations outside of their built-in adaptive mechanisms, and their functionality cannot be easily reused in other contexts.

The semantic mashup frameworks enable their users to create their own applications in a visual editor using the widgets created by various widget developers. The users can create applications tailored explicitly to the solved task and even reuse them in other mashup applications. The employment of W3C widgets enables the reuse of the widgets in various contexts (e.g., potentially across multiple mashup frameworks). The usage of W3C widgets is also their drawback – the widget standard constrains the selection of applications, which can be used in the mashups.

The current interoperability approaches offer only very limited application linking features allowing only single-step transitions between tools and only programmatic definitions of the available actions. The definitions of the steps are not reusable outside of the context of the specific tool (e.g., the LD Viewer [31]).

While the interoperability approaches have demonstrated the basic idea of interoperability, they have not exploited their whole potential. Lukovnikov et al. suggest that mechanisms similar to their Triple Action Framework could be generalized and "*become a standardized framework for human interaction with data across the Semantic Web in the future*" [31]. Lebo et al. suggest that the utilization of Linked Data for descriptions of application and as a format for applications' inputs and outputs could serve a basis for future systems enabling tools' discovery [22], Atemezing & Troncy envision a universal solution

[10] https://www.w3.org/standards/techs/components

for interoperability of Linked Data-based visualization systems and consider their work on the classification of existing Linked Data-based tools as a first step towards the creation of an interoperability platform called *Linked Open Visualizations (LOVIZ)* [36].

The interoperability features don't need to limited to just one tool enabling "jumps" to other tools, the interoperability could be provided as a function of a general framework enabling (a) transitions between the applications with the support of users in the selection of applications (similarly to the existing interoperability approaches and suggested discovery functionality described by Lebo et al. [22]), and (b) communication of multiple applications integrated in compositions (similarly to the components of development-oriented approaches or the widgets of semantic mashup frameworks).

Such interoperability framework-based adaptation strategy is complementary to the existing strategies and can be used synergistically with them. It can be utilized to create systems enabling the users to select actions (i.e., combinations of application and its input data) and facilitating the execution of the actions. While the interoperability framework is providing an additional adaptability layer by enabling data-driven and user-selected transitions between applications (as the existing approaches with interoperability mechanisms, but without their limitations), the integrated applications can utilize other adaptation mechanisms.

At the same time, the ability of the interoperability framework to create compositions of multiple applications can be used to enhance the existing adaptation strategies and mitigate their limitations. The development-oriented frameworks could utilize the interoperability approach to enable integration of Linked Data-based applications as their components; the mashup frameworks could enable integration of the applications in the form of widgets.

In this thesis, we introduce the ILDA (Interoperability of LD-based Applications) framework – a first attempt to shape such general interoperability platform. ILDA framework enables the creation of various ILDA systems implementing the transition or composition functionality described above. Although the complementary nature of ILDA framework and ILDA systems enables to utilize the interoperability approach alongside other adaptation strategies (except for the simpler approaches with interoperability mechanisms), for the completeness we also compare the ILDA strategy with other adaptation strategies in Table 2.1.

Figure 2.6 shows the generalized scheme of how the interoperability systems enabling transitions between integrated applications could enable the task-oriented adaptation of the application environments. The various integrated applications and the ILDA system are developed independently. The ILDA system enables the user to select actions (i.e., applications) from the list of possible actions based on their ability to accept the selected data as their inputs. Adaptations of the system could happen at different levels. The user can choose the appropriate application according to his current needs. The ILDA system can adapt the recommendations of applications according to the contextual information similarly to the applications with adaptive features. New functionality can be added to the system by developing a new integrated application.

Category	Advantages	Disadvantages
Development-oriented	The transition-based adaptation in ILDA systems does not require developer intervention.	ILDA systems have no control over the internal adaptation of integrated applications.
Adaptive Mechanisms	Possible interaction flows could be changed in a much more flexible way in ILDA systems (by addition of new tools).	ILDA systems cannot influence the internal adaptivity of user interfaces of integrated applications.
Semantic mashups	The integrated applications in the ILDA system do not need to be deployed as W3C widgets.	The users of ILDA systems have much less influence on the internal processing and visualization of data.
Built-in Interoperability Mechanisms	ILDA systems allow transitions between any integrated applications, descriptions of the applications do not need to be provided programmatically.	Existing interoperability-based tools are probably more intuitive for the users since the ILDA systems brings a new concept of navigation between applications using an interoperability system

Table 2.1: Comparison of the ILDA system with other task-oriented adaptability approaches

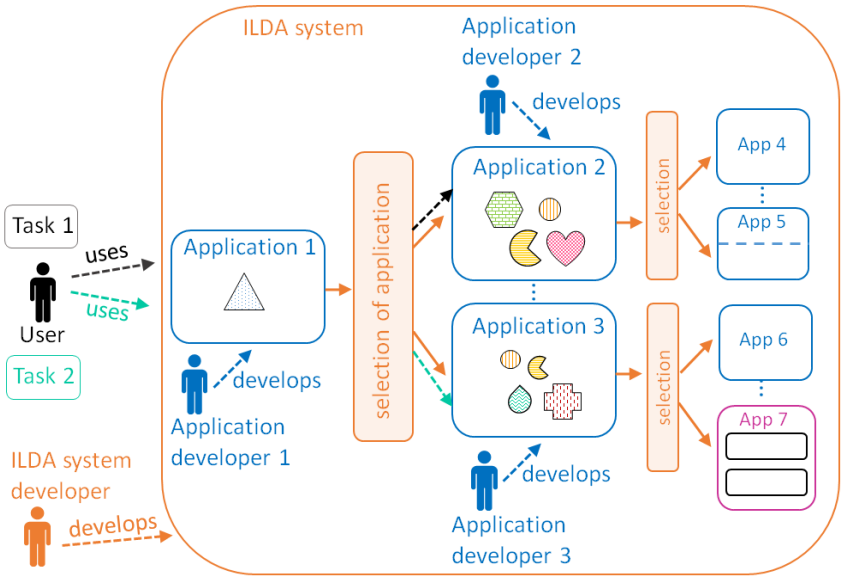

Figure 2.6: Generalized Scheme of the Adaptation in the Form of an Interoperable System Enabling Transitions between Applications

CHAPTER 3

Conceptual Analysis

The interoperability framework for Linked Data (LD)-based applications as identified in the research gap analysis in Section 2.3 is a novel approach targeting the task-oriented adaptation need (cf. Section 2.1.3). While similar approaches have been suggested by multiple authors [31],[22],[36], we are not aware of any previous attempt to implement an interoperability solution of this type. Therefore, in this chapter, we identify the concepts from related research fields and from alternative adaptation approaches, which can be applied in the interoperability solution (ILDA framework) enabling task-oriented adaptation as described in Section 2.3. We are interested in the concepts required to enable the essential functionality of the interoperability systems, but also the concepts enabling potential advanced functionality on top of the essential parts of the systems aiming to provide further support for the users in their tasks.

While the concept of interoperability is novel in the context of LD-based tools, it plays a central role in the Linked Data ecosystem. We start our analysis of the concepts applicable in the context of interoperability framework with the exploration of the relationship between Linked Data and the concept of interoperability in Section 3.1. Since the current approaches introducing interoperability of LD-based applications are very limited in their functionality (cf. Section 2.2.4), in Section 3.2 we explore how researchers approach the issue of interoperability of web applications outside of the Linked Data realm. In Section 3.3 we analyze how the identified concepts of previous sections could be applied alongside with the concepts of alternative adaptation approaches (cf. Section 2.2) to enable the task-oriented adaptation by integrating existing LD-based applications.

We look at the LD-based applications in Section 3.4 with the aim to identify applications' properties related to their ability to be integrated into the interoperability system. We define what properties are required for the integration and what optional requirements the application could meet to enable advanced features of the interoperability systems.

3.1 Interoperability in the Context of Linked Data

In Section 2.2.4 we have already defined for interoperability as "*a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal*" [34]. Kubicek et al. define three layers of interoperability – technical, syntactical and semantical interoperability [59]. The technical interoperability

enables linking of computers or systems and allows the transmission of the data. The syntactic interoperability provides a common "grammar" for the systems, and the semantic interoperability is "*the ability of information systems to exchange information on the basis of shared, pre-established and negotiated meanings of terms and expressions*" [60].

Linked Data (cf. Section 2.1.1) covers all three layers of the interoperability. On the technical level, the transmission of the data is handled by the HTTP protocol (and other protocols from the TCP/IP stack). While there are multiple options for the syntax of the data (e.g., XML/RDF[11], Turtle[12], N-Triples[13], JSON-LD[14]), most of the systems support all of the commonly used formats, and also multiple applications and services offer translations between these serialization types. The semantical level of interoperability is facilitated by the reusable vocabularies defined using the RDFS[15] and OWL[16] languages and additional technologies enabling to define more complex rules and relationships over the data. The authors of vocabularies are encouraged to reuse concepts from existing vocabularies or define relations to them (such as "is similar," "is the same as") [61]. There are multiple common vocabularies available for reuse, such as the Dublin Core metadata vocabulary[17], FOAF (Friend of a Friend) vocabulary[18] describing relations between people, and Geonames[19] for geographical data. The two probably most active research areas in the context of interoperability and Linked Data are data integration and semantic web services.

3.1.1 Data Integration

While Linked Data provides basis for large-scale interoperability of systems [62] and can be used efficiently to *"connect related data that wasn't previously linked, or … to lower the barriers to linking data currently linked using other methods"* [63], there remain important issues to be addressed during the deployment of systems based on common vocabularies and Linked Data. One of them is the data integration characterized by Lenzerini as "*the problem of combining data residing at different sources, and providing the user with a unified view of these data*" [64]. Neish observes that as of 2015 the majority of successful Linked Data deployment projects have been single-institution initiatives [65]. He suggests that the creation of common vocabulary is a complex task and the agreement on the shared concepts can be more easily reached in a single institution. Moser et al. agree with this observation and go even further stating that the need to reach an agreement on the common vocabulary across all project stakeholders (even in a single institution) is a significant limitation in the integration approaches and offer a solution by providing a framework capable of semantic mappings between the data of various stakeholders without the need for a common data schema [66]. The challenges associated with data integration are approached by many more

[11] https://www.w3.org/TR/rdf-syntax-grammar/
[12] https://www.w3.org/TR/turtle/
[13] https://www.w3.org/TR/n-triples/
[14] https://www.w3.org/TR/json-ld/
[15] https://www.w3.org/TR/rdf-schema/
[16] https://www.w3.org/OWL/
[17] dublincore.org/
[18] http://xmlns.com/foaf/spec/
[19] http://www.geonames.org/

research approaches from various domains (e.g., [67], [68]).

For the purposes of the interoperability system, it is important to take into account that the data integration is a complex issue which in many cases can't be easily solved by the creation of a common data vocabulary, and therefore, the system should be capable of enabling integration of applications using heterogeneous data and provide means for data conversions.

3.1.2 Semantic Web Services

A Web service is "a software system designed to support interoperable machine-to-machine interaction over a network" [69]. The standard web services have their interfaces (functions, location, input, and output) described using the WSDL[20] (Web Service Description Language), interact with other Web services based on the rules described using the SOAP[21] (Simple Object Access Protocol) through the HTTP protocol using the XML serialization.

However, the web services technology is capable of describing only the syntax of the interchanged data, not the semantic aspect of the data. This has been shown as a limitation in multiple potential use cases [70]. The Semantic Web Services are an extension of the web services enabling to add a semantic description of the web service and data it accepts as inputs or emits as outputs [70]. Multiple vocabularies for the definition of web services has been proposed (such as OWL-S[22], WSMO[23]), including vocabularies integrating the various web service vocabularies (such as MSM[24] and Linked USDL[25]).

SADI (Semantic Automated Discovery and Integration) is a set of semantic web service design patterns improving the ability of software to discover appropriate services automatically [71]. Lebo et al. in the seamlessness theory (cf. Section 2.2.5) suggested that the SADI framework could be used to discover the applications capable of processing the data outputs from other analytical tools [22].

The web service description vocabularies are modeling similar concepts to those, which will be required in interoperability systems, such as the types of inputs and outputs, and the way of how the service accepts the inputs and emits the outputs [70]. Semantic web services could also be in the future supported as an additional type of integrated systems, and the interoperability systems could allow execution of actions performed by semantic web services.

[20] https://www.w3.org/TR/wsdl
[21] https://www.w3.org/TR/soap12/
[22] https://www.w3.org/Submission/OWL-S/
[23] https://www.w3.org/Submission/WSMO/
[24] https://kmi.github.io/iserve/latest/data-model.html
[25] https://github.com/linked-usdl

3.2 Interoperability of Web Applications

Our proposed interoperability system should enable the integration of web applications, and as such needs to operate as a web application internally in a web browser. The integrated applications, potentially from various sources across the internet, need to be visualized on one web document (i.e., the interoperation system) and be able to communicate with the system. As of 2008, such communication was not possible [71]. Taivalsaari & Mikkonen describe the contemporary state of web browsers as "anti-social"; the browsers lacked capabilities to employ reusable application components and the security mechanisms prohibited any communication between pages or web applications from different origins [71].

The answer to the cross-origin communication issue came in the form of the HTML5 Web Messaging standard[26], providing a messaging system that allows documents and web applications to communicate with each other regardless of their source domain, but is designed in a way which does not enable cross-site scripting attacks [37].

The Web Messaging standard has been already used to create mashups of cross-domain web applications. Matono et al. [72] developed a mashup platform enabling communication between web applications. The system encapsulates the applications by a *wrapper* that enables the application to communicate with the framework. The framework converts the message incoming messages based on an explicit transformation configuration specific for every combination of applications and sends the transformed message to the receiving application. The scheme of the mashup engine's operation is shown in Figure 3.1: Scheme of the Operation of the Mashup Engine by Matono et al. [72]Figure 3.1.

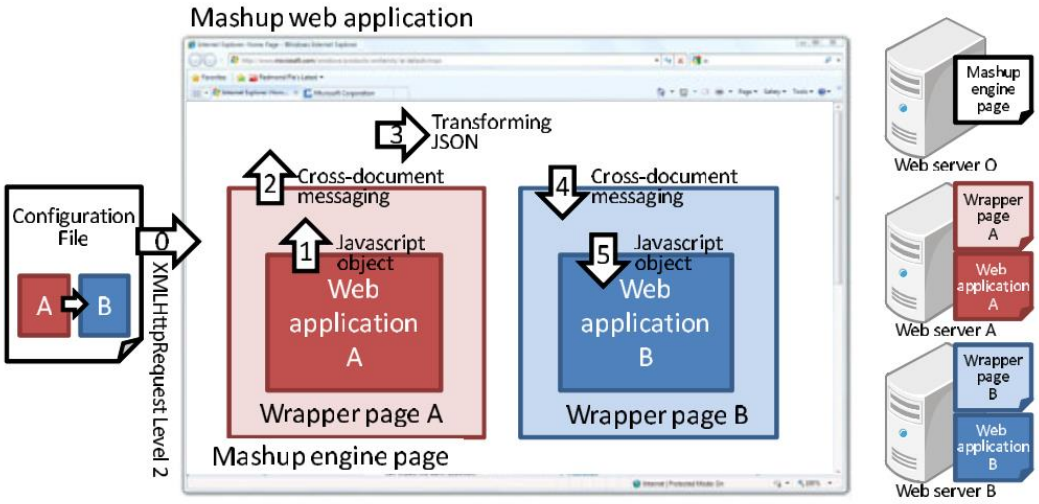

Figure 3.1: Scheme of the Operation of the Mashup Engine by Matono et al. [72]

[26] https://www.w3.org/TR/webmessaging/

While such configuration-based mashup approach enables the reuse the web applications, the need for specific configurations for transformations of data between the applications reduces the flexibility of the mashup creation. The employment of standardized messages and Linked Data to define the interchanged data could enable the creation of interoperable systems for web applications without the need for explicit data transformation configuration.

The communication with integrated cross-origin applications using the HTML5 web messaging and the concept of wrappers adding functionality to the integrated applications will play a central role in the ILDA framework.

3.3 Interoperability System for LD-based Applications

Till now we have observed the development in the sphere of Linked Data-based applications, the emergence of new types of tasks (such as exploratory search) in which the applications aims to support the users, and the recurring calls for task-oriented adaptation of the applications (cf. Section 2.1). We have also explored current approaches answering the calls for adaptation (cf. Section 2.2) and outlined the vision for a complementary adaptation solution based on a framework enabling interoperability of Linked Data-based applications (cf. Section 2.3). In the previous sections of Chapter 3, we have introduced concepts from related research areas with the potential to contribute to the operation of the interoperability framework.

In this section, we analyze how all these concepts can be wired together and formed into the ILDA framework having the goals of addressing the adaptation need and generally supporting the users in reaching their targets in task-solving activities. Considering the novelty of this form of interoperability approach, we analyze what specific targets and functionalities the ILDA framework *needs* to aim to in order to achieve the essential parts of the interoperability and adaptation functionality and what targets the ILDA framework and ILDA systems based on the framework *can* address to further contribute to the achievement of the general goals. The main idea of this analysis is to adapt related targets pursued by the alternative adaptation strategies to the specifics of the interoperability approach (cf. Section 2.3).

From the identified targets of the ILDA framework, we derive the requirements on the ILDA framework and potential ILDA systems contributing to the achievement of the targets. The identified requirements are either describing the essential functionality of the ILDA systems (core requirements) or can further extend the core functionality (optional requirements) to achieve more of the recognized (optional) targets or further contribute to the adaptability and user support. The requirements are inspired by the functionality of alternative adaptation solutions and recognized related concepts. Analogically, we derive specific potential feature of the ILDA framework and ILDA systems from the identified requirements and from recognized related features of alternative adaptation approaches and related research areas. This analytical process is loosely inspired by the analysis of LD-based exploratory systems by Marie & Gandon [13] (graphical summary of their analysis is shown in Section 2.2.1 as Figure 2.1).

We have intended to map the broad landscape of potential future ILDA systems, propose what they could achieve, and design the ILDA framework in a way which will allow future additions of the functionalities to the ILDA systems (cf. Chapter 5). Depending on the specific requirements on the ILDA systems in their deployment contexts, a specific constellation of the targets, requirements, and features should be selected and implemented.

In the following subsections, we describe the recognized targets, requirements and features. For every characteristic, we describe the characteristic and the reason for its inclusion and define its relation to other characteristics and related concepts (from alternative approaches or concepts from Sections 3.1 and 3.2).

3.3.1 Targets of ILDA Framework

The overall goal of the ILDA framework (cf. Section 2.3) and also of the other adaptation strategies (cf. Section 2.2) is to support the user in changing task-solving activities, mainly by providing task-oriented adaptation mechanisms. The various adaptation strategies seek to achieve the overall goal by different means – they are aiming to achieve specific strategy-related targets leading to the fulfillment of our contribution to the overall goal. Since there is no previous attempt to create a general Linked Data-based applications' interoperability platform similar to the proposed ILDA framework, we have explored how the ILDA framework can reach targets similar to the targets of alternative adaptation approaches, and also as a complementary approach, how the framework can support some of the alternative approaches in mitigating their limitations (cf. Section 2.3). The aim of this part of the analysis was to identify the targets the ILDA framework *must* fulfill to enable the core interoperability, and adaptation functionality and the targets the ILDA framework and ILDA systems based on the framework *can* address to further contribute to the achievement of the general goals. During the analysis, we have recognized that the ILDA systems enabling transitions between integrated applications could provide additional support to the users in exploratory search tasks (cf. Section 2.1.2). Therefore, we have also explored how such support can be targeted.

