Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
Ciclo XXXI

Settore Concorsuale: 01/B1
Settore Scientifico Disciplinare: INF/01

Engineering Background Knowledge for Social Robots

Presentata da: Luigi Asprino

Coordinatore Dottorato
Paolo Ciaccia

Supervisore
Paolo Ciancarini

Esame finale anno 2019

To Serena.

Abstract

Social robots are embodied agents that continuously perform knowledge-intensive tasks involving several kinds of information coming from different heterogeneous sources. Providing a framework for engineering robots' knowledge raises several problems like identifying sources of information and modeling solutions suitable for robots' activities, integrating knowledge coming from different sources, evolving this knowledge with information learned during robots' activities, grounding perceptions on robots' knowledge, assessing robots' knowledge with respect humans' one and so on. Semantic Web research has faced with most of these issues and could provide robots with the means for creating, organizing, querying and reasoning over knowledge. In fact, Semantic Web standards allow to easily integrate data generated by a variety of components, thus enabling robots to make decisions by taking into account knowledge about physical world, data coming from their operating environment, information about social norms, users' preferences and so on. Semantic Web technologies provide flexible solutions that allow to extend and evolve robots' knowledge over time. Linked (Open) Data paradigm (a result of research in the Semantic Web field) lets to easily reuse (i.e. integrate with robots' knowledge) existing external datasets so to bootstrap a robot's knowledge base with relevant information for its activities. Linked Data also provides a mechanism that allows robots to mutually share knowledge. Existing solutions for managing robots' knowledge only partially exploit the potential of Semantic Web technologies and Linked Data. This thesis introduces a component-based architecture relying on Semantic

Web standards for supporting knowledge-intensive tasks performed by social robots, and whose design has been guided by requirements coming from a real socially assistive robotic application. All the components contribute to and benefit from the knowledge base which is the cornerstone of the architecture. The knowledge base is structured by a set of interconnected and modularized ontologies which are meant to model information relevant for supporting robots in their daily activities. The knowledge base is originally populated with linguistic, ontological and factual knowledge retrieved from the Linked Open Data. The access to the knowledge base is guaranteed by Lizard, a tool that provides software components with an API for accessing facts stored in the knowledge base in a programmatic and object-oriented way. This thesis also introduces two methods for engineering knowledge needed by robots: *(i)* A novel method for automatically integrating knowledge coming from heterogeneous sources with a frame-driven approach. *(ii)* A novel empirical method for assessing foundational distinctions over Linked Open Data entities from a common sense perspective (e.g. deciding if an entity inherently represents a class or an instance from a common sense perspective). These methods realize two tasks of a more general procedure meant to automatically evolve robots' knowledge by automatically integrating information coming from heterogeneous sources, and to generate common sense knowledge by using Linked Open Data as empirical basis. Feasibility and benefits of this architecture have been assessed through a prototype deployed in a real socially assistive scenario, whose this thesis presents two applications and the results of a qualitative and quantitative evaluation.

Acknowledgements

Riconosco che questo questo lavoro è frutto di un percorso iniziato molti anni fa sui banchi di scuola e se oggi presento una tesi di dottorato lo devo a molte persone. Vorrei iniziare dai miei genitori perchè i loro sforzi mi hanno permesso di studiare in tutta tranquillità e perchè hanno sempre creduto in me e mi hanno incoraggiato in quello che facevo. Vorrei ringraziare le mie sorelle Rosa e Paola insieme con i loro mariti che mi hanno pazientemente sopportato durante gli anni di università. Sfrutto questa sezione per esprimere la mia riconoscenza per Fulvio, Iride, Sara ed Erminia che mi hanno accolto, ospitato, sfamato e aiutato come figlio/fratello. Vorrei inoltre dedicare questa tesi ai miei nipoti Maria Letizia, Anna Luce, Lorenzo, Vittoria e Lorena. Spero che possiate trovare anche voi un lavoro che vi appassioni e che vi renda soddisfatti giorno per giorno.

Vorrei esprimere la mia più profonda gratitudine per la dr.ssa Valentina Presutti che nel 2015, credendo in me e nelle mie capacità, mi ha accettato nel suo gruppo di ricerca e in seguito mi ha guidato pazientemente per tutta la durata del dottorato. Un ringraziamento speciale lo devo al mio supervisore, prof. Paolo Ciancarini, per essere stato sempre pronto ad aiutarmi nelle mie difficoltà, per i suoi consigli e suggerimenti nelle attività di ricerca. Inoltre, vorrei ringraziare in modo particolare il prof. Aldo Gangemi per essersi sempre speso per trasferirmi le sue conoscenze allargando i miei orizzonti di ricerca. Vorrei esprimere la mia riconoscenza verso i revisori di questa tesi, la prof.ssa Eva Blomqvist e il dr. Fabien Gandon, perchè grazie ai loro commenti appassionati ho potuto migliorare questa tesi. Vorrei ringraziare il prof.