In the rest of this section, we present the targets of the ILDA framework and potential ILDA systems derived from related targets of alternative approaches and exploratory search systems. We describe the targets, their meaning in the interoperability context, their relation to the related systems' targets and the reasons for their selection. The overview of the relations between original targets of related systems and derived targets of the ILDA framework and potential ILDA systems are shown in Figure 3.2. Solid arrows denote strong relations between the targets, dotted arrows more remote relationships.

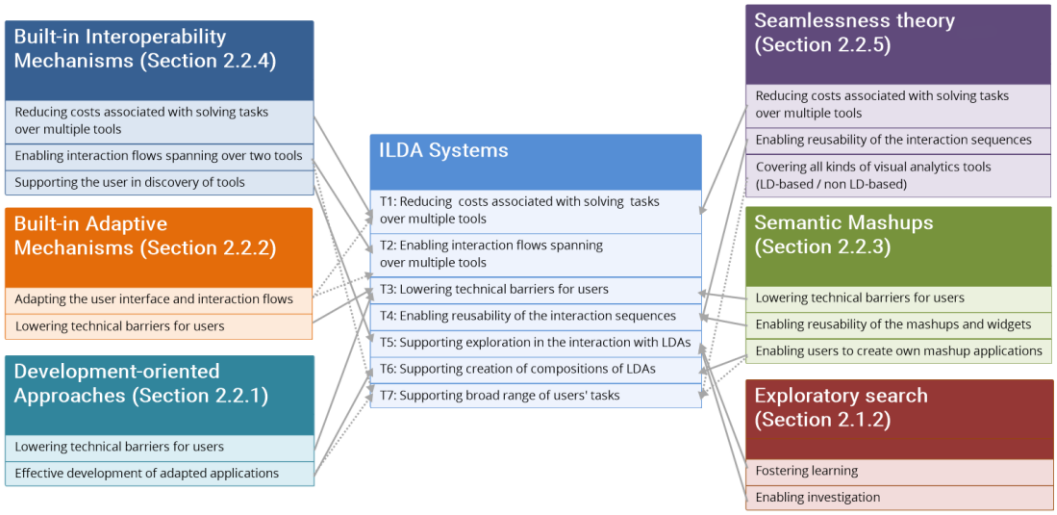

Figure 3.2: Relation between the Targets of Alternative Approaches and the Targets of Interoperability Systems

T1: Reducing Costs Associated with Solving Tasks over Multiple Tools

Supporting the users to in their task-solving activities in order to increase their effectivity is the overall goal of all the adaptation approaches. The approaches differ in specific ways of how they achieve the goal (cf. Section 2.2): Development-oriented approaches enable efficient implementation of adapted applications. Tools with built-in adaptability mechanisms change the user interface, and interaction flows according to the visualized data and contextual information. Mashup frameworks allow the users to create their own mashup application by defining flows of data between widgets in a graphical mashup editor. The interaction flows these approaches enable would not exist without being created by the applications or frameworks.

In contrast to these approaches, the interoperability-based approaches including applications with simple interoperability mechanisms (cf. Section 2.2.4), theoretical interoperability approach by Lebo et al. [22] (cf. Section 2.2.5), and also proposed ILDA framework are not creating new paths, per se. They aim for reducing the *cost* (in terms of effort and cognitive load) of performing a transition from one tool to the another [22]. The interaction flow is not created, it is made more *seamless* to be used [22]: It is possible for the users to navigate between the tools without the interoperability solutions in the form of „manual transitions“ – the users can take the outputs of one tool and provide them as inputs to another tool. Manual transitions could take the form of a well-known “copy and paste” operation (or multiple ones) or downloading and uploading of the data, but in some cases, additional processing of the output could be required (e.g., manual typing of outputs presented as an image, transformations between different formats).

Reducing the costs of solving the tasks over multiple tools (in terms of required effort, cognitive fatigue) in comparison with the “manual” navigation is the primary target of the ILDA framework and ILDA systems. All the other recognized targets are contributing to the achievement of this target from various perspectives.

T2: Enabling interaction flows spanning over multiple tools

The central adaptation concept of approaches with built-in interoperability mechanisms (cf. Section 2.2.2) is the enabling of interaction flows over two application (the application with interoperability features and a set of linked applications). For instance, LD Viewer employs its Triple Action Framework to enable this functionality [31]. One of the core ideas of the ILDA framework is to extend the ability of the existing interoperability solutions and enable interaction flows spanning over basically an unlimited number of tools, which do not need to provide the interoperability features on their own. Enabling interaction flows spanning over multiple tools is alongside the support for application compositions (*T6*) one of the most central targets of the ILDA framework from the technological perspective. It is a prerequisite for the majority of other targets which build on top of this functionality (*T3, T4, T5, and T7*).

T3: Lowering Technical Barriers for Users

The majority of existing LD-based applications still expects the users to understand technical concepts related to the Linked Data technologies, what has repeatedly been described as one of the major obstacles to the broader adoption of the LD-based applications and Linked Data in general [8], [9], [11].

Many of the adaptation approaches are addressing this issue: the development-oriented approaches offer reusable components with fine-tuned components' user interfaces hiding unnecessary technological details [15], [25]; approaches with built-in adaptive mechanisms are also hiding unnecessary technical details [73], [45]. The mashup frameworks are enabling the users to create own tools from bottom up in an accessible and visual way without requiring in-depth knowledge of Linked Data concepts and by that they are overcoming multiple layered-up technical barriers: users not knowing where to find data source, not knowing how to access the located data source, and not being able to perform the required data processing and data integration tasks manually [17].

While the majority of the steps towards lowered technical barriers (e.g., those described by Dadzie et al. [8]) needs to be implemented at the level of the integrated apps, the ILDA systems can contribute to overcoming the barriers supporting the users during the selection of tools and during the transitions between applications.

T4: Enabling Reusability of the Interaction Sequences

Lebo et al. integrates in their concept of *analytical costs* (described in Section 2.2.5) not just the costs associated with performing the analysis (used to derive *T1* of the ILDA systems) but also the *reuse potential* of the analysis – *"the ability to easily reuse and repurpose prior analytical materials"* [22]. Reusability of the widgets is also an essential concept of the Linked Widgets Platform representing the Mashup approaches. The users of Linked Widgets can reuse (a) widgets from different developers in their applications, (b) reuse and reconfigure composed mashup applications created by other users. (c) reuse composed application as a new widget in other applications, and (d) derive new widgets from existing ones [17]. Similarly, the ILDA systems could aim to provide means to enable reusing of the interaction sessions in a new context and by that achieve further reduction of the costs associated with solving complex tasks.

T5: Supporting Exploration in the Interaction with Linked Data-based Applications

One of the types of tasks the users could solve using the ILDA framework are the exploratory search tasks. The functionality of systems supporting users in exploratory search tasks was mapped by Marie & Gandon [13], and we discussed it in Section 2.1.1 (Figure 2.1 shows the overview of their analysis). Even though most of the analyzed *desired effects* of exploratory systems can be achieved only internally in the exploration applications, ILDA systems can support two of the described desired effects. The user could be allowed to "*explore multiple, heterogeneous results and browsing path."* ILDA systems could provide session's history visualizations and enable the user to revisit previously performed actions (pairs of used applications and input data) and possibly take alternative paths (i.e., use the outputs of previous actions to perform other actions than before). The system could "*provoke discoveries" and* offer applications not commonly used by the user (possibly newly added to the system). Even though not applicable in every context, the exploration features could support adoption of newly added tools and assist the user in exploratory tasks.

T6: Supporting Creation of Compositions of Linked Data-based Applications

Enabling effective creation of Linked Data-based applications (LDAs) as compositions of prepared components is the central adaptation mechanism of development-oriented adaptation approaches (cf. Section 2.2). Similarly, enabling the users to create compositions using the provided widgets is the central idea of mashup frameworks (cf. Section 2.2.3). We have also already discussed that the development-oriented approaches and mashup frameworks could profit from the possible integration of complete Linked Data-based applications wrapped in the form of their components or widgets (cf. Section 2.25). Matono et al. [72] have shown how to create compositions of applications using configurations for specific sets of applications (cf. Section 3.2). The ILDA framework could enable a universal type of compositions based on the exchange of Linked Data technologies and potentially contribute to the abilities of development-oriented approaches or mashup frameworks which would adopt the interoperability features.

T7: Supporting Broad Range of Users' Tasks

The aspiration for broad applicability is observable in the majority of the adaptation approaches, what is manifested in the employment of various extensional mechanisms enabling to add functionality (e.g., Uduvudu framework enabling the creation of custom data *matchers* [25], adaptable LD Reactor enabling creation of custom components [15], interoperable LD Viewer enabling to define new integrated apps and when they should be offered [31], the *Linked Widgets* mashup framework supports creation of new widgets [17]). Additional layer of flexibility in the *Linked Widgets* mashup framework is provided by the separation of concerns of the three types of widgets (*data widgets, process widgets,* and *presentation widgets,* described in Section 2.2.3) [17]: the modular architecture enables complex tasks to be divided into *reusable modular functions* represented by the widgets [74, p. 3]. The seamlessness theory applies to any tool and interaction flow in the visual analytics domain [22]. The ILDA framework and ILDA systems also aim to support users in a broad range of tasks both on the level of ILDA framework (enabling the creation of ILDA systems with different sets of functionalities, supporting a broad range of tools), and on

the level of the particular ILDA system's implementation (specific functionalities). Supporting a broad range of users' tasks could be a major factor for the adoption of the technology and justification of the potential costs (in terms of effort or finances) associated with the implementation and deployment of an ILDA system.

Deriving Requirements and Features

We have introduced the seven targets of the ILDA interoperability approach, which were formulated based on the analysis of targets of alternative adaptation strategies and related exploratory search. Based on the targets, we derive the requirements (names starting with the abbreviation *Rq*) for the ILDA framework and ILDA systems (Section 3.3.2) and the specific features of the framework (starting with the abbreviation *Ft*) and systems fulfilling the requirements (Section 3.3.3). A graphical overview of the relations between identified targets, requirements and features is shown in Figure 3.3. The characteristics, which provide the core interoperability functionality are marked as bold (*T1-T3, Rq01-Rq06, Ft01-Ft07*), the characteristics which are specifically targeting the design of the ILDA framework are highlighted (*T7, Rq16, Ft22-23*). The outcomes of the analysis serve as a base for the design of ILDA framework (cf. Chapter 4).

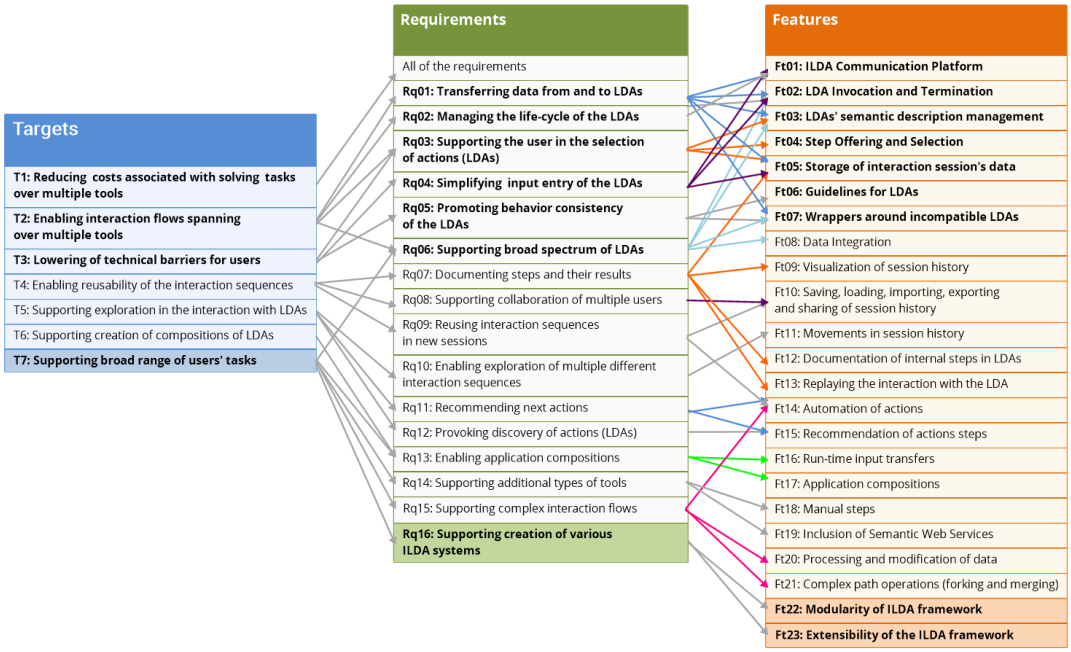

Figure 3.3: Relations between Targets, Functions, and Features of the ILDA Framework

3.3.2 Requirements for ILDA framework

The requirements describe parts of the functionality or properties of the ILDA framework and potential ILDA systems contributing to the achievement of ILDA systems' targets. They describe more concretely what the ILDA framework and ILDA systems should strive to achieve. We have divided the requirements into three categories: (a) *core requirements* which should be fulfilled by the ILDA framework and ILDA systems (at least in a minimal form) to enable the operation of ILDA systems, (b) *optional requirements* describing functions and properties of the ILDA frameworks and systems providing additional contributions to the achievement of targets, and (c) *framework requirements* which should be fulfilled on the level of ILDA framework.

Core Requirements

Core requirements describe the functionalities and properties of ILDA framework and potential systems which must be fulfilled to enable the interoperability of LD-based applications (LDAs) and provide a consistent environment for the users of the systems (and by that contributing to the achievement of the targets *T1* and *T2*).

Rq01: Transferring data from and to LDAs: This requirement describes the basis of the semantic interoperability on the application level – the ability to transfer the outputted data (or its subsets) from one or more applications to other application or applications. To enable communication with a broad set of existing LDAs (*Rq06*), the systems should support existing input methods of the web applications (e.g., HTTP URL parameters, POST parameters, described in [75]). Additional types of data can be transferred through the ILDA communication platform (*Ft01*). Alongside the *Rq02,* the fulfillment of this requirement serves as an enabler for all of the other functionality of the ILDA systems and contributes to the achievement of the enabling of multi-application interaction flows (*T2*).

Rq02: Managing the Life-cycle of the LDAs: All of the approaches to interoperability of web applications (cf. Section 3.2) require a host application to manage the integrated applications contained in frames (HTML's *iframe*). The ILDA system being the host of the integrated applications needs to manage the invocation and termination (*Ft02*) of the applications and possibly communicate them the life-cycle events, e.g., due termination (*Ft01*). All other requirements are dependent on the fulfillment of *Rq02* together with Rq*01,* which together enable multi-application interaction flows (*T2*).

Rq03: Supporting the user in the selection of next step in the interaction flow: When implementing the navigation mode (other ILDA systems' modes are described in Section 3.3.4) the ILDA system should enable the users to select the application used in the following interaction step based on the outputs from the previous step (or multiple steps; cf. *Ft05*)**.** The system manages descriptions of the integrated LDAs (*Ft03*) and offers only the applications accepting the previously outputted data (*Ft04*). *Rq03* is targeted at supporting the users in the selection of actions (i.e., tool combined with provided data) and facilitating the transition between the tools and so reducing the interaction cost (*T1*), enabling multi-application interaction flows (*T2*) and hiding technical details concerning the transferred data (*T2*). *Rq11* is a related requirement describing active recommendations of applications to the user.

Rq04: Simplifying Input Entry of the LDAs: Though not explicitly mentioned by the authors, even the existing interoperability approaches (Section 2.2.4) hide some of the technical details in the transition between tools (e.g., the user does not need to understand the phrase "enter node URI" in the second application – as it is the case in Lodmilla[27]). The same effect should be reached in the ILDA systems: the stored data from previous interactions (*Ft05*) is transferred to the next application (*Ft01* or *Ft02*, based on the supported type of communication) and the user does not need to interact with the default input interface of the tool. This can contribute to the lowering of technical barriers for users (*T3*).

Rq05: Promoting behavior consistency of the LDAs: "Consistency enables people to effectively transfer knowledge to new contexts, learn new things quickly, and focus attention on the relevant aspects of a task."[76] The ILDA systems should try to accomplish the consistency in behavior of links in the integrated application (functional consistency [76]), so that the users could be expecting a standard behavior and should not be surprised by the behavior of the applications (e.g., circumventing the standard behavior of how data is selected). This can be achieved by formulation of guidelines for the integrated LDAs (Ft06) or by the creation of wrappers modifying applications' behaviors (Ft07). This can contribute to the usability of the ILDA systems (T3).

Rq06: Supporting broad spectrum of tools: The framework should enable integration of tools with various types of input methods (see *Rq01,* affecting mainly *Ft02* and *Ft03*) and enable creations of *wrappers* for application not natively supporting the integration into an ILDA system (*Ft07*). Support for tools facilitating heterogeneous vocabularies could be achieved by introduction of data integration features (*Ft08*). Support for a broad spectrum of tools is contributing to the support of a broad range of users' tasks (*T7*) and the realization of multi-application interaction flows (*T2*).

Optional Requirements

Depending on the particular deployment contexts, the fulfillment of the following requirements can contribute to the achievement of the targets of ILDA framework and ILDA systems, but is not required for the operation of systems.

Rq07: Documenting Steps and Their Results: The first step towards enabling reusability of interaction flows (*T4*) is the storage and presentation of the history of steps and their results [22]. Multiple exploratory search tools mapped by Marie & Gandon are also providing similar functionalities [13]. Such documentation could enable either manual reconstruction of the interaction flow, or its more sophisticated reuse (Rq09), and also contribute to the collaboration of multiple users (*Rq08*). The history could be visualized (*Ft09*) and could contain additional information about the steps taken internally in the integrated applications (*Ft12*), which could also be replayed to the user (*Ft13*). The data inputs and outputs could be an important part of the documentation (*Ft05*).

[27] http://lodmilla.sztaki.hu/lodmilla/?

Rq08: Supporting Collaboration of Multiple Users: Different forms of interaction flows' reusability (*T4*) are the reuse of the same interaction session by other users or the collaboration of multiple users on a task [22]. ILDA systems could enable the users to share the workflow with other users (*Ft10*).

Rq09: Reusing Interaction Sequences in New Sessions: Reusability of the interaction flows (T4) could also be supported by enabling the users to load previously stored interaction sessions (*Ft109*). The need for reuse of analytical sequences is described by Lebo et al. [22].

Rq10: Enabling Exploration of Multiple Different Interaction Sequences: The user could be supported to explore the applications and their functions (*T5*) by enabling to move in the history of the interactions and to try to take alternative steps (*Ft11*). The concept is related to the desired effect of exploratory systems labeled "*The user explores multiple, heterogeneous results and browsing paths*" as described by Marie & Gandon [13].

Rq11: Recommending Next Actions: Recommendation of steps is an extension of the user support in step selection (*Fn03*) aiming at the further reduction of task costs (*T1*) by helping the users to overcome issues associated with an overload of choices [77]. Additional functionality of the recommendation could be the support of application discovery (cf. *Rq12*). The ILDA system could contain a recommender component (*Ft15*) providing recommendations of next actions (or even action sequences (*Ft14*)) based on user preferences, histories of interaction flows, or other collected data.

Rq12: Provoking Discovery of Actions (LDAs): Another function targeted at supporting exploration of tools and their functions (*T5*) inspired by the desired effect of exploration systems characterized as "*The system provokes discoveries"* by Marie & Gandon [13], which could influence the recommendation of next steps (*Ft15*).

Rq13: Enabling Application Compositions: ILDA framework should enable the creation of compositions of the integrated applications (*T6*). The particular ILDA systems implementing this functionality need to handle communication with multiple applications and utilize the communication between them (*Ft17*), optionally with the run-time transfers of data (*Ft16*) enabling the application to accept data without the need to reload the application. Application compositions are more described in Section 3.3.4.