Frank Van Harmelen e il dr. Wouter Beek perchè mi hanno calorosamente accolto e ospitato nel loro gruppo durante il mio periodo all'estero. Se sono qui è anche merito di alcuni professori che durante gli anni di università mi hanno trasmesso entusiasmo verso le materie scientifiche. Tra questi, vorrei ringraziare in modo particolare il prof. Giuseppe De Giacomo perchè la sua passione per l'intelligenza artificiale mi ha spinto verso il mondo accademico. Infine, la mia riconoscenza va a tutti i miei insegnanti e professori che si sono spesi per la mia formazione. Qui vorrei ricordarne alcuni a cui sarò sempre riconoscente: Maria Elena Coppa, prof. Vaccarone, prof.ssa Civale, prof.ssa Libertino e prof. Calabrese.

Durante questi anni ho collaborato con diverse persone e anche questo lavoro è frutto del contributo di molti. In modo particolare vorrei esprimere la mia riconoscenza per Alessandro Russo e Andrea Giovanni Nuzzolese con i quali ho condiviso e continuo a condividere molte fatiche e lunghi "pellegrinaggi". Un grazie speciale va alle persone che hanno collaborato al lavoro presentato in questa tesi Mehwish Alam, Valerio Basile, Giorgia Lodi, Stefano Nolfi, Annarita Orsini, Silvio Peroni, Massimiliano Raciti, Diego Reforgiato Recupero, e Chiara Veninata.

Ringrazio tutte le persone che durante questi anni hanno fatto parte del laboratorio per i loro spunti di riflessioni, consigli, chiacchierate e risate: Valentina Carriero, Mario Caruso, Piero Conca, Sergio Consoli, Valerio Di Carlo, Ludovica Marinucci, Misael Mongiovì, Martina Sangiovanni, e Carlo Stomeo. Vorrei ricordare e ringraziare alcuni "compagni di banco e di merenda" che mi hanno accompagnato in questa avventura come Ciriaco Cauteruccio, Ferdinando Cipollone, Carlo Gazzaneo, Simone Lo Tufo, Mattia Lo Tufo, Angelo Maiorano, e Davide Marino.

Vorrei ringraziare con tutto il mio cuore i ragazzi del gruppo gamma. Avete un posto speciale nel mio cuore e non posso fare a meno di voi nella mia vita. Questo lavoro è anche indirettamente merito vostro.

Infine, vorrei concludere questi ringraziamenti con mia moglie Serena. Difficile esprimere a parole quanto ti sono grato. In tutte le cose belle della mia vita c'è il tuo zampino.

Forse dimentico di ringraziare qualcuno e questo mi dispiace, ma qualcun'altro si merita un ringraziamento di persona.

Luigi

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Social robots [21, 37, 56, 64, 61, 77] are autonomous embodied agents that interact, collaborate, communicate with humans, by following the behavioral norms expected by people with whom robots are intended to interact. Several definitions have been proposed for the term "social robot", but all of them broadly agree that a social robot has the following characteristics: *(i) Physical embodiment*, i.e. a social robot has a physical body; *(ii) Sociality*, i.e. a social robot is able to interact with people by showing human-like features while following the social rules (defined through society) attached to its role; *(iii) Autonomy*, i.e. a social robot makes decisions by itself (the autonomy is sometimes limited in testing phase, like in the Wizard of Oz experimental setting [110, 178]). In recent years, the field of socially assistive robotics [72] has emerged given the great potential of social robots in supporting people with cognitive impairment or physical disability [27, 96, 104, 149]. As overviewed in [137], existing robotic technologies for care range from pet-like devices to advanced anthropomorphic mobile robotic assistants. While service robots often focus on providing *physical* support, a socially assistive robot aims to provide cognitive support through *social* interaction.

In order to effectively interact, communicate, collaborate with humans, robots should demonstrate some intelligence. Generally, the approaches for building such an intelligence can be classified into two categories, namely, symbolic and subsymbolic approaches [169, 195]. In symbolic approaches real-world entities, facts, rules and concepts are formalized by means of symbols. Symbols are stored in a knowledge base and are manipulated to make conclusions and take decisions. Conversely, subsymbolic approaches attempt to address problems without building an explicit representation of concepts and entities. An example of subsymbolic approaches are artificial neural networks, i.e. large networks of extremely simple numerical processors (i.e. neurons), massively interconnected and running in parallel. These networks consume and produce numerical vectors. The connections between neurons determine how input vectors is transformed to an output vector. In these systems, knowledge is not encoded with symbols but rather in the pattern of numerical strengths of the connections between neurons. These networks can "learn" to perform a given task by seeing a set of examples. An example is a pair of a possible input for the task at hand with (possibly) the corresponding desirable output. From these examples a neural network can learn the optimal weights to predict the desirable output from a given input. There are benefits and drawbacks of both kinds of approaches. While subsymbolic approaches achieve better performance in specific tasks, they need a large quantity of examples to learn optimal strengths. These examples could be hardly available for certain tasks. The other main issue with subsymbolic systems is the inability of explaining why a certain output is provided for a given input. This could be a non-negligible problem for many domain, such as medicine. In contrast, symbolic systems are able to explain their decisions and do not require examples. However, these systems need considerable effort for defining symbols and designing rules and methods to manipulate symbols for solving problems.

Integrating the two approaches have also become more common to benefit of both strategies (e.g. [92, 99, 144]). The framework proposed in this thesis benefits of both symbolic and subsymbolic techniques. Subsymbolic techniques are used in perceptual tasks (such as translation of spoken language into text), whereas symbolic techniques are used for controlling the robot at an higher level [93]. In particular, subsymbolic subsystems of the robot transform low-level perception in symbols so to enable the symbolic processing of the control system. In this way the framework benefits of the state-of-the-art performance on perceptual tasks of subsymbolic techniques without compromising the possibility of having a system that is deterministic and able to explain its behavior and decisions (important requirements for the case study of this thesis, cf. Section 1.3).

In order to show human-like feature the robot should be able to manipulate a human-like set of symbols, called background knowledge. This could be informally defined as the knowledge that a robot need in order to operate. This background knowledge includes (but it is not limited to): linguistic, encyclopedic and procedural knowledge as well as knowledge concerning the physical world and social norms. All of these kind of knowledge are involved in a potential human-robot interaction. For example, suppose that a person asks a robot to "cut a slice of bread". In order to fulfill this request, the robot should resort to: *(i) linguistic knowledge* in order to understand what the person says (e.g. to associate the word "cut" to the meaning "detaching with a sharp-edged instrument"); *(ii) encyclopedic knowledge* in order to figure out what are the entities involved in the request (e.g. what is a *a slice of bread*); *(iii) procedural knowledge* in order to realize the steps to undertake (e.g. take the bread knife, put the bread on a cutting board etc.); *(iv) physical world knowledge* in order to figure out where the entities involved in the request are usually located, and if, what the person asks, is feasible and is something that the robot can do (e.g. knives and bread can be usually found in a kitchen); *(v) social norms knowledge* in order to check if, what the person asks, is something socially acceptable (i.e. cutting a slice of bread is acceptable but stabbing someone is generally immoral). This simple example shows the complexity and the need of these types of knowledge. Once the robot receives a user request, it is expected to interpret it, namely, it needs to associate the user request with an internal representation of its meaning. This representation should use referents with a formal semantics for the robot. From these referents the robot can access the other kinds of knowledge. For example from a referent representing the meaning of the word "cut", i.e. "detaching with a sharp-edged instrument", the robot is be able to access to procedural knowledge to figure out how to cut something. This is possible if all these types of knowledge are connected, well-organized and available to the robot.

Providing such a framework for engineering robots' knowledge raises several problems like identifying sources and modeling information relevant for robots' activities, integrating knowledge coming from different sources, evolving this knowledge with information learned during robots' activities, grounding perceptions on robots' knowledge, assessing robots' knowledge with respect humans' one and so on. Several knowledge representation approaches for robots can be found in literature, we present the most relevant that are employed in robotic architectures. The choice of a knowledge representation formalism has an impact on the expressiveness (the breadth of concepts that can be represented), on the ability infer logical consequences form asserted facts in a tractable manner (i.e. tractability) and on the amount of tools and knowledge already available that can be provided to robots for supporting their tasks.