Rq14: Supporting Additional Types of Tools: ILDA systems could provide support for additional types of integrated tools. Incompatible types of applications (e.g., desktop applications) could be used in the context of the ILDA system by the concept of manual steps enabling the users to manually download the input data for the incompatible application and then later to upload the outputted data (*Ft18*). Semantic Web Services could also be incorporated into ILDA systems (*Ft19*). Support for additional types of tools would contribute to the support for broad range of tasks (*T7*).

Rq15: Supporting Complex Interaction Flows: The ILDA systems could if required facilitate complex interaction flow operations like forking and merging of the path (*Ft21*), i.e., enable the user to use outputs of a step in multiple tools and potentially use the outputs of such parallel sequences as inputs of a single action and by that "merge" the two parallel sequences. Other features of complex interaction flows could be the automation of steps (*Ft14*) and internal data processing of ILDA systems (*Ft20*). The ILDA framework should

enable the addition of this type of functionalities (*Ft23*). These complex interaction flow functionalities can contribute to support the users in their complex tasks (*T7*).

Framework Requirement

Rq16: Supporting Creation of Various ILDA Systems: Our adaptation approach based on universal interoperability of Linked Data-based applications aims to assist the users in various contexts (*T7*). Therefore, we propose not a single system with a defined set of functionalities, but rather the ILDA framework establishing guidelines for the creation of various ILDA systems implementing different sets of functionalities (fulfilling the optional requirements described in this Section, or additional ones) according to the particular deployment contexts. The ILDA framework (cf. Chapter 5) should be designed in a way allowing reuse of functionality between ILDA systems (*Ft22*) and addition or modification of functionality (*Ft23*).

3.3.3 Features of ILDA Framework

The features are more specific means of how the high-level functionality and properties described by the requirements for the system could be implemented in the ILDA framework and ILDA systems. Similarly to the requirements, the features are divided into three categories: (a) *core features* representing mainly the core functions of the system, (b) *optional features* – more advanced features which could be considered in some implementations of the ILDA system and (c) *framework features* which should be incorporated already in the design of the ILDA Framework.

Core Features

Ft01: ILDA Communication Platform: To enable communication with integrated Linked Data-based applications (LDAs) and by that the essential functions of the system (*Rq01*, *Rq02*), the system needs to provide a standardized communication platform. The platform needs to facilitate communication even with application originating from different domains than the ILDA system, similarly as in the web application mashup approach proposed by Matono et al. [72] (the issue with cross-domain origin is discussed in Section 3.2), and provide support for the existing ways of how the applications expect their inputs (supporting a broad range of tools – *Rq06*). The platform should be used to transfer data to and from the application and enable the exchange of messages required for other features like informing the application about the usage inside of ILDA system (*Ft02*), internal steps taken inside of the application (*Ft12*).

Ft02: LDA Invocation and Termination: During the transition, the current LDA could be informed about the termination (*Rq02*), and the selected one should be started and informed about its usage in the context of ILDA system. The HTTP request for the selected application can contain the inputs for the application (*Rq01*). If some of the application's inputs were not provided, the system should prompt for their manual input (e.g., input boxes, selections).

Ft03: LDAs' semantic description management: An ILDA system needs to maintain an updated store of descriptions of integrated applications. These descriptions are used for the offering of possible steps to the user (*Rq03*) and describe how the data should be provided to

the selected tool (*Rq01*). The data model of the description should allow for extensions, like new input types or additional descriptions of the tools (*Rq16*). The application descriptions are inspired by the descriptions of semantic web services (cf. Section 3.1.2).

Ft04: Step Offering and Selection: Based on the current set of outputted data the system should offer (or recommend – *Ft15*) a list of *available steps* – applications accepting the data of the same types as stored data or its subset. The list is used for the selection of next step (*Rq03*).

Ft05: Storage of interaction session's data: The system needs to store the data provided as outputs from one or multiple tools to enable the action selection (*Rq03*) and assist the user in input entry (*Rq04,* e.g., by selecting which of the stored outputs should be considered in the step selection). After the selection, the stored data should be transferred to the selected tool (*Rq01*). The storage of data could also play an important role in the documentation of the path's history (*Rq07*).

Ft06: Guidelines for LDAs: To enable a consistent experience [76] across the integrated application in an ILDA system (*Rq05*) the application should react consistently when selecting data (e.g., not following a link as outside of the ILDA content, but emitting the data) and clicking on links.

Ft07: Wrappers around incompatible LDAs: Existing applications, which are not compatible with ILDA systems could be extended using a *wrapper* adding required functions (mainly communication with the system) to the applications (*Rq01*). Wrappers can also alter the behavior of the integrated application to promote behavior consistency (*Rq05*). This feature of the system is inspired by the wrappers used to facilitate communication in the cross-domain web application mashup framework [72] described in Section 3.2.

Optional Features

Ft08: Data Integration: To support the integration of tool using heterogeneous data (*Rq06*), ILDA systems could support the data integration processes (cf. Section 3.1.1) used in its deployment context. For instance, some of the integrated application (or services – *Ft19*) could enable translations or enrichment of the selected data; the systems could support the automatic transformation of data based on semantic mappings between the data, similar to the functionality of system described by Moser et al. [66].

Ft09: Visualization of Session history: The path of interactions with integrated LDAs in a session can be visualized for documentation purposes (*Rq07*), and could also enable the user to return to previous steps in the session (*Ft11*). The feature is also known as breadcrumbs and is quite widespread in the LD-based exploration systems [13].

Ft10: Saving, Loading, Importing, Exporting and Sharing of Session History: The ILDA system could provide means to save or export the history of interactions in a session to support the reusability of interaction flows (*Rq09*). The stored interaction flows could be shareable with other users to support the collaboration of users (*Rq08*). Additionally, the users could be allowed to annotate the steps in the history and store the annotations with the history. Sharing of complete information about the performed analysis (including data and taken actions) has been described as an essential concept in the reduction of analytical costs by Lebo et al. [22].

Ft11: Movements in Session History: This feature extends the visualization of history (Ft08) by enabling the users to revisit the previous steps in the interaction flow to foster exploration of the tools (*Rq10*) or to support the creation of complex paths (*Rq15,* e.g., starting a parallel path from a point in the history of interaction). In the context of exploratory search systems, similar features have been described as follows: "*They considerably lower the perceived cost of browsing and consequently encourage the exploration*" [13].

Ft12: Documentation of Internal Steps in LDAs: If supported by both the ILDA system and the integrated application, the history (*Ft09*) could also store the information of internal steps carried out inside the application (emitted by the application in the ILDA communication platform (*Ft01*)). Such information could enhance the documenting function of the system (*Rq07*).

Ft13: Replaying the Interaction with the LDA: The interaction with the application could be recorded (by storing information about the DOM events) and replayed back to the user when revisiting the step in history (*Ft11*), and by that contribute to the documentation of interaction flows (*Rq07*). The recording of DOM events has been researched by Yildiz et al. [78].

Ft14: Automation of Action Steps: The system could enable the user define a meaning of the relation of the action step's inputs and outputs (either inside of the LDA or in the *Navigator*). E.g., the automated step could be configured always to return the largest city of the visualized country. Such automated steps would be similar to the functionality of *processing widgets* of Linked Widgets framework [17] (and a slightly modified mashup framework could be used to create this kind of steps). This could contribute to the reuse potential of the interaction flow (*Rq09*), but could also be used to enable recommendation of whole sequences of steps (*Rq11*).

Ft15: Recommendation of Actions: The system could incorporate a recommender system suggesting the steps (*Rq11*) to the user based on the specific characteristics of the applications (Ft03) and additional data, such as user preferences (implicit or explicit), and history of interaction flows. The recommendation of actions is a feature inspired by the visualization recommendation functionality of some of the adaptation approaches with built-in adaptive mechanisms described in Section 2.2.2 (e.g., Vizboard [28]).

Ft16: Run-time input transfers: If supported by the application, the system could make it possible to send inputs to the application even after it has been started. The feature could be necessary for some forms of application compositions (*Rq13, Ft17*), where the applications would flexibly react to the emitted outputs of other applications. A similar concept was used in the web application mashup approach proposed by Matono et al. [72].

Ft17: Application compositions: The system could enable the creation of compositions of multiple simultaneous applications and facilitate communication between the applications. The creation of compositions is a feature similar to the ability of development-oriented adaptation approaches to compose applications from existing or newly created components (cf. Section 2.2.1) or the ability of semantic mashups frameworks to enable the user to compose mashup applications from provided widgets (cf. Section 2.2.3). The functionality can be used to enable those alternative adaptation approaches to integrate ILDA-compatible applications. Application compositions are more described in Section 3.3.4.

Ft18: Manual steps: The system could provide limited support for incompatible applications (*Rq14*). The user could be allowed to download the outputs of previous steps, process them manually or to use an incompatible application (e.g., a desktop application), and upload them back to the navigator.

Ft19: Inclusion of Semantic Web Services: Outside of the integrated LD-based applications (discussed in 3.4), the system could enable the integration of Semantic Web Services and by that widen the spectrum of supported tools (*Rq14*), and play a similar role in the ILDA systems as the process widgets of Linked Widgets mashup framework [17]. For instance, a service could be able to enrich an RDF graph with additional data, or even order selected product.

Ft20: Processing and modification of data: The system could provide means to process the data internally (e.g., making collections from the data, editing the data) without the need of an integrated application. The feature could contribute to the ability of the system to enable complex interaction flows (*Rq15,* e.g., combining results from multiple steps and using such collection as input for the next step).

Ft21: Complex path operations (Forking and Merging): For some tasks-solving processes, it could be advantageous for the users to be able to process the same data in parallel using multiple applications. Such parallel threads could be then again merged and their outputs provided as inputs to a single application. This feature would contribute to the support of complex interaction flows (*Rq15*). A similar feature is contained in some of the mashup frameworks: In the *Linked Widgets* platform the results of data or processing widgets could be linked to multiple processing or presentation widgets, and multiple data outputs can be merged using different kinds of *Merger* widgets [74].

Framework features

Ft22: Modularity of ILDA Framework: Creation of various ILDA systems (*Rq16*) adapted to the users' needs (*T7*) can be supported by utilizing a component-based architecture wrapping functionalities into coherent modules. Creators of ILDA systems could then reuse some of the components in multiple systems.

Ft23: Extensibility of the ILDA Framework: Extensibility is another needed feature of the ILDA framework enabling the creation of newer ILDA systems with new sets of functionalities (*Rq16*) while preserving the architecture, interfaces, and data models of the ILDA framework and sustaining the reusability of framework's components (*Ft22*). Extensibility is a major feature of alternative adaptation approaches and has been more discussed in the description of target *T7* in Section 3.3.1.

3.3.4 Modes of Operation

The ILDA framework can be used as a basis for various types of ILDA systems. One of the characteristics in which the systems could differ is their *mode of operation.* Mode of operations describes the art of how the integrated Linked Data-based applications: as steps in workflows (*T2*), as multi-application composition's components (*T6*). In the following text, we describe possible variants of the modes of operation, which can be built based on the ILDA framework. Enabling multiple modes of operation in the ILDA systems aims to

contribute to the support of users in solving a broad range of tasks (*T7*). The three basic modes of operation are (a) LDA Navigation, (b) Application compositions and (c) In-app ILDA module.

***LDA Navigation*:** LDA Navigation is the primary mode of ILDA systems enabling the user to *navigate* between LD-based applications (LDAs) – select data outputted from previously used applications and put the data (or its subset) to another selected tool accepting the provided type of data. The ILDA system will be represented by a side panel (or in a similar way) showing the data, enabling the selection of the next application and possibly control elements of other optional features.

This mode aims to support the users in tasks previously requiring performance of manual data transfers between the tools ("copy-paste," or downloading and uploading) and to assist the users with a selection of appropriate tools. The mode is a realization of systems helping to guide *munging* (a term used for operations required to enable transfer of data between tools) described by Lebo et al. [22]. The majority of functions and features of ILDA systems introduced in Sections 3.3.2 and 3.3.3 are applicable mainly in this mode.

Application compositions: In application compositions, the integrated applications are used as modules of a composition, i.e., a mashup application. Such compositions powered by the ILDA framework will be similar to the mashups described by Matono et al. [72], but the utilization of Linked Data would simplify the creation of the compositions (cf. Section 3.2). This functionality of the ILDA framework could support the integration of standalone applications into the development-oriented adaptation approaches and semantic mashup frameworks (cf. *T6* in Section 3.3.1).

Application compositions can facilitate the creation of coordinated view user experience pattern [35] over multiple applications, where the applications are visualized simultaneously, and they coordinate the visualized content based on user actions. The flows of data are predefined so that the outputs of an application are transferred to another application or multiple applications by the ILDA system. Multiple known UX patterns based on coordinated views, such as brushing, drill down, overview and detail view, and synchronized scrolling [79] could be enabled over compatible applications, where the LDAs would be taking the role of visualization components in traditional coordinated views approaches. Such composed applications could be created manually or using a visual editor (similarly to the editor of Semantic Web applications described by Bakshi & Karger [23]).

In-app ILDA Module: The ILDA framework could also be used to create applications with the same functionality as the original adaptation approaches with interoperability features (cf. Section 2.2.4). In-app ILDA Module is a variation of the *LDA Navigation* mode, where the control elements and interoperability features of the ILDA system are provided as an internal component of a Linked Data-based application similarly to the Triple Action Framework's components in LD Viewer [31]. The result of performing a transition from such a component could lead to the start of a session in the LDA Navigator or enable just a single-step transition (again similarly as in the existing interoperability approaches). The mode could be used to create or enrich existing *dashboard* applications by providing links to integrated applications.

3.4 ILDA-Compatible Applications

In Section 3.3 we have analyzed the characteristics of ILDA framework and potential ILDA systems contributing to the task-oriented adaptation and general support of the user in task-solving activities by enabling interoperability of Linked Data-based applications (LDAs). In this section, we are observing the relationship between the ILDA framework and integrated applications from the applications' perspective. We strive to support the broadest possible spectrum of applications, but as we will observe in Section 3.4.1, there are several requirements for the applications which need to be fulfilled to enable the integration of applications into the ILDA framework. Additional requirements which could be met by the applications to enable optional features of the ILDA systems are discussed in Section 3.4.2. As part of their *Seamlessness theory,* Lebo et al. [22] have proposed a 5-star application rating scheme. In Section 3.4.3, we will discuss the rating scheme in the context of the analyzed ILDA framework.

3.4.1 Essential Requirements

This section describes the requirements which need to be fulfilled to enable the application's deployment in the context of ILDA system and to enable the system to fulfill its core requirements (Section 3.3.2.1) and provide core features (Section 3.3.3.1):

Browser-accessible: The ILDA systems will be implemented as web applications. Therefore, the integrated applications also need to be implemented as web applications accessible in standard web browsers on a specified URL address. However, there are potential exceptions to this requirement, which could be integrated through optional functionalities of the ILDA systems: (a) Semantic Web Services not providing user interfaces, and (b) non-web applications (e.g., desktop applications) optionally supported by the *manual step* feature (*Ft18*).

Online: Following the previous requirement, the applications must be online and accessible in user's environment (Internet or intranet) while used in the context of ILDA systems.

Enabling at least one-way data transfers: The applications need to enable at least one direction of data transfers (either accepting inputs or emitting outputs) in a way supported by the ILDA system. Alternatively, the applications can also be integrated, if such functionality can be added to the application by the implementation of a wrapper (*Ft07*) adding the required data transfer functionality.

Linked Data-based: Though it is technically possible to integrate applications not utilizing the Linked Data as a format of their inputs or outputs, integration of such applications would limit the functionality of the ILDA framework (e.g., the ability to support the user in selection of next steps – *Rq03*), and therefore we are not considering them in this thesis.

Described: The semantic descriptions enable ILDA systems to communicate with the applications (*Rq01*, *Rq02*), and to offer them as possible actions to the users (*Rq03*). A semantic description of the application should be accessible to the ILDA system (*Ft03*).

Consistent behavior: The applications' behavior related to the communication with the ILDA systems (emission of data and how the behavior of applications' links) should be consistent across all of the integrated applications (*Rq05*).

3.4.2 Optional Requirements

The fulfillment of multiple optional requirements of the ILDA framework and ILDA systems requires cooperation from the integrated applications. In this section, we list the features which can be implemented by the ILDA-compatible applications to enable operation of the optional features of ILDA systems:

Internal steps (supporting *Ft18*): The application could emit messages about the steps done internally inside of the application through the ILDA communication platform. These messages could be visualized in the session history and improve the documentation of the session (*Rq07*) or even used to replay the interaction.

Replaying interaction (supporting *Ft12*): When revisiting the application previously used in the session the application could enable replaying the internal interaction to help the user understand previous interactions with the application (*Rq07*).

Run-time inputs (supporting *Ft16*): Some applications could accept the inputs not only at their start but also during their operation. Run-time inputs could be used to enable communication in compositions of applications (*Rq13, Ft17*).

Automation of steps (supporting *Ft14*): To enable the automation of steps, the integrated application would need to emit the information about the "meaning" of the selection (e.g., the largest city in the visualized country), and also accept such information to update the output according to the new inputs (e.g., the selected country has changed).

3.4.3 Comparison with the Five-star Application Rating Scheme

The *5-star application rating scheme* is a linear scheme proposed by Lebo et al. evaluating the applications used in the visual analytics with regard to the concept of the *application seamlessness* [22]. We describe the scheme and the concept in Chapter 2, but in this section, we want to discuss the relation of the 5-star scheme to the ILDA-compatible applications.

There are two significant differences between Lebo's and our approach. (a) The 5-star scheme is used to evaluate any application employable in the visual analytical process, whereas we are interested specifically in the applications visualizing or processing Linked Data. (b) We argue that it is not possible to build a *linear* rating for the applications: for instance, Lebo's scheme ranks an application with three stars if it accepts RDF data as input and five stars if the outputs are emitted as RDF data. We believe the precedence of enabling inputs or outputs to be handled by the ILDA system is context-dependent and cannot be modeled in a universal scheme. Even Lebo et al. are aware of this issue of the scheme and are using special notation to distinguish the application fulfilling, for instance, the fifth-star requirements, but not some of the prior ones (and thus rendering the linear-like scheme nonlinear).

Nevertheless, we have been interested in the ratings and have analyzed them in context of the interoperability with the target of reusing useful concepts. In the following points we go through the meaning of the five-star ratings and discuss its relation to the ILDA approach:

- **1-star** – gets an application if it fulfills the criterion that *the sets of analysts, tool developers, and data providers should be disjoint.* The idea of this criterion is that it reduces the chance of the application being rigidly tied to a specific data set, or uses case and so be more flexible. While this might be true, possibly even in a majority of cases, it is not a heuristic based on observation. The criterion has been already criticized in the review of the paper for these reasons as being disputable [80].
- **2-stars** – 2-star *applications accept data via URL and always cite that URL in the future:* The ability of the application not to require manual input of the data is also a central component of our approach. However, the ILDA system could also support additional means of providing data to the applications (*Fn01*) to integrate existing tools into the IS. Lebo's insistence on the preservation of data in the application's URL is a means to document the analytical path and thus improve its reusability, an essential concept of the optimization of Lebo's *analytical seamlessness*. We also recognize the importance of documentation (*Fn06*) and reusability (*T3*), but the ILDA systems are targeting these concepts with different means.
- **3-stars** – application accepting *RDF data via URL:* We also support and encourage the usage of RDF data in the ILDA systems. The form of putting the data into the URL is not important as we enable the documentation (*Fn06*) and reusability (*T3*) with different means.
- **4-stars** – applications using *a tool's input semantics (OWL, SPARQL) to help guide munging* (a term from visual analytics used for operations required to enable transfer of data between tools): The 4-stars application is required to have a semantic description of its inputs, which is also an essential part of the ILDA systems (*Ft03*).
- **5-stars** – applications *outputting results as linked data:* the ILDA framework will support multiple types of output, one of them being the RDF data.