Predicate logic is a collection of formal systems that includes: *propositional logic* (dealing with zero-arity predicates, called propositions), *first-order logic* (FOL) (dealing with terms of higher arity) and higher order logic (in which predicates can be arguments of other predicates). Unlike propositional logic, first-order logic and higher order logic are undecidable (i.e. it does not exist a method for validating all the formulas). To overcome the undecidability of FOL, some fragments of the logic have been proposed (e.g. horn clause and description logic). Propositional logic, horn clause (which constitutes the foundation of logic programming languages), description logic [17] (at the basis of the W3C's OWL-DL standard) are used in several robotic frameworks (e.g. [141, 152, 186, 206]). *Modal logic* extends propositional and predicate logic to include the modality operator. A modal is an operator that qualifies statements (e.g. usually, possibly). An example of robotic framework based on modal logic is Tino [60]. *Temporal logic* allows to qualify proposition in terms of time (e.g. "I am *always* hungry"). It is used in some robotic frameworks such as [69, 114]. Another logics particularly relevant for robotic frameworks (e.g. [121])

is the *probabilistic logic* which provides a formalism able to handle uncertainty.

All of these logic frameworks allow to potentially encode robot's knowledge and are supported by several off-the-shelf reasoning tools. Although some of them show high expressive power and attractive computational features, they almost lack of existing resources that could provide robots with a comprehensive background knowledge for their tasks. The choice of a framework impacts on the effort required to provide a robot with a considerable amount of knowledge for its tasks. This problem is even more significant for social robots which need of a sizable and heterogeneous knowledge bases to operate.

The Semantic Web [28] standards give a good trade-off among the expressiveness, tractability and availability of resources and tools for manipulating such knowledge. The Semantic Web (cf. Section 2.2) is an extension of the World Wide Web aimed at providing a framework that allows data to be shared with a common syntax and semantics. It based on a language defining the syntax of data to be shared (i.e. XML), a model defining the format of data (i.e. RDF) and a language to formally specify the semantics of data (i.e. OWL). OWL is derived from description logics and comes in three forms: *(i)* OWL Full for maximum expressiveness, but undecidable; *(ii)* OWL DL designed to provide the maximum expressiveness while retaining decidability; *(iii)* OWL Lite that supports taxonomies and simpler constraints with a limited expressiveness but attractive computational features. Semantic Web standards provide a good expressive power (equivalent to Description Logics), without compromising decidability and tractability. It is supported by a pletora of off-the-shelf reasoners, knowledge management systems, as well as tools for creating, organizing and integrating knowledge. Another benefit of implementing a framework that relies on Semantic Web technologies is the opportunity of exploiting knowledge available as Linked Open Data (LOD). LOD is a huge web of data ($\sim$200 billion linked facts[1]) formally (and uniformly) represented in RDF and OWL, and openly available on the Web.

[1]LODCloud, https://lod-cloud.net/

1.1 Goals of the Thesis

The aim of this thesis can be summarized in the following question.

> *RQ0: To what extent Semantic Web technologies and Linked Data can be used to create, organize, access to, and evolve robot's background knowledge?*

Existing solutions for managing robots' knowledge (such as RoboEarth [208], RoboBrain [186], ORO [123], ORA [171], RACE [182], and OUR-K framework [127]) only partially exploit the potential of Semantic Web technologies and Linked Data. In these projects, Semantic Web technologies are mostly employed to address syntactic heterogeneity of data, to define conceptualizations of the robots' knowledge, and, more rarely, to include external datasets within the robots' knowledge base. Most of these frameworks focused on a single dimension of knowledge while disregarding more social aspects. In fact knowledge manipulated by these frameworks are often related to the interaction of the robot with the physical environment for supporting navigation [152], inter-robot communication [208], manipulation of the objects [206], representation of robot's actions [209].

We claim that robots' architectures can profoundly benefit of Semantic Web technologies and Linked Data paradigm. *(i)* Semantic Web technologies could enable an incremental and iterative development of the architecture. *(ii)* Semantic Web standards allow to easily integrate data generated by a variety of components, thus enabling robots to make decisions by taking into account knowledge about the physical world, data coming from their operating environment, information about social norms, users' preferences and so on. *(iii)* Semantic Web technologies provide flexible solutions to extend and evolve robots' knowledge over time. *(iv)* Linked (Open) Data paradigm lets to easily reuse (i.e. integrate with robots' knowledge) existing external datasets so to bootstrap knowledge base with relevant information for robots' activities. *(v)* Linked Data also provides a mechanism that allows robots to mutually share knowledge.