3.5 Summary

In this chapter, we have examined the concepts required for the creation of a general interoperability framework for Linked Data-based Web Applications, proposed the targets, requirements, and features of the ILDA framework and potential ILDA systems, and analyzed the ILDA framework from the perspective of the ILDA-compatible applications.

In Section 3.1, we explored the research areas focused on the interoperability in the context of Linked Data (data integration and semantic web services), and observed concepts which could be used in the application interoperability solution. In Section 3.2, we have mapped the approaches enabling interoperability of web applications outside of the Linked Data sphere. In this area, we have observed concepts enabling interoperability of applications in the modern web browsers.

Based on these findings, we have analyzed the required and potential advanced characteristics of the ILDA framework and ILDA systems enabling the adaptability of interaction flows. We have defined the targets the ILDA ecosystem based on the analysis of the targets of alternative adaptation approaches and exploratory systems. The recognized targets were then used to derive a structure of requirements and features aiming at achieving the targets and inspired by related research areas (Section 3.3).

The integrated LD-based application being an essential component of the ILDA systems will play a crucial role in the prospective spreading of the technology. They provide the central part of system's functionality, the role of ILDA system is just to enable *seamless* linking of the application functionalities. In the last part (Section 3.4), we have looked at the ILDA systems from the perspective of compatible applications and analyzed what the necessary and optional requirements for the applications are. We have also compared Lebo's rating scheme of the applications with our approach.

The findings of the conceptual analysis constitute the answer to the Research question 2 and provide a basis for the design of ILDA framework (cf. Chapter 5). In Chapter 4, we will see how the existing LD-based applications (with the focus on visualization and exploration applications) are capable of fulfilling the essential requirements and being integrated into the ILDA framework.

CHAPTER 4

Survey of Integrable LD-based Applications

In Chapter 3, we have analyzed how ILDA systems could operate and what are the requirements on the integrated LD-based applications (cf. Section 3.4). This chapter aims to explore the state of LD-based applications in the context of how they could be integrated into an interoperability system (using a wrapper, cf. Section 5.2.4). The results of the survey are partly reflected in the design of the ILDA framework in Chapter 5 (affecting the modeling of application descriptions and selection of supported input types and methods) and are used to model the test scenarios of the feasibility assessment of implemented prototype in Chapter 6. In Section 4.1, we present the design of the survey. Section 4.2 contains descriptions of the reviewed applications, and the results of the survey are discussed in Section 4.3.

4.1 Survey Design

To explore how the currently existing application could be integrated into the ILDA framework, we mapped the existing visualization and exploration systems with the aim to identify what types of input data and input delivery methods the tools are employing, what types of output the applications could potentially emit, and observe the capability of the tools to be adjusted by a wrapper to provide the output functionality.

4.1.1 Focus and Sources

The survey is focused on visualization and explorations systems. The reason for this selection is the very good coverage of these types of tool by existing surveys and also the fact that this selection offers a good representation of the current tools, since as of 2014 the significant majority (91%) of LD-based tools mapped by a study by Hayes et al. [44] contained visualizations of the data (but we were not able to find more recent sources to confirm this trend).

This survey is for the most part building on previous surveys on the LD-based visualization and exploration applications by Dadzie & Rowe [9], Bikakis & Sellis [12], Marie & Gandon [13], and Dadzie & Pietriga [11]. We have also added few additional applications:

Default DBPedia view[28] and LD Viewer [31] as the standard interfaces for DBpedia-related resources, and Linked Widgets Platform [17] representing the mashup frameworks.

4.1.2 Criteria for Inclusion

From all the applications reviewed in the source surveys we have selected the ones fulfilling following criteria:

Web applications: Since the interoperability system will be based on the HTML Web Messaging [37] functionality, we are interested only in the web applications.

Running version available on the Internet: We have reviewed implementation-related characteristics of the application that were described neither in the research papers describing the tools nor in the source surveys (cf. Section 4.1.3). For that, we need to observe the running applications.

We have considered the inclusion of applications without available online version but with the published source code, but the building and running of the applications have turned out to be a very complicated task (because of the often lacking documentation and non-existent support) beyond the scope of this survey.

4.1.3 Observed Integrability Characteristics

We have observed the characteristics of the selected applications relevant to the data transportation functionality of the interoperability system, specifically:

Inputs and output types: We have observed which types of data the application accepts as its inputs and could emit as outputs. These are the types observed in the survey:

- **RDF resources, instances and classes**: provided in the form of their URI (cf. Section 2.1.1), in some cases the type is further restricted, and the tool accepts or provides resources only from a specific domain (e.g., DBpedia)
- **RDF graph (file)**: a serialized version of an RDF graph
- **SPARQL Endpoint:** URL address of a SPARQL engine enabling the application to query for data (cf. Section 2.1.1)
- **String:** names of concepts or whole SPARQL query

Input methods: Input methods are the means of how the application accepts the inputs. The reviewed applications supported following input methods:

- **URL parameter**: The application expects the input as a part of the URL of the HTTP request for the application.
- **Manual**: The input is provided manually by the user using HTML input elements or in a similar way, and it is not possible to send the data in the HTTP request for the application. Such manual inputs are handled by the client-side part of the application.

[28] http://dbpedia.org/page/*

Output potential: Output potential is an assessment of the possibility to create a wrapper (cf. Section 5.2.4) providing the output emission functionality. The wrapper is able to access the client-side part of the application and inject or alter functionality or HTML elements of the application. Specific situation is in the case of applications employing Adobe Flash technology on the client-side. Without the explicitly enabled communication with JavaScript application (i.e., the wrapper) from the applications written in ActionScript[29] the JavaScript-based wrappers are not able to affect the behavior or interface of Flash applications.[30]

However, even from the applications which can be altered by the injected wrapper, not every application can be enriched with the output emission functionality. If the wrapper cannot access the data is the original client application in its complete form (e.g., whole URIs of the resources), the wrapper cannot reconstruct the data and emit it as output. Figure 4.1 shows part of the Aemoo[31] tool's user interface and a related snippet of its DOM tree containing the whole URI of the selected resource.

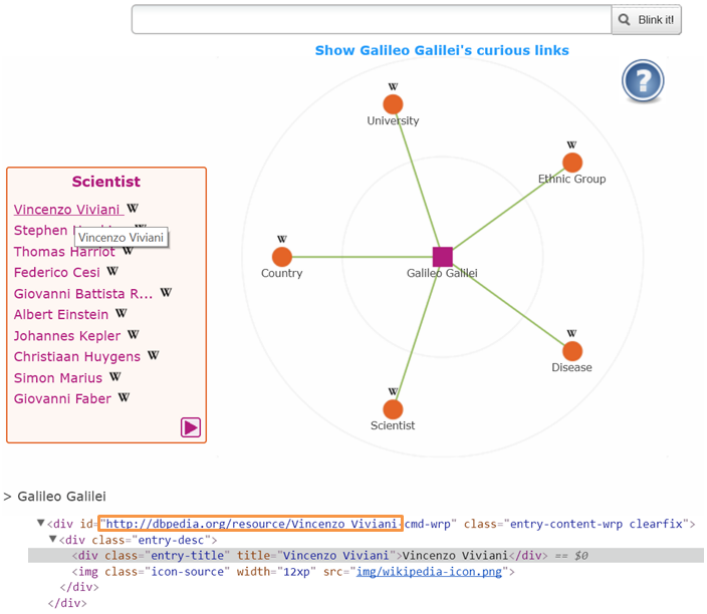

Figure 4.1: Part of the Aemoo's user interface and
the related section of DOM tree containing resource's URI

[29] https://www.adobe.com/devnet/actionscript.html
[30] https://help.adobe.com/en_US/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7cb1.html
[31] http://wit.istc.cnr.it/aemoo

4.2 Reviewed Applications

The four source surveys reviewed together 82 applications, of them 55 web applications. From these applications, 25 tools were available either in the form of source code or as an online live version (full or demo). The source codes of 16 applications were available online, live version (full or demo) was available for 14 applications. Together with the three additional tools, we have observed the integrability characteristics of 17 LD-based applications. In the rest of this section, we discuss the tools and their integrability characteristics.

Aemoo [14] is a graph-based exploratory search application based on the Encyclopedic Knowledge Pattern (EKP). EKPs were developed based on the graph analysis of DBpedia and Wikipedia, and are used to define patterns of knowledge for a particular type of data (e.g., "Galileo Galilei" as a "scientist" will have a relation to "university", as a "human" to "ethnic group" and "country", etc.). For a selected resource, Aemoo shows the relations of the resource to other resources of types provided by the EKPs and explains the relation based on a snippet of DBpedia's text. Ameoo also enables to display unusual relations with its "curiosity" function. The evaluation tests have shown that the application is capable of outperforming the Google search in some search tasks. From the interoperability point of view, the available version of the tool[32] uses its graphs analysis of DBpedia and Wikipedia, it accepts the transformed DBpedia resources (instances) in the form of an URL parameter, and its implementation enables to create wrappers capable of outputting DBpedia resources (classes and instances).

CubeViz [81] is a faceted browser capable of visualizing statistical data. The tool provides data visualizations in the form of different types of charts (line, bar chart, column, area, and pie). The source of data can be any SPARQL endpoint or an uploaded RDF graph containing RDF DataCube[33] data. In the available demo version[34], the SPARQL endpoint or the RDF file are provided to the application as manual inputs. CubeViz displays aggregations of the data and provides no opportunity to output particular data.

Facete [82] is an exploration and visualization tool available online[35] enabling the faceted browsing of data with a geospatial dimension. It enables selection of data according to the computed facets, showing the data in a tabular form and on a map. As a source of the data Facete accepts any SPARQL endpoint, which can be inputted manually. The implementation enables the creation of a wrapper outputting the data as RDF resources.

graphVizdb [83] is a graph-based visualization platform enabling interactive presentation of large RDF graphs. The platform targets the problem of visualization of large graphs both from the technical point of view (indexing of the graph and storing the index in the database) as well as from the perspective of user experience (providing an overview, search, filtering and zoom functionality). The available version[36] of graphVizdb allows selection

[32] http://wit.istc.cnr.it/aemoo/
[33] http://www.w3.org/TR/vocab-data-cube/
[34] http://cubevizjs.demo.aksw.org/
[35] http://cstadler.aksw.org/facete/
[36] http://www.graphvizdb.com/

of two data sources – DPLP computer science library[37] and the DBpedia's person subset. The application does not accept any inputs, and adding the functionality of outputting the data is not possible since the implementation does not contain the exact URIs and is even trimming the longer names of resources.

LD-VOWL [84] a web application extracting and visualizing schema information of Linked Data sources. The ontology information is extracted from a SPARQL endpoint and visualized using VOWL notation[38]. The available version[39] enables to define the source of the visualized information as any SPARQL endpoint, which can be provided to the application in the URL parameter. It is possible to extract the URIs of visualized classes and use the information to implement output emission functionality in a wrapper.

Linked Jazz [85] is a web application[40] visualizing the network of the social and professional relations among American jazz musicians based on interviews from jazz archives. The project aims to explore how the usage of Linked Data can assist in processing and presentation of cultural heritage materials. The information can be explored through an interactive graph visualization providing an overview of the relations and supporting a more focused exploration. Linked Jazz combines data from DBpedia with manually curated archive data. The application does not support any inputs, and it's not possible to collect outputs from the application.

LinkedPPI [67] is a framework for semantic integration of various data sources from the biological domain related to the interactions between proteins (PPI – protein-protein interaction). The available web application[41] enables visualization of the integrated data. It uses internal data sources, enables to input only the names of searched proteins, the data is visualized using the HTML canvas[42], so even the visualized texts are contained in a bitmap, and the potential wrapper cannot access them.

LodLive [86] provides a simple and user-friendly general graph-based Linked Data visualization interface[43] to explore the Web of Data. The interactive graph enables to show or hide related data and displays relations between them. Geographical data can be visualized on a map; images are also displayed in a dedicated component. LodLive can visualize any resource based on its URI (provided as a URL parameter). The implementation enables the creation of a wrapper outputting the visualized resources.

NHGRI GWAS Diagram[44] is a visual representation of the Catalog of Published Genome-Wide Association Studies employing Linked Data in the data integration process[68]. It uses internal data, and the visualization does not enable input or output functionality from the interoperability point of view.

OpenLink Data Explorer[45] represents the provided resource in multiple textual as well as visual (i.e., showing related images, showing geographical resources on a map) ways. Data

[37] http://dblp.uni-trier.de/
[38] http://vowl.visualdataweb.org/v2/
[39] http://vowl.visualdataweb.org/ldvowl/#/
[40] http://linkedjazz.org/network/
[41] http://srvgal78.deri.ie/linkedppi/
[42] http://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas
[43] http://en.lodlive.it/
[44] https://www.ebi.ac.uk/gwas/diagram
[45] http://linkeddata.uriburner.com/ode/

Explorer uses only the data it obtains by dereferencing the resource's URI. The resource has to be inputted manually. Resources (not just instances, but even the properties and classes) could be easily collected by the wrapper and outputted.

RelFinder [87] offers discovery of relationships between selected resources and visualizes the relations in the form of an interactive graph. The demo version of the application[46] can use any provided SPARQL endpoint for the discovery process (or use the default DBpedia endpoint). It accepts any number of resources (URIs) in the form of URL parameters. The application is Flash-based, and because of this, the potential wrapper could not provide output functionality.

Semantic Wonder Cloud (SWOC) [88] allows the exploration of the relations between resources in a simple user interface. The user selects a starting topic, and the SWOC visualizes the topic together with 10 most similar concepts in a graph form. The demo version[47] uses an IT-related subset of DBpedia dataset and provides a search box for the concepts. SWOC is implemented in Adobe Flash and does not allow any outputting functionality.

SynopsViz [89] is an exploration tool targeted at large datasets already discussed in Section 2.2.2. The demo application[48] is using the DBpedia's infobox or Persondata subsets and supports no input or output functionality.

VISUalization Playground (VISU) [90] is a simple interactive tool for visualizing data based on results of a SPARQL query. The demo application[49] enables manual input of the SPARQL endpoint and a SPARQL query. Outputs can be emitted by a potential wrapper.

Default DBPedia view is the standard text-based representation for DBpedia resources. When the user tries to access a DBpedia resource (e.g., http://dbpedia.org/resource/Spain) in a web browser, the request is automatically redirected to this representation (e.g., http//dbpedia.org/page/Spain). From the interoperability point of view, it can be modeled as an application accepting DBpedia resources in the request's URL. The interface enables multiple types of potential outputs – DBpedia resources (instances, classes, and relations), but also the RDF graph visualized in the interface.

LD Viewer [31] is a customizable framework for the presentation of Linked Data. The data is shown in a tabular layout with additions of graphical components such as images or map visualizations. The interoperability features of LD Viewer were already discussed in Section 2.2.4. The DBpedia-based version of LD Viewer[50] accepts any resources of DBpedia in the request's URL and can be adjusted to provide outputs of DBpedia resources.

Linked Widgets Platform [17] is a sophisticated mashup platform already described in Section 2.2.3. The widgets can use any data sources, the platform does not take any inputs, but some of the widgets can be adjusted to provide outputs.

[46] http://www.visualdataweb.org/relfinder/relfinder.php
[47] http://sisinflab.poliba.it/semantic-wonder-cloud/index/
[48] http://synopsviz.imis.athena-innovation.gr/
[49] http://data.aalto.fi/visu
[50] http://ldv.dbpedia.org

4.3 Conclusions

From the 17 review applications, four of the applications in their current form provide no options for the integration into the ILDA system ([83], [85], [68], [89]). The integration of applications enabling the manual type of input ([81], [82], [67], ODE, [88], [90]) only would be more difficult, since such input type will require implementation of a wrapper emulating user activity. On the other hand, the tools accepting data as part of their URL addresses and having output potential ([14], [84], [86], DBPV, [31]) are the most suitable choices for the integration into the ILDA framework.

Table 4.1 summarizes our findings related to the integrability of the applications. Column survey shows the source survey of the application (DR – Dadzie & Rowe [9], BS – Bikakis & Sellis [12], MG – Marie & Gandon [13], and DP – Dadzie & Pietriga [11]). Columns Input data and Output data describe the types of data accepted or potentially outputted by the applications. Resources, Instances, and Classes are RDF resources, instances and classes in the form of the URI, in some cases restricted by a specific domain; Endpoint is a SPARQL endpoint URL address.

In this chapter, we have provided an overview of the ILDA system integration capabilities of existing LD-based applications. We have selected the sources of the survey, defined the inclusion criteria for the applications based on the availability of their online version, defined the observed integrability characteristics of the applications and reviewed the tools according to these characteristics. The findings of this survey are used as an input to the modeling of the ILDA framework (cf. Chapter 5) and are used to model the test scenarios of the feasibility assessment of implemented prototype (cf. Chapter 6).

Tool Name	Survey	Input Data	Input Type	Output Data
Aemoo [14]	MG	Instances (DBPedia)	URL	Resources (DBPedia)
CubeViz [81]	BS	Endpoint / RDF file	manual	-
Facete [82]	BS	Endpoint	manual	Instances
graphVizdb [83]	BS	-	-	-
LD-VOWL [84]	DP	Endpoint	URL	Classes
Linked Jazz [85]	MG	-	-	-
LinkedPPI [67]	DP	String (Protein name)	manual	-
LodLive [86]	BS	Resource	URL	Instances
NHGRI GWAS Diagram[68]	DP	-	-	-
OpenLink Data Explorer (ODE)	DR	Resource	manual	Resources
RelFinder [87]	DR, BS	Resources, Endpoint	URL	-
Semantic Wonder Cloud [88]	MG	String (Concept name)	manual	-
SynopsViz [89]	BS	-	-	-
VISU [90]	BS	Endpoint, String (SPARQL query)	manual	Resources
Default DBPedia view (DBPV)		Resource (DBpedia)	URL	Resources
LD Viewer[31]		Resource (DBpedia)	URL	Resources
Linked Widgets [17]		-	-	Resources

Table 4.1: Reviewed Applications and their Input and Output potential

CHAPTER 5

ILDA Framework

Building on the analysis of required and optional characteristics of the ILDA framework and ILDA systems from Chapter 3, in this chapter, we introduce the ILDA framework – a set of design principles defining the components and inter-component communication of an ILDA system. The purpose of the ILDA framework is support development of adapted ILDA system for a broad range of use cases and contexts. The framework should provide guidance for implementations of ILDA systems and enable reuse of components in different implementations and modes (cf. Section 3.3.4). To show the relationships between components of the framework and identified characteristics of ILDA framework and systems from Section 3.3, in the descriptions of the framework concepts, we will refer to the requirements and features enabled or supported by the currently described concept.

We start by discussing the principles guiding the creation of ILDA framework (Section 5.1). The functionality of the ILDA systems identified in Section 3.3 is then divided into components (Section 5.2). To enable the reuse of components, standardized ways of communications between them need to be defined. Therefore, we propose the data models used in the inter-component communication in Section 5.3 and component interfaces in Section 5.4.

5.1 Principles

The principles followed during the design of ILDA framework are directly connected to the identified required features of ILDA framework (cf. Section 3.3.3) and aiming to allow the creation of various ILDA systems (*Rq16*) and by that supporting the users in a broad range of tasks (*T7*). We have based the ILDA framework on the following principles:

Reuse of ILDA-compatible applications and descriptions: The access to the integrated applications and their descriptions should not be bound to a specific ILDA system, but rather it should be possible to share the applications and their descriptions across multiple ILDA systems. This can contribute to the effectivity of development of adapted ILDA systems (*Rq16*) – the developers can implement new functionality over an existing set of integrated applications.