As anticipated with the main research question RQ0, the goal of this thesis is to investigate feasibility and benefits of engineering background knowledge of social robots with a framework based on Semantic Web technologies and Linked Data. This investigation has been carried out in a socially assistive context (presented in Section 1.3) by following the methodology described in Section 1.4. The research question RQ0 can be decomposed into the following sub-questions which will be investigated and discussed in the following chapters (one question for each Chapter, from Chapter 3 to Chapter 7).

RQ1: What kind of knowledge a robot needs to operate in socially assistive context? What exiting ontologies can be used to organize the robot's knowledge? What ontologies need to be advanced? What domains of interest in this context miss of a conceptualization?

RQ2: What Linked Data can provide background knowledge for social robots tasks?

RQ3: How to provide robots with access to knowledge?

RQ4: How to integrate robot's knowledge with data coming from robot's experience?

RQ5: How Semantic Web technologies can be orchestrated to support robot tasks?

1.2 Contributions of the Thesis

This thesis contributes to goals presented in Section 1.1 as follows.

- We have devised a set of interconnected and modularised ontologies, i.e. the MARIO Ontology Network (MON), which are meant to model all *knowledge areas* that are relevant for robots' activities in socially assistive contexts (cf. **RQ1**). This ontology network defines reference models for representing and structuring the knowledge processed by the robot. MON provides a robot with the means for creating, organizing, querying and reasoning over a background knowledge. MON reuses and integrates state-of-the-art ontologies in various domains (such those related to personal information, social and multimedia contents), and, proposes novel solutions in medical domain (e.g. CGA Ontology, cf. Section 3.3.2) and in robotic domain (i.e. Affordance Ontology, cf. Section 3.3.1). These modules enables robots to assess the medical, psycho-social and functional status of a person and to decide the most appropriate action to perform in a given situation.

- The knowledge base is originally populated with lexical, linguistic and factual knowledge retrieved from Linked Open Data. The thesis presents a novel process for generating, integrating and assessing this knowledge (cf. **RQ2**). Moreover, we propose a novel empirical method for assessing foundational distinctions over Linked Open Data entities from a common sense perspective (e.g. deciding if an entity inherently represents a class or an instance from a common sense perspective). This method realizes the first step of a more general procedure meant to automatically generate common sense knowledge from Linked Open Data (cf. **RQ2**). These methods advance state-of-the-art in Semantic Web by proposing standardized techniques for creating an integrated repository of linguistic, factual, encyclopedic, ontological and common sense knowledge. The benefits of these techniques as well as the resulting datasets are not limited to robotic domain, but also extend to every application domain that requires a rich knowledge base to operate [59].

- We have developed an object-RDF mapper (called *Lizard*) that facilitates software components to interact with an RDF knowledge base (cf. **RQ3**). In particular, given an RDF knowledge base and an OWL ontology describing its structure, Lizard provides applications with an API for accessing RDF facts stored in a knowledge base following the object-oriented paradigm. The API reflects the semantics of the input ontology and allows transparent access to the knowledge base, Differently from existing systems, Lizard exposes the API following the REST architectural style over HTTP. This tool is aimed at easing the software development of knowledge-aware systems by filling the gap between Semantic Web technologies and Object-Oriented applications. The benefits of Lizard are not to be intended only for robotic domain, but every application that need to access to a knowledge base compliant with Semantic Web standards could potentially use Lizard's API.

- We introduce a novel approach for automatically integrating knowledge coming from different ontologies with a frame-driven approach (cf. **RQ4**). This method aimed at finding complex correspondences between ontology entities according the intensional meaning of their models, hence abstracting from their logical types. In this proposal, frames are considered as "unit of meaning" [89] for ontologies and are used as a mean for representing intentional meaning of ontology entities. The frame-based representation of entities' meaning enables at finding complex correspondences among entities abstracting from their logical type thus leading a step ahead the state of the art of ontology matching. Other potential benefits of this method related to understanding and generating natural language will be discussed in Section 6.3.

- This thesis introduces a component-based architecture relying on Semantic Web standards for supporting knowledge-intensive tasks performed by social robots, and whose design has been guided by requirements coming from a real socially assistive robotic application. The ultimate goal aim of the architecture is to create a platform for easing the development of robotic applications by providing developers with off-the-shelf software artifacts, models and data. The strategy to pursue this goal is to massively reuse Semantic Web technologies due to their intrinsic availability and interoperability. Moreover, we present a prototype which is aimed at demonstrating feasibility and benefits of such a architecture and two applications running on top of the architecture prototype. We claim that the prototype is only an example of robotic systems that benefit of framework proposed in the thesis. This framework could potentially be integrated, with appropriate adaptions, with every autonomous agents (not limited to embodied systems).