Internal Modularity: The internal functionality of framework should be modular and should allow reuse of the components in different contexts (e.g., in systems implementing various modes of operation, cf. 3.3.4) and replaceability of the components. This principle is

a realization of the identified modularity feature of the ILDA framework (*Ft22*) aiming to support the creation of adapted ILDA systems (*Rq16*).

Functional Extensibility: The framework components, data models, and interfaces should enable the addition of new functionality without affecting the operation of existing ILDA systems (*Ft23*).

Development Language-Agnostic: The framework should enable the creation of components using various languages and technologies, independent from technologies used in other modules. The framework should determine the use of specific technologies only for the interfaces. This principle is also aiming to support the developers of ILDA systems in their ability to create the systems effectively (*Rq16*).

5.2 Components of the Framework

The internal functionality of ILDA systems should be split into two main system components – *ILDA Service* and *ILDA Manager*, and an *integrated application* (or in the case of application compositions multiple applications). The application can be possibly extended by a *wrapper* facilitating the compatibility of the application with the framework. Such modular architecture supports the creation of ILDA Managers facilitating different modes of operations, such as the interaction flows spanning over multiple tools (*T2*) or application compositions (*T6*), complex interaction paths (*Rq15)* and could enable the creation of ecosystems, where different implementations of ILDA Manager share the same ILDA service (or multiple services), applications and wrappers.

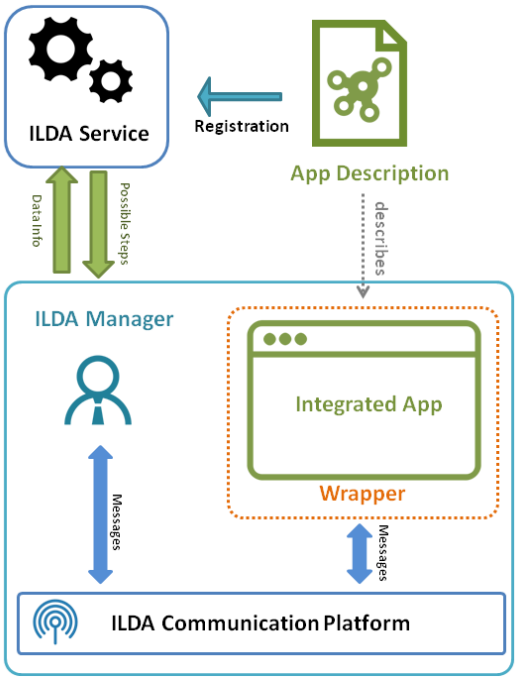

Figure 5.1: Architecture of the ILDA System

Figure 5.1 provides an overview of the architecture of ILDA systems. The ILDA Manager (Section 5.2.2) communicates with the integrated application (Section 5.2.3) through the ILDA Communication platform (Section 5.4.1) by sending the Interaction messages (Section 5.3.1). The integrated applications are registered (Section 5.4.3) in the ILDA Service using the Application Descriptions (Section 5.3.2). The ILDA Service based on the information from Application Descriptions provides the ILDA Manager with the list of possible next applications through the ILDA Service API (Section 5.4.2).

5.2.1 ILDA Service

ILDA Service is a Server Side component responsible for selection or recommendation of applications employable as the next step in the interaction session given the currently selected data. The service manages the descriptions of the integrated applications (cf. Section 5.3.2 and Section 5.4.3) and selects or recommends the possible next applications as a response to the request from ILDA Manager (cf. Section 5.4.2).

The service supports the user in selection of next step in interaction path (*Rq02*) by providing the list of possible next applications based on the selected data, it provides the ILDA Manager with the information describing how the integrated applications accept their inputs, and by that assists with the transfer of data from and to applications (*Rq01*). Advanced implementations of ILDA Service can recommend the applications not just based on the types of data, but could use additional information (e.g., user history, explicit user preferences) to order, filter or highlight the list of possible next applications (*Rq11, Ft15*) or help with discovery of new applications (*Rq12*). The ILDA Service can be implemented in any language supported by the server and communicates using a standardized API (cf. Section 5.4.2).

5.2.2 ILDA Manager

ILDA Manager is a client-side web application hosting the integrated application or multiple applications, facilitating communication with the applications and providing all the user interaction features of the ILDA system. It operates on the client-side of the system alongside the integrated applications (cf. Section 5.2.3) and their potential wrappers (cf. Section 5.2.4).

ILDA Manager facilitates all the transfers of data from and to the integrated applications (*Rq01*) through the ILDA Communication Platform (cf. Section 5.4.1) based on the information from the ILDA Service, and by that helps the user with the entry of applications' inputs (*Rq04*). It loads and closes the hosted integrated applications (Rq*02, Ft02*), enables the user to select next steps based on the selection or recommendation from ILDA Service (*Rq03, Rq11, Ft4, Ft15*).

The Manager can also implement the advanced functions of the ILDA systems: it can visualize the history of the interaction session (*Rq07, Ft09*) and enable the user to come back to previous steps in the history (*Rq10, Ft11*) or even could enable forking of the interaction path into multiple parallel paths (*Rq15*, *Ft21*). The collaboration of multiple users (exporting, importing, and sharing of the interaction session) could also be implemented on the ILDA Manager's level (*Rq08*, *Ft10*). Its implementation also defines the mode of operation

of the system (cf. Section 3.3.4). ILDA Manager is a client-side web application, and therefore must be implemented in a language supported by the web browsers (i.e., JavaScript).

5.2.3 ILDA-Compatible Applications

ILDA-compatible applications are stand-alone LD-based web applications which could be integrated into ILDA systems. The applications can be from different origins than the ILDA system (cf. Section 3.2). The application could either natively support the ILDA framework (i.e., enabling the communication with ILDA Manager through the ILDA Communication Platform, cf. Section 5.4.1), or be integrated using a *wrapper.* The application is contained in an *iframe* HTML element[51], which is the most convenient way of how to visualize a web page or application from the different origin in the same window.

The applications need to support at least one-way communication (taking inputs or emitting outputs) to be ILDA-compatible (*Rq01*). The requirements for the ILDA-compatible applications are more discussed in Section 3.4. Some of the advanced functions of ILDA systems require cooperation with the applications, such as emissions of the internal steps for documentation of history (*Rq07*, *Ft12*), replaying of interaction and automation of steps (*Ft13*, *Ft14*). The ILDA framework places no constraints on the implementation of applications (languages or libraries), as far as the applications are fulfilling the criteria described in Section 3.4.1.

The integrated applications should behave consistently in actions related to the system (*Rq05*). The application (or its wrapper) can recognize when it is used in the context of ILDA system and possibly alter its behavior. The aim is to clearly distinguish which user actions (e.g., clicking on links) lead to a continuation in the interaction with the tool and which lead to emissios of outputs. Specific guidelines describing how the applications should alter their behavior will need to be determined as part of future user studies.

5.2.4 Application Wrappers

Wrappers are extension mechanisms enabling the applications not compatible with ILDA system (currently all of the LD-based applications) to become compatible and integrable into the ILDA system (*Ft07*). The main functions of application wrappers are (a) to change application's behavior to promote the consistency of applications' behaviors (*Rq05*), and (b) to enable the communication with ILDA Manager through the ILDA Communication Platform (cf. Section 5.4.1).

We have explored different ways of how to create the wrappers to find the right approach as the modern web browsers are prohibiting the programmatic access to DOM trees[52] of documents of different origin (e.g., domain) than the origin of the script trying to manipulate the DOM tree. The ILDA framework introduces wrappers encapsulated as a browser extension (specifically Chrome extension[53]). Taking the form of browser extension allows overriding of the browsers' security controls. The drawback of this approach is

[51] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe
[52] https://www.w3.org/TR/DOM-Level-2-Core/introduction.html
[53] https://developer.chrome.com/extensions

the necessity to install a browser extension containing the wrappers to enable the use of applications requiring wrappers for their integration into the ILDA framework.

5.3 Data Models

The ILDA System exchanges internally two types of data. The interaction messages are exchanged between the integrated applications and the ILDA Manager and can contain transferred data or other information required for advanced functionalities of ILDA systems. The other type of exchanged data are descriptions of integrated applications used by the ILDA Service for the selection of possible next actions, and to provide the ILDA Manager with all the information required for communicate with integrated applications.

5.3.1 Interaction Messages

Interaction messages are exchanged between the ILDA Manager and the integrated applications through the ILDA Communication Platform (cf. Section 5.4.1). From the implementational point of view, they are JavaScript objects with a defined structure. There are four main types of messages: (a) *hello message*, (b) *hello-ack message* and (c) *data-input message* and (d) *data-output message*, but additional types could be defined for the advanced functions such as internal steps (*Ft12*) and replaying and automation (*Ft13*, *Ft14*).

The *hello* and *hello-ack* messages are used to initiate communication between the ILDA Manager and the integrated application. After loading the integrated application, the manager sends *hello* messages to inform the application about its usage in the ILDA context (so that the application could adjust its behavior, cf. Section 5.2.3), the application should answer with a *hello-ack* message.

The data between ILDA Manager and the application is transferred either directly inside of the HTTP request for the application or using the *data-input* and *data-output* interaction messages. Data inputs are sent from ILDA Manager to the application and vice versa for the data outputs. The types of exchanged data are described in Section 5.3.2.3.

5.3.2 Application Descriptions

Application descriptions contain information about the functionality of integrated applications, mainly regarding the applications' inputs and outputs. They are a concept similar to the descriptions of semantic web services (c.f. Section 3.1.2). The information is used for the creation of lists of available next actions offered to the users. The application descriptions also inform the ILDA Manager of how to facilitate communication with the described application.

ILDA Service could provide a registration interface for application descriptions (cf. Section 5.4.3). The information should be described using the ILDA Ontology vocabulary (cf. Section 5.3.2.5). In the following subsections, we describe different aspects of the application descriptions and show how they are modeled in the ILDA Ontology.

5.3.2.1 Integrated Application

The description can contain basic information about the application, such as its name, associated keywords, information about the creator, when it was created, and license information. Integrated applications should be modeled in the ILDA Ontology as instances of the *ilda:LDApplication* class. For the descriptions of applications various resources from the Dublin Core metadata vocabulary[54] could be reused, such as *dcterms:description, dcterms:creator.*

5.3.2.2 Profiles

One application can have one or multiple profiles to enable different combinations of inputs or variations of the functionality. To model such internal diversity of the application, we introduce a concept called *profile*. Every profile has an associated *URL template* describing how to access the functionality described by the profile. In the ILDA Ontology, application profiles are instances of the *ilda:ApplicationProfile* class. The URL template is attached to the application profile instance using the *ilda:hasURLTemplate* property. If the application profile contains parameters delivered through the HTTP POST method (described in Section 5.3.2.3), application profiles can also contain information about the used HTTP POST encoding method[55] modeled by the *ilda:hasPostEncodingMethod*.

5.3.2.3 Data Parameters

Data Parameters are describing the data coming into the application – input parameters or the data application has outputted – output parameters. The ability to define parameters (specifically the inputs) of applications, and use this information to facilitate selections of actions and transitions between applications is the most fundamental feature of the ILDA framework. The descriptions of parameters need to contain three different perspectives: (a) the data type of the parameter, (b) the method of how the parameter is transferred to or from the application, and (c) the temporality - when is the parameter delivered. The general class of data parameters in ILDA Ontology is the *ilda:Parameter*, which has subclasses *ilda:Input* and *ilda:Output.* The parameters must have an identifier defined using *ilda:hasParamId* property and a name shown to the users defined using *ilda:ParamName.* Input parameters can be described as optional (*ilda:isOptionalInput*).

Inputs and outputs: Input and output parameters can obviously describe the same types of data (the same data is once in the position of an output and later in the position of input), but they differ in other aspects:

- **Required inputs optional outputs**: The description of inputs of an integrated application is required; otherwise the system would not know how to operate with the integrated application. The description of the output parameter is optional. While it is advantageous to know the type of outputs beforehand (e.g., for prefetching of the available tasks), the ILDA system can flexibly react to the received data and look up the available steps according to the output.

[54] http://dublincore.org/
[55] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

- **Transfer methods**: It was our intention to support as many existing tools as possible while minimizing the effort required to create the wrappers enabling the integration of these applications. Therefore, we have enabled multiple methods to transfer input parameters into the application, with the aim to match the methods the applications are currently using. On the other hand, the outputs are a new concept, don't need to support multiple existing transfer methods, and can only be transferred using the JavaScript events.
- **Different temporality**: As we will see, the temporality is modeled in a different way than by inputs.

Data Types: An integrated application can have no inputs or outputs (but not both, since it would not be possible to use such application in the interaction paths), or one or multiple inputs and outputs of supported data types. Supported data types are modeled in the ontology as subclasses of the *ilda:DataDescription*. ILDA framework support following data types:

a. ***RDF resource (URI)****:* A string representing the URI of a resource; some tools will require the URI to be transformed when used as an input parameter of the applications (e.g., ., the resource with URI *http://dbpedia.org/resource/Spain* is delivered to the LD Viewer in its URL as *http://ldv.dbpedia.org/#/page/**Spain***), but as *http://en.lodlive.it/?**http://dbpedia.org/resource/Spain*** (i.e., without transformation) to the LodLive tool. The ILDA Ontology enables definition of RDF resource data types using the *ilda:RdfResourceDescription* class. The transformations of the URI are modeled as instances of the *ilda:UriTransformation* class related to the *ilda:InputMethod* (described below) using the *ilda:hasResourceUriTransformation* property. The transformation contains a regular expression pattern (*ilda:hasRegExpMatcher*) and an output template (*ilda:hasOutputTemplate*) specifying how the parts matched by the regular expression should be combined into the form accepted by the application.
   Specific constraints for the resource could be modeled:
   i. *Domain-based*: the URI of the resource needs to follow a pattern (i.e., contain a specified domain), e.g., the URIs of the resource starting with *http://dbpedia.org;* domain-based constraint is modeled in the ontology using the *ilda:hasDomainRestriction* property.
   ii. *Type-based*: the resource is of a specific type (e.g., country, person). For this type of constraint, the IS will require a types lookup functionality. In the ontology, the type-based constraint is modeled through the *ilda:hasClassRestriction* property.
   iii. *Shape-based*: More complex constraints (e.g., "states only in Europe") based on SHACL language[56] or a similar technology. The system would need to be able to retrieve additional information about the resource (for instance, using a SPARQL endpoint whose address is contained in the contextual information). This type of constraint is more suitable for the RDF graphs as the constraints could be evaluated

[56] https://www.w3.org/TR/shacl/

without the need to retrieve additional data. ILDA Ontology enables definition of shape constraints using the property *ilda:hasShape.*

Additional information about the resource could be sent in the message as its c*ontext*, i.e., type of the resource, and the context in which it was located (RDF triple).

b. ***RDF graph***: The RDF graph serialized in a form supported by the LDA. The graph data is either transferred in a serialized form or specified by a URL containing the graph. The ILDA system could provide data storage and assign URLs to the data. The content of the RDF graph could be constraint using the already mentioned SHACL language[57] or similar technology. ILDA ontology enables definition of RDF graph parameters using the class *ilda:RdfGraphDescription.* Accepted serialization types can be defined using the *ilda:acceptedGraphSerialization* property. Whether the graph can be delivered in the serialized form, as a URI referring to the graph, or both options are supported, can be expressed using the *ilda:graphDeliveryMethod* property.
c. ***Data source address***: e.g., a SPARQL endpoint's[58] location in the form of an URL address. SPARQL endpoints are modeled as *ilda:SparqlEndpointDescription*'s instances with the address specified using the property *ilda:sparqlEndpointAddress.*
d. ***RDF literal or non-RDF data*:** Basically any type of data could be an input of the application. The data would need to be provided by the user inputted manually or selected from a predefined list, or the ILDA system could enable the user to fetch a literal value from an outputted RDF graph. Examples of this type of inputs are text, numbers, dates, or more complex data in the form of JSON. These types of data are modeled in the ontology as instances of *ilda:LiteralDescription* class with data type specified using the *ilda:hasLiteralType* property.

Delivery methods

Delivery methods describe how the parameter is transferred from or to the application. Input parameters can be transferred using multiple types of delivery methods to support a broad spectrum of tools (*Rq06*). We have identified three main method types for input delivery. The input delivery methods are modeled in the ontology as subclasses of the *ilda:InputMethod* class having relation with the *ilda:Input* through the *ilda:hasInputMethod* property. The first two types could be combined (some of the inputs sent as URL parameters while the others as POST parameters). Outputs can be transferred only by event emission.

a. ***URL parameters***: In this case, the parameter is delivered as part of the URL of the request for the application. This method is not suitable for longer forms of data (e.g., RDF graphs), but could contain links (URL) pointing to the location of such forms of data. This input method is modeled in the ontology as *ilda:UrlParam* with the name of the parameter defined using the *ilda:hasReqParamName* property. In some cases,

[57] https://www.w3.org/TR/shacl/
[58] http://semanticweb.org/wiki/SPARQL_endpoint.html

the parameter needs to be encoded (e.g., the RelFinder[59] application expects its parameters to be Base64 encoded). The encoding method could be specified using the *ilda:isEncodedBy* property.

b. ***POST parameters*:** Both smaller and larger forms of data could be sent as HTTP POST parameters. If the POST parameters are employed, the request uses the HTTP POST method in the request for the application. POST parameters are modeled as instances of *ilda:PostParam* class. They can have defined their names (*ilda:hasReqParamName*) and potential encoding (*ilda:isEncodedBy*) similarly to the URL parameters.

c. ***Event emission*:** The data could be delivered from and to the applications using the JavaScript events of the ILDA Communication Platform (cf. Section 5.4.1). Event parameters are modeled in the ontology using the classes *ilda:StartupEventParam* and *ilda:RuntimeEventParam* based on the temporality aspect described below. Both of the classes need to have defined the name of the event (*ilda:hasEventName*).

Parameter temporality

Inputs: The standard way of delivering the inputs to the application should be at the start of the application (inside of the HTTP request or *'on load'* event, modeled as *ilda:StartupEventParam*). The event-based input method can be used to deliver some additional inputs to the application even during the interaction (*ilda:RuntimeEventParam*).

Outputs: For outputs, the temporal aspect is modeled using the concept of the *trigger. The t*rigger describes how the emission of output is triggered:

- **User selection**: The output is emitted based on user's interaction with the integrated application; the users select the data for output.
- **Automatic**: The output is emitted automatically by the integrated application when reaches its final state. This can happen with or completely without the interaction with the user.

Standardly, outputs are expected to be user selected. Automatic outputs can be modeled in the ontology using the *ilda:isAutoOutput* property of the *ilda:Output* class.

5.3.2.4 Process information

Process information can contain descriptions of internal capabilities of the applications. Currently, the ILDA ontology models only user interface related aspects of the applications, but additional information could be added in the future describing the application's support for advanced ILDA framework features. Description of the application type and the utilized visualization types could also be added to be shown to the users and support them in the selection of next actions (*Rq03*) or to be used as an input for the recommendation of actions (*Rq11*). The process information is an optional part of the application description and is modeled in the ontology as an instance of class *ilda:ProcessDescription*.

The currently modeled user interface-related process information is describing whether the application has a user interface at all (i.e., if it is not a web service) and whether the

[59] http://www.visualdataweb.org/relfinder.php

application allows interaction with the user. The information about the existence of the user interface is modeled using the *ilda:hasUI* property and application's support of the interaction with the user is modeled using the *ilda:*is*Interactive* property.

The process information can optionally contain a description of application's support for advanced ILDA features such as internal steps (*Ft12*), replay of steps (*Ft13*) or automated steps (*Ft14*). Unrecognized elements of the descriptions will be ignored by the ILDA systems.