1.3 Case Study: Companion Robots in Socially Assistive Context

A case study for this thesis work has been provided by the H2020 European Project MARIO[2]. This project has investigated the use of autonomous companion robots as cognitive stimulation tools for people with dementia. The MARIO robot and its capabilities were specifically designed to provide support to people with dementia, their caregivers and related healthcare professionals. Among its capabilities, MARIO can help caregivers in the patient assessment process by autonomously performing Comprehensive Geriatric Assessment (CGA) evaluations, and is able to deliver reminiscence therapy through personalized interactive sessions. These capabilities are part of a robotic software framework (inspired to the architecture presented in Chapter 7) for companion robots, and, they are supported by the knowledge representation and management framework proposed in this thesis. The overall framework and the applications presented in these thesis have been deployed on Kompaï-2 robots (showed in Figure 1.1), evaluated and validated during supervised trials in different dementia care environments, including a nursing home (Galway, Ireland), community groups (Stockport, UK) and a geriatric unit in hospital settings (San Giovanni Rotondo, Italy). There was a clear mutual benefit between this thesis work and the work carried out within the context of the MARIO Project. On the one hand, the MARIO benefited of the framework proposed in this thesis. On the other hand, the MARIO project provided a real-world application that fine-tuned the requirements and tested the capabilities of the contributions of this thesis.

1.4 Research Methodology

The structuring and organization of the research activities have been following two complementary paths, though interlinked and interleaved among each other. On the one hand, we approached a case study with an explorative strategy aimed at

[2]MARIO project, http://www.mario-project.eu/portal/

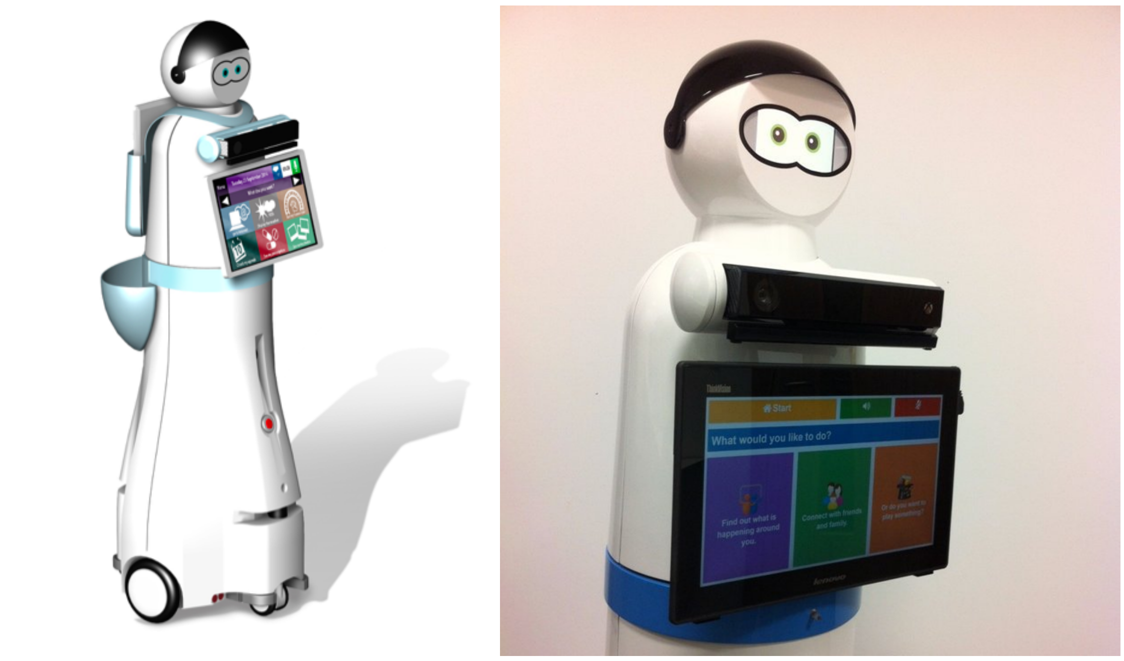

Start
What would you like to do?
Find out what is happening around you
Contact with friends and family

Figure 1.1: The Kompaï-2 robot and its user interface.

investigating the needs of a real socially assistive robotic application and highlighting the limits of current solutions. On the other hand, problems that came from the real setting have been generalized in order to contribute with their solutions to advance the state of the art. The two activity paths are summarized hereafter together with the strategy for evaluating the contributions of the thesis.

Explorative Approach to the Case Study. The work carried out along this path has been focusing on identifying the needs of a real socially assistive robotic application, highlighting the limits of current solutions, and, designing, developing, deploying and testing working solutions within a concrete robotic application. In line with the overall principles and methodology adopted in the project, we have been following an incremental and iterative design and development approach, inspired by Agile principles. As a consequence, the implemented approaches and solutions have been: *(i)* designed following a requirements-driven and user-centered approach, taking into account pilot sites' needs and scenarios; *(ii)* incrementally integrated, tested and validated during trial activities; *(iii)* gradually refined and improved on the basis of trials feedback.

Research Activities and Solutions Targeting Open Problems. The work carried out along this path has been focusing on research activities aimed at the identification of solutions targeting open problems in the broad field of knowledge representation and engineering. These research problems are either inspired by and abstracted from concrete use cases, or derive from general challenges that can be specialized in the context of socially assistive robots. These problems have been synthesized in the research questions RQ0-RQ5 outlined in Section 1.1. When such a problem is identified an analysis of the current solutions is performed, and, if limitations emerge from the state of the art, then, new hypotheses are defined and tested in the real scenario. In particular, the prototype developed within the context of the MARIO project is to be interpreted as a proof-of-concept implementing the framework proposed in this thesis and evaluated in a real assistive context.

Evaluation strategy. The contributions of the thesis have been evaluated by following two different strategies, one targeting the whole robotic system and the other focusing on individual components of the architecture. A prototype of the framework presented in this thesis has been developed within the context of the MARIO project (cf. Section 1.3) and assessed during supervised trials in different dementia care environments. The evaluation of the prototype consisted of a quantitative assessment, involving the use of standardized questionnaires, and a qualitative assessment, aimed at capturing impressions of the stakeholders. Architectural components were individually evaluated through suitable experiments (meant to assess the accuracy of components) or proof-of-concepts (intended to demonstrate the feasibility of components).

1.5 Thesis Outline

The remainder of this thesis is structured as follows:

Chapter 2 - Background. This chapter overviews the research areas related to this work, including a quick introduction to the Semantic Web, Socially (Assistive) Robotics and Common Sense Knowledge. Mentions of related work are also featured in other chapters.

Chapter 3 - An Ontology Network for Social Robots in Assistive Context. Chapter 3 presents a set of interconnected and modularised ontologies, called MARIO Ontology Network (MON), which are meant to model all *knowledge areas* that are relevant for robots' activities in socially assistive contexts. This ontology network defines reference models for representing and structuring the knowledge processed by the robot. MON provides a robot with the means for creating, organizing, querying and reasoning over a background knowledge.

Chapter 4 - Providing Linked Open Data as Background Knowledge for Social Robots. This Chapter investigates the possibility of populating the extensional level with data retrieved from the web. To this end two lines of research have been carried out in parallel focusing on linguistic and common sense knowledge respectively. Regarding the first line of research the chapter presents Framester, a huge linguistic knowledge graph integrating lexical, linguistic, ontological and encyclopedic data. This Chapter also introduces a novel empirical method for assessing foundational distinctions over Linked Open Data entities from a common sense perspective (e.g. deciding if an entity inherently represents a class or an instance from a common sense perspective). This method realizes the first step of a more general procedure meant to automatically generate common sense knowledge from Linked Open Data.

Chapter 5 - Accessing Background Knowledge using Lizard. Chapter 5 presents Lizard, an Object-RDF mapper providing software components with the access to the knowledge base following the Object-Oriented paradigm.

Chapter 6 - A Frame-based Approach for Integrating Ontologies. Chapter 6 describes the frame-based approach for integrating ontologies. This method enables to integrate knowledge from different structured (namely, ontologies and knowledge graphs) and unstructured sources (e.g. text).

Chapter 7 - A Knowledge Base Centered Software Architecture for Social Robots. This chapter presents a component-based architecture relying on semantic web technologies for supporting knowledge-intensive tasks performed by social robots. Moreover, this chapter presents a prototype which is aimed at demonstrating feasibility and benefits of such a architecture and two applications running on top of the architecture prototype.

Chapter 8 - Conclusion and Future Work. A summary of the overall research activity and possible lines of future work to be followed from the current state concludes this dissertation.