5.3.2.5 ILDA Ontology

The ILDA ontology is the vocabulary for descriptions of the integrated applications developed using the RDF[60] and OWL[61] languages reusing concepts from the Dublin Core metadata vocabulary[62]. It was developed following the seven-step ontology creation methodology by Noy & McGuinness [61]. The ontology can be used for the registration (cf. Section 5.4.3), and also internally in the ILDA Service (cf. Section 5.2.1).

Storing the application descriptions in form of ontology provides multiple advantages: the ontology (a) defines a standardized form of description for the LD-based applications probably familiar to the developers of LD-based tools, (b) enables reuse of some of the concepts from existing vocabularies, (c) makes possible the future integration with semantic web services, and (d) enables future extensions. The ILDA Ontology is visualized in Figure 5.2.

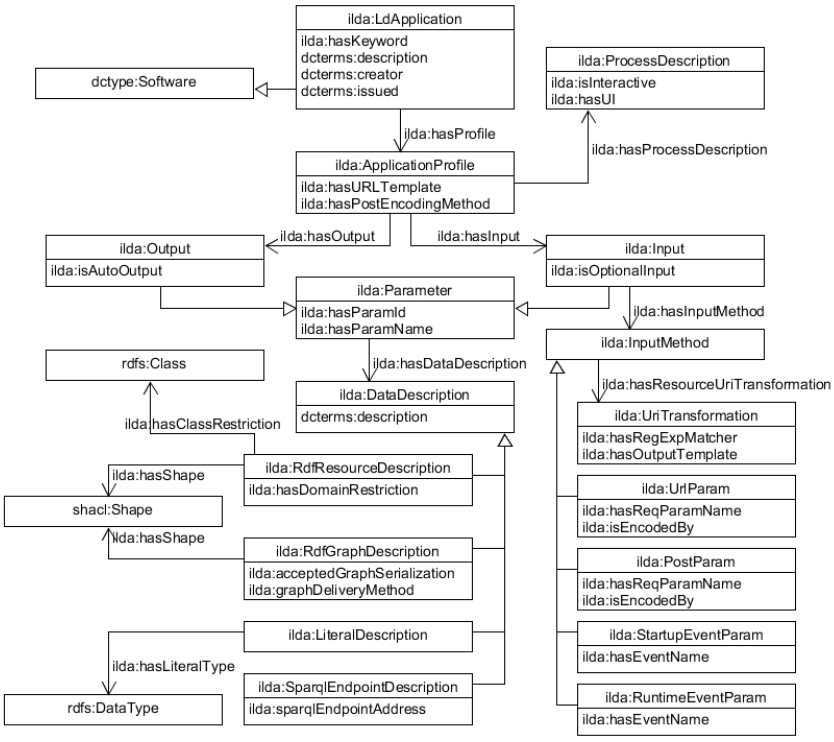

Figure 5.2: Graphical Representation of the ILDA Ontology

[60] https://www.w3.org/RDF/
[61] https://www.w3.org/OWL/
[62] http://dublincore.org/

5.4 Interfaces

To enable the required interchangeability of the IDLA systems' components (cf. Section 5.1), the ILDA framework defines standardized interfaces for communication between the components. The discussed interfaces are the ILDA communication platform, ILDA Service's API and the registration of applications.

5.4.1 ILDA Communication Platform

ILDA Communication Platform facilitates the communication between the ILDA Manager and integrated LD-based applications (*Ft01*). The platform enables the exchange of various types of interaction messages (cf. Section 5.3.1) between the manager and the applications. The communication takes places in two different forms.

The majority of input methods (URL parameters or POST parameters, cf. 5.3.2.3) require the application's inputs to be a part of the HTTP request for the integrated application. The specific information describing how the data should be provided in the request is contained in the application descriptions (cf. 5.3.2), and the ILDA Manager receives the information from the ILDA Service.

The rest of the communication between ILDA Manager and the integrated applications happens through the messaging platform based on the HTML5 Web messaging[63] technology.

5.4.2 ILDA Service API

The ILDA Service offers a standardized RESTful API[64] to enable the ILDA Manager to query for the possible next applications based on the selected data. The manager contacts the service whenever a new output is selected to get a new list of possible steps. In the HTTP POST request, the ILDA Manager sends information about the selected data and the ILDA service answers with the selection of integrated applications accepting the provided data as their inputs.

The API should be extensible (*Ft23*) and allow the introduction of additional information in the requests and responses. For instance, recommender functionality (*Ft15*) would add rankings of the list of possible next applications. Both the ILDA Manager and the ILDA service should ignore unknown additional data in the requests and responses.

5.4.3 Application Registration

The ILDA Application Registration is the process of how the new Linked Data-based applications are added into the ILDA framework. The specific workflow is very dependent on the specific deployment context; the descriptions could be added using an API, manually inputted through a web form, or through a configuration file. The addition could be automatic or could need to be approved. Many factors are influencing the registration process; therefore we have decided not to include any standardized registration processes to the framework. However, independent from the registration procedure we define

[63] https://www.w3.org/TR/webmessaging/
[64] https://en.wikipedia.org/wiki/Representational_state_transfer

a scheme for the description of tools characteristics related to their integration into the framework. The scheme should be used as part of the Linked Data-based application registration, but could also be utilized internally in the ILDA Service to store and process the definitions.

5.5 Summary

In this chapter, we have proposed the ILDA framework – a set of guidelines for the creation of ILDA systems with the aim to assist the developers in the implementation of ILDA systems and to enable the reusability of components of ILDA systems.

ILDA framework defines main components of ILDA systems, their interfaces and data models used in the inter-component communication. Based on the ILDA framework a prototypical ILDA System is implemented and evaluated (cf. Chapter 6).

CHAPTER 6

ILDA Prototype – Proof of the Concept

In the previous chapters, we have explored the concept of Linked Data-based application interoperability shaped into the ILDA framework as a potential complementary solution for the task-oriented adaptation need. The aim of this chapter is to show that the interoperability approach and specifically the ILDA framework are feasible. To do that, we implement a prototype of the ILDA system and test the system using a set of test scenarios targeted on the core functionalities of the ILDA system (cf. Section 3.3).

In Section 6.1, we discuss the specific targets of the feasibility assessment. Based on the targets we model test scenarios described in Section 6.2. The prototype of the ILDA system presented in Section 6.3 is then assessed according to the test scenarios. The results of the assessment are presented and discussed in Section 6.4.

6.1 Test Targets

The aim of the feasibility assessment is to show that an ILDA system implementing core features of ILDA systems and following the guidelines of ILDA framework can be created and can fulfill the core requirements described in Section 3.3. Figure 6.1 offers a recapitulation of core requirements and features from Section 3.3.1.

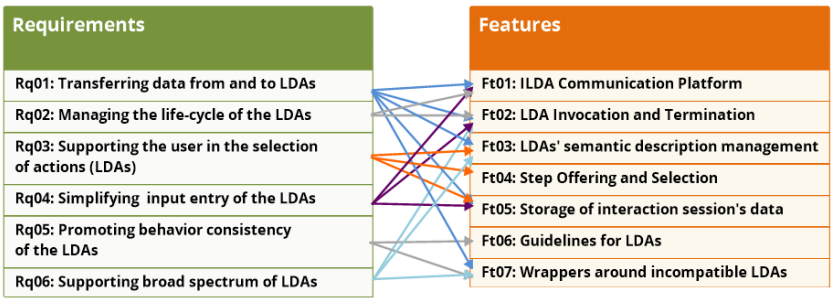

Figure 6.1: Core Requirements and Features of the ILDA framework

Specifically, the tests need to verify achievement of following targets:

1. **Data transfer** *(Rq01)*: The prototype is capable of collecting outputs of one application (using the ILDA Communication Platform – *Ft01,* cf. Section 5.4.1), storing the outputted data until the user decides about the next step (*Ft05*), and using the data as an input for another integrated application (inside of the request for the application – *Ft02*) according to the definition of application's input type and method (*Ft03,* cf. Section 5.3.2).
2. **Application selection** (*Rq03*): The prototype is capable of providing the user with a selection of possible next applications accepting inputs of the type of currently selected data (*Ft04*). The previously outputted data is stored in the application until the user makes the selection (Ft05). The list of offered actions is created using the Application descriptions (*Ft03,* cf. Section 5.3.2)
3. **Application wrappers** *(Ft07)*: The wrappers are able to adjust the behavior of integrated applications (i.e., changing link behavior and adding output functionality for the tools capable of creating outputs) in a way which would enable them to promote the consistency of applications' behaviors (*Rq05, Ft06*).
4. **Input methods:** The system supports all parameter delivery methods from Section 5.3.2.3 (GET parameters, POST parameters, event emission)
5. **Main data types:** The system supports data in the form of RDF resources and RDF graphs (cf. 5.3.1). RDF resources are the most widely supported type of data according to our survey (cf. Section 4.3), RDF graphs are expected as the main type of exchanged data by Lebo et al. [22].
6. **RDF resource restrictions:** The system takes into account the domain-based and type-based restrictions of RDF resources (cf. Section 5.3.2.3) in the application selection process.
7. **Multiple existing applications:** The system supports multiple applications recognized as integrable in Chapter 4.
8. **Application compositions:** Even though it is not part of the core requirements, we want to show that it is possible to create compositions of two applications and transfer data between them (*Rq13, Ft17*).

6.2 Feasibility Assessment Scenarios

Based on the test targets from Section 6.1 we have designed three test scenarios:

1. **Existing Applications:** In this scenario, we focus on integrating the applications recognized in Chapter 4, specifically the applications recognized as integrable but not requiring the manual inputs (cf. Section 4.3), i.e., Aemoo[65], DBpedia default viewer[66], LD viewer[67], LodLive[68], and RelFinder.[69] The system should enable transitions between

[65] http://wit.istc.cnr.it/aemoo/
[66] http://dbpedia.org/page/*
[67] http://ldv.dbpedia.org/
[68] http://en.lodlive.it/
[69] http://www.visualdataweb.org/relfinder/relfinder.php

the applications (between any pair of applications, with the exception of RelFinder, which does not support the creation of a wrapper enabling output emission, cf. Section 4.2). Successful execution of this scenario achieves the testing targets 1., 2., 3., 7., and partly 4. and 5.

2. **"Future Applications":** In this scenario, we create a mocked version of applications with native support for the ILDA system. The aim of this scenario is to test the features currently not supported by the existing applications:
   - *The "hello" and "hello-ack" messages* (cf. Section 5.3.1): A mocked application answers the "hello" message and updates its user interface.
   - *Restriction of the application selection based on the resource's type*: The system should correctly filter the selection based on the type of selected data. Application "Start" emits data of different types ("*test:Person*" and "*test:Place*"), "PersonApp" accepts data of type "*test:Person*," "PlaceApp" accepts a data of type "*test:Place*."
   - *Restriction of the application selection based on the resource's domain*: The "Start" application emits data from other domain than DBpedia (*http://testdomain.org*), "*DomainApp*" is the only application accepting data from the test domain.
   - RDF graph as data type: "Start" emitting an RDF graph, "GraphApp" accepting the graph as its input

   Successful execution of this scenario achieves the testing targets 1., 2., 4., 5., and 6.
3. **Application composition:** In this scenario, we create a composition of the DBpedia default view and the LODLive tool. The selection in one tool is automatically propagated to the other tool. Successful execution of this scenario achieves the testing target 8.
4. **Linked Widgets:** In this small scenario we test the integrability of Linked Widgets and its widgets as the first application of the interaction path in the system. The aim of this scenario is to show that the two approaches to adaptability can be combined. We also wanted to create a widget for the platform enabling the use of ILDA framework inside of the platform, but the functionality to add own widgets is currently not enabled.

6.3 Implementation

The prototype is implemented according to guidelines of the ILDA framework. The existing applications are altered by the implemented wrappers to enable emissions of data. Figure 6.2 shows how the user interface of DBpedia default view was adjusted by the addition of buttons enabling emission of data.

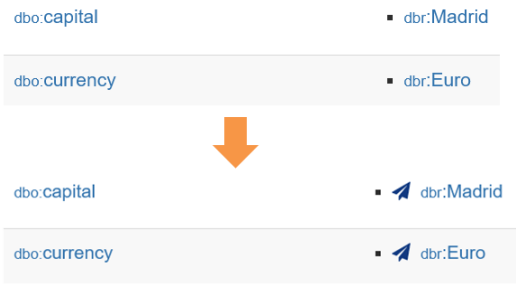

Figure 6.2: Addition of Buttons Emitting Data by the Wrapper in DBpedia Default View

When the user clicks on the data emission button, the application (or the wrapper in the case of applications without native support for ILDA systems) emits the data. The system receives the data and changes the list of available actions based on the received data. When the user chooses an action, the system facilitates transition to the selected application and provides it with the data. Figure 6.3 presents the user interface of implemented prototype and shows a transition between applications. In the presented interaction sequence, the user visits the default DBpedia view for the *dbr:Spain* resource and selects the *dbr:Madrid* as output. In the list of available actions, the user selects the LODLive applications to visualize the outputted data. After clicking on the action, the system facilitates transition to the selected application.

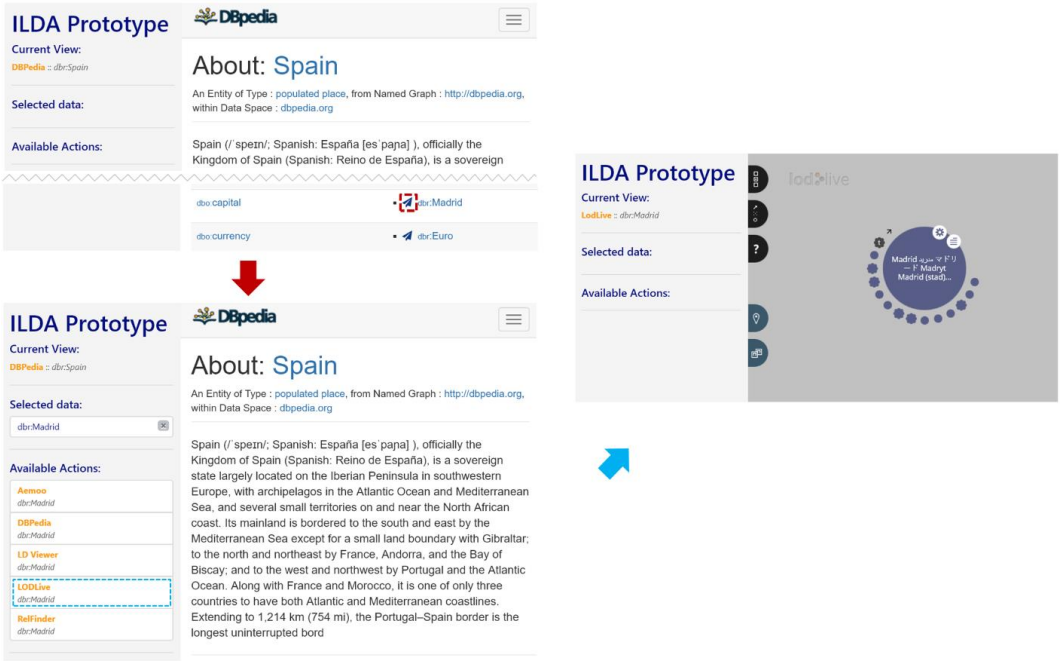

Figure 6.3: Transition between Applications

Considering the suitability of the technologies as well as our familiarity with them, we have selected the following technologies for the implementation of the ILDA framework components (except for the already existing Linked Data-based applications):

ILDA Navigator is a client-side web application written in TypeScript[70] language using the Angular 4 framework[71]. The ILDA Navigator's interface is implemented using the HTML[72] and CSS[73] languages. The application is managed using the NPM package manager[74] and Angular CLI[75].

[70] https://www.typescriptlang.org/
[71] https://angular.io/
[72] https://www.w3.org/html/
[73] https://www.w3.org/Style/CSS/

The wrapper module contains wrappers for all integrated applications and is implemented as a Google Chrome extension written in JavaScript using the Google Chrome Extension API[76] and jQuery library[77].

The server-side ILDA Service is written in Java languages utilizing multiple libraries: (a) Jena library[78] for handling of the ontology, (b) Jersey[79], a JAX-RS[80] implementation enabling the creation of RESTful[81] interfaces, (c) Jetty web server[82] for the deployment of the service and (d) Jackson[83] for transformation of data to and from JSON format. The service is built and started using the Maven software project management tool[84].

6.4 Results

Using the implemented prototype, we were able to successfully complete all the proposed scenarios (cf. Section 6.2) while achieving the test targets (cf. Section 6.1). The positive outcome of the feasibility assessment shows that it is possible to implement the core functions and features of ILDA Systems as proposed in Section 3.3 and shape them into an ILDA System according to the guidelines of the ILDA Framework (cf. Chapter 5).

The implemented prototype enables the user to navigate between the integrated applications and enables the users to select the next actions according to the expectations both with the real applications as in the "future applications" scenario. This has shown the ability of the ILDA framework to enable the creation of adapted multi-application interaction flows. We have also proved the ability of the ILDA framework to enable the creation of application compositions and to be combined with the Linked Widget platform (with some of its widgets).

Even though the tested functionality was derived from the targets of ILDA Systems (cf. Section 3.3.1), the feasibility assessment cannot directly prove the achievement of the targets and evaluate the contribution of the ILDA approach for the users. User evaluation studies will need to be conducted in specific contexts (types of tasks and sets of integrated applications) to evaluate the contribution. The studies will need to take into account the specific nature of the ILDA system's ecosystem. The results of the studies will likely be influenced by the selection of integrated tools and the specific sets of implemented ILDA system's functionality.

---

[74] https://www.npmjs.com/
[75] https://cli.angular.io/
[76] https://developer.chrome.com/extensions
[77] https://jquery.com/
[78] https://jena.apache.org/
[79] https://jersey.github.io/
[80] https://github.com/jax-rs
[81] https://en.wikipedia.org/wiki/Representational_state_transfer
[82] https://eclipse.org/jetty/
[83] https://github.com/FasterXML/jackson
[84] https://maven.apache.org/

6.5 Summary

The aim of this chapter was to provide a proof of the feasibility of the ILDA framework by the creation of a prototypical ILDA system implementing the core functionality (cf. Section 3.3) following the guidelines of the ILDA Framework (cf. Chapter 5). We have defined the targets of the feasibility assessment covering the whole core functionality of the ILDA system and selected additional features. Based on the assessment targets, we have proposed the test scenarios. A prototype of ILDA system has been implemented and assessed according to the testing scenarios. The prototype has successfully shown that the ILDA system based on the ILDA framework can be built and can fulfill the core functionality of ILDA systems. The assessment has shown that the system is capable of enabling transitions between integrated applications and by that facilitates the creation of adapted multi-application interaction flows. The positive test results provide the proof of the ILDA framework's feasibility and by that answer the research question 5.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusions

In the recent years, we can observe a rapid growth of the Web of Data. At the same time, even though at a slower pace, we are witnessing the evolution of Linked Data-based applications which, growing past their infancy of simple prototypes, are gradually improving in their ability to provide assistance to the users solving complex real-world problems.

Given the semantic nature of Linked Data, a growing body of research is investigating how such user support can be further enhanced by adaptations of user interfaces and interaction flows according to the content of the data, type of the solved task, user specifics and preferences, and other factors. In this thesis, we have explored how the concept of Linked Data-based applications' interoperability can be enabled and utilized to contribute to the task-oriented adaptation of Linked Data-based application environments.

To be able to conduct such exploration, we have defined the task-oriented adaptability and analyzed how it is targeted in current adaptation approaches. We have identified space for an approach which is complementary to the existing adaptation strategies and is based on a universal interoperability platform for the Linked Data-based applications. Following this finding and taking inspiration from related research, we have analyzed the required and optional characteristics of ILDA framework and ILDA systems which can be developed based on the ILDA framework. The results of the analysis were then used to shape the ILDA framework. The prototype of a basic ILDA system has been developed, and its positive assessment using defined test scenarios verified the feasibility of the ILDA framework. The answers to research questions we have proposed during this process are summarized in Table 7.1.

To our knowledge, there is currently no other adaptation approach based on the interoperability of LD-based applications except for the very limited applications with interoperability features described in Section 2.2.4. We intended to lay the foundations for further research in the field by providing multiple contributions: (a) a broad analysis of the characteristics of interoperability systems showing functionality required for the operation of interoperability systems and describing advanced features interoperability systems could provide, (b) the ILDA framework describing the reusable components and communication between them, intended to guide the creation of future

interoperability systems, (c) survey of integrable applications helping to orient in the applications' ecosystem, and (d) a proof of the feasibility of the ILDA approach.

Research Question	Answer
RQ 1 Existing strategies to task-oriented adaptability in the context of Linked Data-based applications	**Chapter 2.** We have discussed the concept of task-oriented adaptability and shown how it is targeted by contemporary LD-based approaches. Limitations of the approaches have been discussed, and we have shown how the general interoperability strategy could provide adaptability on the level of multi-application workflows and how such approach could support existing adaptation approaches.
RQ 2 Required and optional functionalities of the ILDA framework and systems enabling application interoperability and contributing to task-oriented adaptability	**Chapter 3.** Considering the related concepts from research areas of (a) interoperability in context of Linked Data (data integration and semantic web services), (b) interoperability of web applications, and (c) alternative task-oriented adaptation strategies, we have analyzed what targets can be achieved by the ILDA framework and ILDA systems, and what functionality can contribute to the achievement of the targets
RQ 3 Integrable LDAs	**Chapter 4.** Taking into account the requirements for applications to be used in the context of ILDA systems, we have conducted a survey concentrating on applications features affecting their compatibility with the ILDA framework.
RQ 4 ILDA framework's design supporting creation of task-adapted ILDA systems	**Chapter 5.** Building on the analysis of ILDA framework and ILDA systems characteristics (cf. Chapter 3) and the survey of integrable applications (cf. Chapter 4), we have designed the ILDA framework, a set of guidelines defining the internal exchangeable components of ILDA systems and enabling creation of various types of ILDA systems while reusing some of the components.
RQ 5 Feasibility of the ILDA approach	**Chapter 6.** Based on the guidelines of the ILDA framework we have created a prototypical version of an ILDA system implementing the core functionality of potential ILDA systems (cf. Section 3.2). Positive results of assessment designed to test the agreement with the proposed core functionality provided a proof of the feasibility of the ILDA framework.
Main RQ Could interoperability framework contribute to task-oriented adaptation?	We have shown that it is possible to design and implement a general interoperability approach following the same or similar targets as the alternative task-oriented adaptation approaches and contribute to their targets by enabling integration of applications.

Table 7.1: Answers to Research Questions

We see significant **benefits** of the general interoperability adaptation approach followed by the ILDA framework:

Complementary nature: ILDA systems are providing the adaptation functionality on the level of multi-application workflows or compositions and thus can be used synergistically with other adaptation approaches. Applications facilitating the alternative adaptation methods could be integrated into the ILDA system and used as steps in the interaction sequences. Development-oriented approaches and semantic mashup frameworks could adopt the interoperability technology and enable integration of standalone applications and their usage in the same way as in the case of existing components and widgets.

Flexible integration of new applications: Integration of a new Linked Data-based application to the ILDA framework should be usually simple. An ILDA ontology-based description of the application needs to be created. If the application does not support the communication with ILDA systems natively, a wrapper injecting required functionality (mainly the output emission ability) needs to be created. In contrast to this integration process, the creation of web application mashups proposed by Matono et al. [72] requires manual creation of configurations for every application composition. Also, the addition of new functionality to the majority of alternative adaptation approaches requires changes in the source code of the tools and deployment of the updated version (with potential downtimes and interrupted sessions of the users).

Simple adjustments of applications to the ILDA framework: The adaptations of applications required from the developers to make the application compatible with ILDA framework are simple and should not require significant amounts of effort. Our implementations of the wrappers consist of just a few lines of code.

Reusability of applications: Once a Linked Data-based application is compatible with the framework, it can be used in multiple different workflows, compositions and in various contexts.

Lowering barriers for users: ILDA systems assist the users in the selection of actions (combinations of application and its inputted data) and in the execution of transitions between applications. In this process, the systems do not expect the user to understand Linked Data-related technological concepts in accordance with the calls for support for users without experience with Linked Data [8].

Working implementation: The ILDA prototype provides a first working implementation of the universal interoperability approach to the task-oriented adaptation, which could be utilized as a basis for future extension.

We are also aware of the **open issues** of the ILDA framework and the universal interoperability approach to adaptation in general:

Need for evaluations: In this thesis, we have verified the feasibility of our approach by showing that the implemented prototype fulfills the core requirements for ILDA systems assuming that the fulfillment of requirements leads to the achievement of targets from which the requirements were derived. However, to assess the real contribution of the ILDA approach for the users, further evaluations will need to be performed. Such evaluations will need to take into account the specific context in which they would take place, such as the selection of the LD-based applications, the familiarity of the users with the integrated

applications, the specifics of the task (e.g., simple tasks being solvable in a single application without the need for application interoperability), what the evaluation will compare (the framework compared with a single tool or with the same set of applications but without the employment of the ILDA system). Standard usability evaluation methods, such as the SUS usability scale [91] would need to be adjusted to evaluate an ILDA system since it is not a standalone application but rather a whole ecosystem of integrated tools combined with the functionality provided by the ILDA system.

Potentially ineffective action selection process: The action selection functionality employed while navigating the integrated applications offers the possible actions in an equalitarian way, what could be ineffective in environments with many integrated applications (e.g., the user would need to scroll through the list of actions to find the right one) when the selection of the tools is dependent on the context or when some of the tools are used more frequently than others. The issue could be mitigated or solved by introducing a recommender system for the step offerings.

Dependence on the integrated applications: ILDA systems are depending on the existence and availability of integrated Linked Data-based applications (and their potential wrappers). The usability of the ILDA system depends on the usability of the applications and their conformance with the ILDA framework guidelines. Naturally, there are right now no applications directly supporting the ILDA framework, but, as we have seen in the survey of integrable Linked Data-based applications, even the overall number of Linked Data-based applications (specifically visualization and exploration systems) available online is relatively small (we have identified 17 tools), and the number of integrable applications even smaller (overall 13 applications could be integrated, five applications were most suitable for the integration).

We believe that the presented issues can be solved or mitigated to the extent that the benefits of the interoperability approach outweigh them and the framework could be a viable addition to the existing adaptation strategies.

The ILDA framework or similar approaches adopting the idea of Linked Data-based application interoperability could be a relevant addition to the Linked Data ecosystem, where the idea of *Linked Data* is extended to "*linked applications*." We have described, how the application linking mechanism could function and shown that the core of our interoperability mechanism is feasible. Evolving into a standard for application interoperability it could become a superset of the *Linked Open Visualizations (LOVIZ)* suggested by Atemezing & Troncy [36] and the "*standardized framework for human interaction with data across the Semantic Web in the future"* envisioned by Lukovnikov et al. [31] integrating a broad spectrum of applications and services, and enabling creation of various types of application compositions and interaction flows supporting the users in their analytical, exploratory or even ordinary daily-life activities. In the 2001 article [2] while outlining the semantic web's vision, Berners Lee et al. suggested that "*the real power of the Semantic Web will be realized when people create many programs that collect information and exchange the results*." We do believe that – alongside the Semantic Web services – the ILDA framework or similar application interoperability platforms could contribute to the realization of this vision.

7.2 Future Work

This thesis has aimed to map the potential of a universal interoperability approach to task-oriented adaptation and to propose an initial version of the approach in the form of the ILDA framework and the implemented prototype of ILDA system. While being able to show the rudimentary feasibility of the approach, the ILDA framework, to be applicable in real-world scenarios, would require additional work in multiple directions: (a) providing more than just the basic functionality of the prototype, (b) evaluations of potential deployments in different areas, and (c) research in the area of synergy with alternative adaptation approaches.

7.2.1 Improved Functionality

The prototypical implementation of the framework fulfills only the core requirements for ILDA system (cf. Section 3.3.2). For real-world deployments, but also for further evaluations, additional functionality would need to be implemented by the system. We have emphasized the extensibility of the ILDA framework as an important concept enabling future additions of functionality without affecting the existing ILDA systems. The complete set of potential advanced functionality of the ILDA systems is mapped in Sections 3.3.2 and 3.3.3. Here, we want to highlight few categories of the potential improvements:

Extended support for input and output types and methods: The prototype of ILDA system described in Chapter 6 contains only a subset of the possible input and output types and methods described in Section 5.3.2. Also, the wrappers implemented as part of the prototype did not support the applications with manual inputs described in Chapter 4. Based on the set of Linked Data-based applications which will need to be integrated into future versions of ILDA systems, additional types and input methods will need to be supported.

New types of integrated tools: Integration of *Semantic Web services* handled similarly to the process widgets of Linked Widget platform [17] could enrich the ILDA systems with relevant functionalities, such as data integration, data enrichment, data processing, and validation of data. The *manual steps* would significantly extend the range of supported applications (e.g., desktop application) by enabling the users to manually download inputs for the application and later upload outputs of the application back into the ILDA system.

History and workflow reusability: Visualization of the history of interaction with the ILDA system could enable the user to orient in the workflow and can serve as a basis for additional features like the documentation of internal steps of applications (e.g., which relations in the graph visualization were expanded), saving and sharing of the interaction history, or introducing automation of the workflows (replaying internal action in the tool, capturing the meaning of the relation between inputs and output of an application)

Recommendation of actions: The possible actions could be recommended according to the user's context, ratings of the tools and other variables. The recommendation could influence the order of the steps, how are they displayed or even remove some steps from the list.

7.2.2 Evaluation and Deployment in Specific Environments

We have shown the feasibility of the framework. However, we were able to make only assumptions about the benefits of the framework. To validate these assumptions multiple evaluations need to be conducted to assess the benefits (usability, improvements in the users' ability to solve the tasks) of the framework in comparison to other approaches. As the framework enables different modes of operation (a navigator facilitating transitions between tools, application compositions) and various selections of the integrated LD-based applications, framework's performance should be evaluated in different evaluation setups covering these variations as well as the different types of users and performed tasks (such as information seeking, exploratory search).

7.2.3 Synergy with Other Adaptation Approaches

While being a complementary approach to other adaptation approaches enabling the users to use Linked Data-based applications (both the ones employing other adaptation strategies and standard applications) as parts of multi-application workflows, the ILDA framework could also be utilized to contribute internally to the development-oriented adaptation approaches and the semantic mashup frameworks. The framework could enable the creation of development framework components or mashup widgets containing existing Linked Data-based applications. As part of the prototypical implementation of ILDA system, we have shown that it is possible to create application compositions, but additional steps towards the integration into the particular development and mashup frameworks will be required.

A significant move in the direction of simplifying the utilization of ILDA framework would be a further modularization of the framework and employment of Web Components technologies[85] for the packaging of the framework components. Web components are a family of future web standards (currently in the state of a "working draft") enabling the creation and deployment of custom fully-featured DOM elements [92]. Such custom elements wrap both the visual aspect of the module and its functionality and enable the reuse of components across websites and web applications. In the context of ILDA framework, the web components could be employed to define a container for the integrated application with defined communication interface enabling very simple integration of ILDA-compatible Linked Data-based applications into any web page or web application. Moreover, the web components could be used in the ILDA framework as an additional type of integrated tool along the Linked Data-based applications and other potentially supported tools described in Section 7.2.1.

[85] https://www.w3.org/standards/techs/components

Appendix A

ILDA Ontology

The ILDA Ontology is a component of the ILDA framework (cf. Chapter 5) introducing the vocabulary for the definitions of integrated Linked Data-based applications. The concepts of ILDA Ontology are described throughout the Section 5.3.2, and a graphical representation of the ontology is presented as Figure 5.2 in Section 5.3.2.5 along additional information about the ontology. In this appendix, we present the serialized form of the vocabulary in a Turtle serialization format.[86]

```
@prefix : <http://ilda.org/ontology#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcam: <http://purl.org/dc/dcam/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix shacl: <http://www.w3.org/ns/shacl#> .
@prefix dctype: <http://purl.org/dc/dcmitype/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@base <http://ilda.org/ontology> .

<http://ilda.org/ontology> rdf:type owl:Ontology ;
                        owl:versionIRI <http://ilda.org/ontology/0.0.1> ;
                        rdfs:label "ILDA Ontology" .

#################################################################
#    Datatypes
#################################################################

xsd:date rdf:type rdfs:Datatype .

#################################################################
#    Object Properties
#################################################################

:hasClassRestriction rdf:type owl:ObjectProperty ;
                     rdfs:subPropertyOf owl:topObjectProperty ;
                     rdfs:domain :RdfResourceDescription ;
                     rdfs:range rdfs:Class .
```

[86] https://www.w3.org/TR/turtle/

```
:hasInput rdf:type owl:ObjectProperty ;
          rdfs:subPropertyOf owl:topObjectProperty ;
          rdfs:domain :ApplicationProfile ;
          rdfs:range :Input .

:hasInputMethod rdf:type owl:ObjectProperty ;
                rdfs:subPropertyOf owl:topObjectProperty ;
                rdfs:domain :Input ;
                rdfs:range :InputMethod .

:hasOutput rdf:type owl:ObjectProperty ;
           rdfs:subPropertyOf owl:topObjectProperty ;
           rdfs:domain :ApplicationProfile ;
           rdfs:range :Output .

:hasProcessDescription rdf:type owl:ObjectProperty ;
                       rdfs:subPropertyOf owl:topObjectProperty ;
                       rdfs:domain :ApplicationProfile ;
                       rdfs:range :ProcessDescription .

:hasProfile rdf:type owl:ObjectProperty ;
            rdfs:subPropertyOf owl:topObjectProperty ;
            rdfs:domain :LdApplication ;
            rdfs:range :ApplicationProfile .

:hasResourceUriTransformation rdf:type owl:ObjectProperty ;
                              rdfs:subPropertyOf owl:topObjectProperty ;
                              rdfs:domain :InputMethod ;
                              rdfs:range :UriTransformation .

:hasDataDescription   rdf:type owl:ObjectProperty ;
                      rdfs:subPropertyOf owl:topObjectProperty ;
                      rdfs:domain :Parameter ;
                      rdfs:range :DataDescription .

:hasLiteralType rdf:type owl:ObjectProperty ;
                         rdfs:subPropertyOf owl:topObjectProperty ;
                         rdfs:domain :LiteralDescription ;
                         rdfs:range rdfs:DataType .

:hasShape   rdf:type owl:ObjectProperty ;
            rdfs:subPropertyOf owl:topObjectProperty ;
            rdfs:domain [ rdf:type owl:Class ;
                owl:unionOf ( :RdfGraphDescription
                              :RdfResourceDescription
                            )
                ] ;
            rdfs:range shacl:Shape .
```

```
#################################################################
#    Data properties
#################################################################

:acceptedGraphSerialization rdf:type owl:DatatypeProperty ;
                            rdfs:subPropertyOf owl:topDataProperty ;
                            rdfs:domain :RdfGraphDescription ;
                            rdfs:range xsd:string .


:graphDeliveryMethod rdf:type owl:DatatypeProperty ;
                     rdfs:subPropertyOf owl:topDataProperty ;
                     rdfs:domain :RdfGraphDescription ;
                     rdfs:range xsd:string .

:hasDomainRestriction rdf:type owl:DatatypeProperty ;
                      rdfs:subPropertyOf owl:topDataProperty ;
                      rdfs:domain :RdfResourceDescription ;
                      rdfs:range xsd:string .

:hasID rdf:type owl:DatatypeProperty ;
       rdfs:subPropertyOf owl:topDataProperty ;
       rdf:type owl:FunctionalProperty ;
       rdfs:domain :Parameter ;
       rdfs:range xsd:string .

:hasKeyword rdf:type owl:DatatypeProperty ;
            rdfs:subPropertyOf owl:topDataProperty ;
            rdfs:domain :LdApplication ;
            rdfs:range xsd:string .

:hasOutputTemplate rdf:type owl:DatatypeProperty ;
                   rdfs:subPropertyOf owl:topDataProperty ;
                   rdfs:domain :UriTransformation ;
                   rdfs:range xsd:string .

:hasRegExpMatcher rdf:type owl:DatatypeProperty ;
                  rdfs:subPropertyOf owl:topDataProperty ;
                  rdfs:domain :UriTransformation ;
                  rdfs:range xsd:string .

:hasUI rdf:type owl:DatatypeProperty ;
       rdfs:subPropertyOf owl:topDataProperty ;
       rdfs:domain :ProcessDescription ;
       rdfs:range xsd:boolean .

:hasURLTemplate rdf:type owl:DatatypeProperty ;
                rdfs:subPropertyOf owl:topDataProperty ;
                rdfs:domain :ApplicationProfile ;
                rdfs:range xsd:string .

:isInteractive rdf:type owl:DatatypeProperty ;
               rdfs:subPropertyOf owl:topDataProperty ;
               rdfs:domain :ProcessDescription ;
               rdfs:range xsd:boolean .
```

```
:hasEventName       rdf:type owl:DatatypeProperty ;
                    rdfs:subPropertyOf owl:topDataProperty ;
                    rdfs:domain [ rdf:type owl:Class ;
                                  owl:unionOf ( :RuntimeEventParam
                                                :StartupEventParam
                                              )
                                ] ;
                    rdfs:range xsd:string .

:hasParamId rdf:type owl:DatatypeProperty ;
                    rdfs:subPropertyOf owl:topDataProperty ;
                    rdfs:domain :Parameter ;
                    rdfs:range xsd:string .

:hasParamName   rdf:type owl:DatatypeProperty ;
                rdfs:subPropertyOf owl:topDataProperty ;
                rdfs:domain :Parameter ;
                rdfs:range xsd:string .

:hasPostEncodingMethod  rdf:type owl:DatatypeProperty ;
                        rdfs:subPropertyOf owl:topDataProperty ;
                        rdfs:domain :ApplicationProfile ;
                        rdfs:range xsd:string .

:hasReqParamName    rdf:type owl:DatatypeProperty ;
                    rdfs:subPropertyOf owl:topDataProperty ;
                    rdfs:domain [ rdf:type owl:Class ;
                           owl:unionOf ( :PostParam
                                         :URLParam
                                       )
                         ] ;
                    rdfs:range xsd:string .

:isAutoOutput rdf:type owl:DatatypeProperty ;
              rdfs:subPropertyOf owl:topDataProperty ;
              rdfs:domain :Output ;
              rdfs:range xsd:boolean .

:isEncodedBy rdf:type owl:DatatypeProperty ;
             rdfs:subPropertyOf owl:topDataProperty ;
             rdfs:domain [ rdf:type owl:Class ;
                           owl:unionOf (  :PostParam
                                          :URLParam
                                       )
                         ] ;
             rdfs:range xsd:string .

:isOptionalInput    rdf:type owl:DatatypeProperty ;
                    rdfs:subPropertyOf owl:topDataProperty ;
                    rdfs:domain :Input ;
                    rdfs:range xsd:boolean .

:sparqlEndpointAddress    rdf:type owl:DatatypeProperty ;
                          rdfs:subPropertyOf owl:topDataProperty ;
                          rdfs:domain :SparqlEndpointDescription ;
                          rdfs:range xsd:anyURI .
```

```
#################################################################
#    Classes
#################################################################

:ApplicationProfile rdf:type owl:Class .

:DataDescription rdf:type owl:Class .

:Input rdf:type owl:Class ;
       rdfs:subClassOf :Parameter .

:InputMethod rdf:type owl:Class .

:LdApplication rdf:type owl:Class ;
               rdfs:subClassOf dctype:Software .

:LiteralDescription rdf:type owl:Class ;
                    rdfs:subClassOf :DataDescription .
:Output rdf:type owl:Class ;
        rdfs:subClassOf :Parameter .

:Parameter rdf:type owl:Class .


:PostParam rdf:type owl:Class ;
           rdfs:subClassOf :InputMethod .

:ProcessDescription rdf:type owl:Class .

:RdfGraphDescription rdf:type owl:Class ;
                     rdfs:subClassOf :DataDescription .

:RdfResourceDescription rdf:type owl:Class ;
                        rdfs:subClassOf :DataDescription .

:RuntimeEventParam rdf:type owl:Class ;
                   rdfs:subClassOf :InputMethod .

:SparqlEndpointDescription rdf:type owl:Class ;
                           rdfs:subClassOf :DataDescription .

:StartupEventParam rdf:type owl:Class ;
                   rdfs:subClassOf :InputMethod .

:URLParam rdf:type owl:Class ;
          rdfs:subClassOf :InputMethod .

:UriTransformation rdf:type owl:Class .
```

Bibliography

[1] Tim Berners-Lee, "Semantic Web road map," *W3C*, 1998. [Online]. Available: https://www.w3.org/DesignIssues/Semantic.html. [Accessed: 10-Oct-2017].

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific american*, vol. 284, no. 5, pp. 28–39, 2001.

[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009.

[4] T. Berners-lee, "Linked Data - Design Issues," *W3C*, 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 10-Oct-2017].

[5] A. Abele, J. P. McCrae, P. Buitelaar, A. Jentzsch, and R. Cyganiak, "Linking Open Data cloud diagram 2017." [Online]. Available: http://lod-cloud.net/. [Accessed: 10-Oct-2017].

[6] A. Bernstein, J. Hendler, and N. Noy, "A new look at the semantic web," *Communications of the ACM*, vol. 59, no. 9, pp. 35–37, Aug. 2016.

[7] J. M. Brunetti, S. Auer, R. García, J. Klímek, and M. Nečaský, "Formal Linked Data Visualization Model," in *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, 2013, pp. 309–318.

[8] A.-S. Dadzie, M. Rowe, and D. Petrelli, "Hide the Stack: Toward Usable Linked Data," *The Semantic Web: Research and Applications*, pp. 93–107, 2011.

[9] A. Dadzie and M. Rowe, "Approaches to visualising Linked Data: A survey," *Semantic Web*, vol. 2, no. 2, pp. 89–124, 2011.

[10] M. Schraefel and D. Karger, "The Pathetic Fallacy of RDF," in *International Workshop on the Semantic Web and User Interaction (SWUI)*, 2006.

[11] A. S. Dadzie and E. Pietriga, "Visualisation of Linked Data - Reprise," *Semantic Web*, vol. 8, no. 1, pp. 1–21, 2017.

[12] N. Bikakis and T. Sellis, "Exploration and Visualization in the Web of Big Linked Data: A Survey of the State of the Art," in *Proceedings of the Workshops of the (EDBT/ICDT) 2016 Joint Conference, (EDBT/ICDT) Workshops*, 2016.

[13] N. Marie and F. Gandon, “Survey of linked data based exploration systems,” *IESD 2014 - Intelligent Exploitation of Semantic Data*, 2014.

[14] A. G. Nuzzolese, V. Presutti, A. Gangemi, S. Peroni, and P. Ciancarini, “Aemoo: Linked Data exploration based on Knowledge Patterns,” *Semantic Web*, vol. 8, no. 1, pp. 87–112, 2017.

[15] A. Khalili, “Linked data reactor: A framework for building reactive linked data applications,” *LIME/SemDev@ ESWC*, 2016.

[16] G. Vega-Gorgojo, L. Slaughter, M. Giese, S. Heggestøyl, A. Soylu, and A. Waaler, “Visual query interfaces for semantic datasets: An evaluation study,” *Web Semantics: Science, Services and Agents on the World Wide Web*, no. 39, pp. 81–96, 2016.

[17] T.-D. Trinh, P. Wetz, B. Do, A. Anjomshoaa, E. Kiesling, and A. M. Tjoa, “Linked Widgets Platform: Lowering the Barrier for Open Data Exploration,” in *European Semantic Web Conference*, 2014, pp. 171–182.

[18] F. Gandon, M. Sabou, and H. Sack, “Weaving a Web of linked resources,” *Semantic Web*, vol. 8, no. 6, pp. 767–772, 2017.

[19] “Open PHACTS: semantic interoperability for drug discovery,” *Drug Discovery Today*, vol. 17, no. 21, pp. 1188–1198, Nov. 2012.

[20] Y. Gil, F. Michel, V. Ratnakar, and M. Hauder, “A Semantic , Task-Centered Collaborative Framework for Science,” in *International Semantic Web Conference*, 2015, pp. 58–61.

[21] R. Peinl, “Semantic Web: State of the Art and Adoption in Corporations,” *KI-Künstliche Intelligenz*, vol. 30, no. 2, pp. 131–138, 2016.

[22] T. Lebo, N. Del Rio, P. Fisher, and C. Salisbury, “A five-star rating scheme to assess application seamlessness,” *Semantic Web*, vol. 8, no. 1, pp. 43–63, 2017.

[23] K. Bakshi and D. R. Karger, “Semantic Web Applications,” in *Proceedings of the ISWC 2005 Workshop on End User Semantic Web Interaction, Galway, Ireland*, 2005.

[24] D. Burkhardt, T. Ruppert, and K. Nazemi, “Towards process-oriented Information Visualization for supporting users,” in *2012 15th International Conference on Interactive Collaborative Learning (ICL)*, 2012.

[25] M. Luggen, A. Gschwend, B. Anrig, and P. Cudré-mauroux, “Uduvudu : a Graph-Aware and Adaptive UI Engine for Linked Data,” *Proceedings of LDOW*, 2015.

[26] J. Koch and T. Franz, “LENA - Browsing RDF data more complex than FOAF,” in *ISWC*, 2008, pp. 165–166.

[27] S. Battle, D. Wood, J. Leigh, and L. Ruth, “The Callimachus project: RDFa as a web template language,” in *Proceedings of the Third International Conference on Consuming Linked Data*, 2012, vol. 905, pp. 1–14.

[28] M. Voigt, S. Pietschmann, and K. Meißner, “Towards a Semantics-Based , End-User-Centered Information Visualization Process,” in *Proc. of the 3rd international workshop on semantic models for adaptive interactive systems (SEMAIS 2012)*, pp. 1–12.

[29] N. Bikakis and M. Papastefanatos, “rdf: SynopsViz–A Framework for Hierarchical Linked Data Visual Exploration and Analysis,” in *European Semantic Web Conference*, 2014.

[30] D. Lukovnikov, C. Stadler, D. Kontokostas, S. Hellmann, and J. Lehmann, “DBpedia Viewer-An Integrative Interface for DBpedia Leveraging the DBpedia Service Eco System,” *Proceedings of the 7th Workshop on Linked Data on the Web*, 2014.

[31] D. Lukovnikov, C. Stadler, and J. Lehmann, “LD viewer - linked data presentation framework,” in *Proceedings of the 10th International Conference on Semantic Systems - SEM ’14*, 2014, pp. 124–131.

[32] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. D. M. Silveira Neto, Y. C. Cavalcanti, and S. R. D. L. Meira, “Twenty-eight years of component-based software engineering,” *Journal of Systems and Software*, vol. 111, pp. 128–148, Jan. 2016.

[33] K. Nazemi, *Adaptive Semantics Visualization*. Springer, 2016.

[34] N. Ide and J. Pustejovsky, “What Does Interoperability Mean, Anyway ? Toward an Operational Definition of Interoperability for Language Technology,” in *Proceedings of the Second International Conference on Global Interoperability for Language Resources.*, 2010.

[35] C. Weaver, “Building Highly-Coordinated Visualizations in Improvise,” in *IEEE Symposium on Information Visualization*, 2004, pp. 159–166.

[36] G. A. Atemezing and R. Troncy, “Towards Interoperable Visualization Applications Over Linked Data,” in *Talk Given at the 2nd European Data Forum (EDF), Dublin, Ireland*, 2013.

[37] I. Hickson, “HTML5 Web Messaging,” *W3C*, 2015. [Online]. Available: https://www.w3.org/TR/webmessaging/. [Accessed: 01-Oct-2017].

[38] T. Berners-Lee, “Linked Open Data,” *W3C*, 2008. [Online]. Available: https://www.w3.org/2008/Talks/0617-lod-tbl. [Accessed: 01-Oct-2017].

[39] C. Bizer, “The emerging web of linked data,” *IEEE Intelligent Systems*, vol. 24, no. 5, pp. 87–92, 2009.

[40] D. Brickley and R. V. Guha, “RDF Schema 1.1,” *W3C*, 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/. [Accessed: 01-Oct-2017].

[41] “W3C Semantic Web Activity,” *W3C*. [Online]. Available: https://www.w3.org/2001/sw/. [Accessed: 01-Oct-2017].

[42] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou, “Ontology visualization methods---a survey,” *ACM Computing Surveys }(CSUR)*, vol. 39, no. 4, p. 10, Nov. 2007.

[43] F. Alahmari, J. A. Thom, L. Magee, and W. Wong, “Evaluating semantic browsers for consuming linked data,” in *Proceedings of the Twenty-Third Australasian Database Conference (ADC 2012)*, 2012, vol. 124, pp. 89–98.

[44] C. Hayes, S. Decker, B. Heitmann, R. Cyganiak, A. Harth, K. Hose, and R. Schenkel, “Architecture of Linked Data Applications,” in *Linked Data Management: Principles and Techniques*, CRC Press (Taylor & Francis), 2014.

[45] A. Khalili, A. Loizou, and F. van Harmelen, “Adaptive Linked Data-Driven Web Components: Building Flexible and Reusable Semantic Web Interfaces,” in *Lecture Notes in Computer Science*, vol. 9678, 2016, pp. 677–692.

[46] N. Subramanian and L. Chung, “Software Architecture Adaptability : An NFR Approach,” in *Proceedings of the IWPSE ’01 Proceedings of the 4th International Workshop on Principles of Software Evolution*, 2001, pp. 52–61.

[47] B. Tekinerdogan and M. Aksit, “Adaptability in Object-Oriented Software Development: Workshop report,” in *10th European Conference on Object-Oriented Programming*, 1996.

[48] A. Thomas, “Site Flows vs. User Flows: When to Use Which,” 2015. [Online]. Available: http://uxmovement.com/wireframes/site-flows-vs-user-flows-when-to-use-which/. [Accessed: 01-Oct-2017].

[49] F. Alahmari, “User Interfaces Supporting Entity Search for Linked Data,” RMIT University Melbourne, Victoria, Australia., 2014.

[50] R. García-Castro, A. Gómez-Pérez, and Ó. Muñoz-García, “The Semantic Web Framework: a component-based framework for the development of Semantic Web applications,” in *Database and Expert Systems Application, 2008. DEXA ’08. 19th International Workshop on*, 2008, pp. 185–189.

[51] E. Pietriga, C. Bizer, D. Karger, and R. Lee, “Fresnel : A Browser-Independent Presentation Vocabulary for RDF,” in *International Semantic Web Conference*, 2007, vol. 4273, pp. 158–171.

[52] A. Bunt, G. Carenini, and C. Conati, “Adaptive Content Presentation for the Web,” in *The Adaptive Web*, Springer-Verlag, 2007, pp. 409–432.

[53] A. Khalili and K. A. de Graaf, “Linked Data Reactor: Towards Data-aware User Interfaces,” in *Proceedings of the 13th International Conference on Semantic Systems, SEMANTiCS*, 2017.

[54] J. M. Brunetti, R. Gil, and R. García, “Facets and Pivoting for Flexible and Usable Linked Data Exploration,” in *Interacting with Linked Data Workshop*, 2012, vol. 12, pp. 22–35.

[55] D. Fichter, “What Is a Mashup ?,” in *Library mashups: Exploring new ways to deliver library data*, Information Today, Incorporated, 2009, pp. 3–17.

[56] M. Jarrar and M. D. Dikaiakos, “MashQL : A Query-by-Diagram Topping SPARQL,” in *Proceedings of the 2nd international workshop on Ontologies and information systems for the semantic web*, 2008, pp. 89–96.

[57] D. Le-Phuoc, A. Polleres, C. Morbidoni, M. Hauswirth, and G. Tummarello, “Rapid prototyping of semantic mash-ups through semantic web pipes,” in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 581–590.

[58] M. Cáceres, “Packaged Web Apps (Widgets) - Packaging and XML Configuration (Second Edition),” *W3C*, 2012. [Online]. Available: https://www.w3.org/TR/widgets/. [Accessed: 01-Oct-2017].

[59] H. Kubicek, R. Cimander, and H. J. Scholl, “Layers of Interoperability,” in *Organizational interoperability in e-government: lessons from 77 European good-practice cases*, Springer Science & Business Media, 2011, pp. 85–97.

[60] K. H. Veltman, “Syntactic and semantic interoperability: New approaches to knowledge and the semantic web,” *New Review of Information Networking*, vol. 7, no. 1, pp. 159–183, Jan. 2001.

[61] N. F. Noy and D. L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology,” *Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory*. 2001.

[62] C. Binding and D. Tudhope, “Improving interoperability using vocabulary linked data,” *International Journal on Digital Libraries*, vol. 17, no. 1, pp. 5–21, 2016.

[63] “Linked Data - Connect Distributed Data across the Web.” [Online]. Available: http://linkeddata.org/. [Accessed: 01-Oct-2017].

[64] M. Lenzerini, “Data Integration: A Theoretical Perspective,” in *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, 2002, pp. 233–246.

[65] P. Neish, “Linked data: what is it and why should you care?,” *Australian Library Journal*, vol. 64, no. 1, pp. 3–10, 2015.

[66] T. Moser, S. Biffl, W. D. Sunindyo, and D. Winkler, “Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains,” in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 2010, pp. 352–359.

[67] L. Kazemzadeh, M. R. Kamdar, O. D. Beyan, S. Decker, and F. Barry, “LinkedPPI: Enabling Intuitive, Integrative Protein-protein Interaction Discovery,” in *Proceedings of the 4th International Conference on Linked Science - Volume 1282*, 2014, pp. 48–59.

[68] D. Welter, J. MacArthur, J. Morales, T. Burdett, P. Hall, H. Junkins, A. Klemm, P. Flicek, T. Manolio, L. Hindorff, and H. Parkinson, “The NHGRI GWAS Catalog, a curated resource of SNP-trait associations,” *Nucleic Acids Research*, vol. 42, no. D1, pp. D1001–D1006, 2014.

[69] W3C, “Web Services Glossary.” [Online]. Available: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice. [Accessed: 01-Oct-2017].

[70] H. Nacer and D. Aissani, “Semantic web services: Standards, applications, challenges and solutions,” *Journal of Network and Computer Applications*, vol. 44, pp. 134–151, 2014.

[71] A. Taivalsaari and T. Mikkonen, “Mashups and modularity: Towards secure and reusable web applications,” in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops*, 2008, vol. 553, pp. 25–33.

[72] A. Matono, A. Nakamura, and I. Kojima, “A Mashup Tool for Cross-Domain Web Applications Using HTML5 Technologies,” in *Web Technologies and Applications: 13th Asia-Pacific Web Conference, APWeb 2011, Beijing, Chiina, April 18-20, 2011. Proceedings*, 2011, pp. 382–385.

[73] F. Giunchiglia, S. R. Ojha, and S. Das, “SemUI: A Knowledge Driven Visualization of Diversified Data,” *Proceedings - IEEE 11th International Conference on Semantic Computing, ICSC 2017*, pp. 234–241, 2017.

[74] T.-D. Trinh, “Mashup-based Linked Data Integration,” TU Wien, 2016.

[75] H. Lampesberger, “Technologies for Web and cloud service interaction: a survey,” *Service Oriented Computing and Applications*, vol. 10, no. 2, pp. 71–110, 2016.

[76] W. Lidwell, K. Holden, and J. Butler, *Universal principles of design: 125 ways to enhance usability, influence perception, increase appeal, make beter design decisions, and teach through design*. Rockport Pub, 2010.

[77] V. Codina and L. Ceccaroni, “A Recommendation System for the Semantic Web,” in *Distributed Computing and Artificial Intelligence. Advances in Intelligent and Soft Computing*, vol. 79, Berlin, Heidelberg: Springer, 2010, pp. 45–52.

[78] A. Yildiz, B. Aktemur, and H. Sözer, “Rumadai: A plug-in to record and replay client-side events of web sites with dynamic content,” in *2012 2nd International Workshop on Developing Tools as Plug-Ins, TOPI 2012 - Proceedings*, 2012, pp. 88–89.

[79] C. North and B. Shneiderman, “Snap-together visualization,” in *Proceedings of the working conference on Advanced visual interfaces - AVI ’00*, 2000, pp. 128–135.

[80] J. Howse, “Review of A Five-Star Rating Scheme to Assess Application Seamlessness,” 2015. [Online]. Available: http://semantic-web-journal.net/content/five-star-rating-scheme-assess-application-seamlessness. [Accessed: 01-Oct-2017].

[81] I. Ermilov, M. Martin, J. Lehmann, and S. Auer, “Linked Open Data Statistics: Collection and Exploitation,” in *International Conference on Knowledge Engineering and the Semantic Web*, 2013, pp. 242–249.

[82] C. Stadler, M. Martin, and S. Auer, “Exploring the web of spatial data with facete,” in *Proceedings of the 23rd International Conference on World Wide Web - WWW ’14 Companion*, 2014, pp. 175–178.

[83] N. Bikakis, J. Liagouris, M. Kromida, G. Papastefanatos, and T. Sellis, “Towards Scalable Visual Exploration of Very Large RDF Graphs,” in *International Semantic Web Conference*, 2015, pp. 9–13.

[84] M. Weise, S. Lohmann, and F. Haag, “LD-VOWL: Extracting and Visualizing Schema Information for Linked Data,” in *Visualization and Interaction for Ontologies and Linked Data (VOILA! 2016)*, 2016, pp. 120–127.

[85] M. C. Pattuelli, M. Miller, L. Lange, S. Fitzell, and C. Li-Madeo, “Crafting Linked Open Data for Cultural Heritage: Mapping and Curation Tools for the Linked Jazz Project,” *Code4Lib Journal*, no. 21, p. 4, 2013.

[86] D. V. Camarda, S. Mazzini, and A. Antonuccio, “LodLive, exploring the web of data,” in *Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS ’12*, 2012, pp. 197–200.

[87] P. Heim, S. Lohmann, and T. Stegemann, “Interactive Relationship Discovery via the Semantic Web,” in *The Semantic Web: Research and Applications*, 2010, pp. 303–317.

[88] R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio, “Semantic wonder cloud: Exploratory search in DBpedia,” in *Current Trends in Web Engineering. ICWE 2010*, 2010, pp. 138–149.

[89] N. Bikakis, G. Papastefanatos, M. Skourla, and T. Sellis, “A hierarchical aggregation framework for efficient multilevel visual exploration and analysis,” *Semantic Web*, vol. 8, no. 1, pp. 139–179, Nov. 2016.

[90] M. Alonen, T. Kauppinen, O. Suominen, and E. Hyvönen, “Exploring the Linked University Data with Visualization Tools,” in *The Semantic Web: ESWC 2013 Satellite Events*, 2013, pp. 204–208.

[91] J. Brooke and others, “SUS-A quick and dirty usability scale,” *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[92] D. Denicola, “Custom Elements,” *W3C*, 2016. [Online]. Available: https://www.w3.org/TR/2016/WD-custom-elements-20161013/. [Accessed: 01-Oct-2017].