Chapter 2

Background

2.1 Social Robots

Social Robots are embodied agents designed to socially interact with people and can be categorized depending on the application domain or depending on the type of tasks are designed to perform (for comprehensive overviews please refer to [39, 77, 122, 181]). Leite et al. [122] identified four different application domains: health care, education, work environments and public spaces, and home. Socially Assistive Robots can be broadly classified into three categories (a review of the assistive social robots is provided by [39, 181]): *(i) Service Robots* are devices designed to support people living independently by assisting them with mobility, completing household tasks, and monitoring health and safety; *(ii) Companion Robots* [57] are meant to create companionship for human beings; *(iii) Coaching Robots* (e.g. [70, 71]) that act as a coach to encourage human beings through a series of therapeutic tasks for enhancing their health conditions.

2.1.1 Software Architectures for Social Robots

Despite the different application domains and the intended functions, most of the architectures of social robots [44, 70, 71, 94, 95, 101, 105, 135, 143, 211] are constituted by the following elements:

1. A subsystem that manages the hardware devices allowing the robot to perceive the environment (such as lasers used for navigation, cameras, touch sensors, microphones etc.).
2. A set of components dealing with the robot's motors and actuators (e.g. wheel engines, speakers).
3. A knowledge base storing information for supporting the robot's behaviors, tracing the users' activities or preferences, and collecting from the operating environment (e.g. maps).
4. A multi-modal user interface that provides users with multiple modes to interact with robots. This is typically delivered by a voice-user interface and/or a touch-screen device.
5. A behavior controller that gathers information from perceptual components, knowledge base and user interfaces, and decides the next actions to perform.
6. If necessary, a supervision interface that enables to remotely control the robot and to possibly interrupt the robot's operation. Using such a interface is part of one of the most common Human-Robot Interaction experimental techniques called Wizard of Oz [110, 178].

The architecture presented in Chapter 7 follows the structure of existing architectures and defines a subsystem that allows to dynamically (i.e. at run-time without need of re-deploy) extend the capabilities of the robot, thus enabling an agile and evolutionary development of the architecture. Examples of such flexibility mechanisms can be also found in literature. Fritsch et al. [79] proposed a flexible infrastructure to extend the capabilities of the companion enabling the interaction with humans. XML is used In this proposal as language for defining the format of messages exchanged by the components and to define the sequence of operations the robot have to perform. Similarly, the extensible architecture introduced by Rossi et al. [183] allows to modify and expand the multi-modal interface without impacting the rest of the architecture. These mechanisms partially fulfill the flexibility

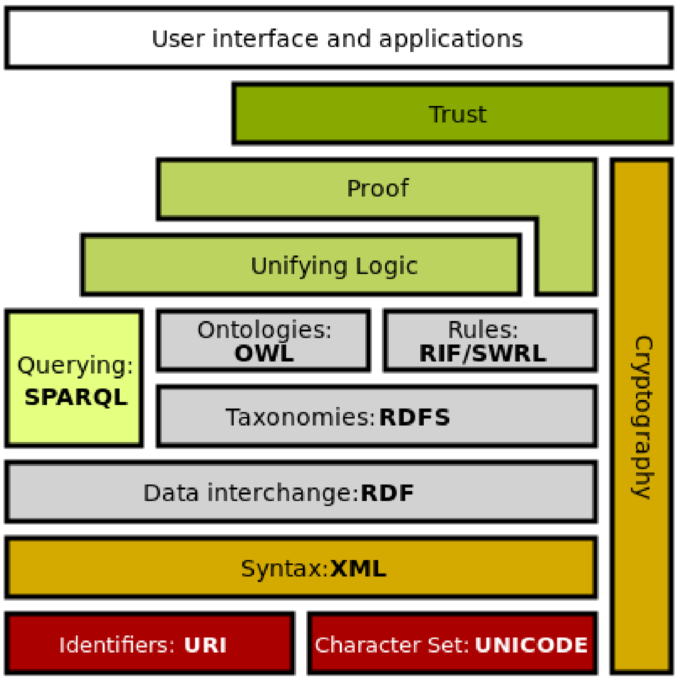


Figure 2.1: The Semantic Web stack.

and extensibility requirement since does not allow to dynamically deploy new software components. This feature is provided by our architecture which guarantees extensibility of the robot behaviors and capabilities. Other novel elements of our architecture concern with the use of semantic web and linked data for managing and bootstrapping the knowledge base.

2.2 The Semantic Web

The Semantic Web [28] is an extension of the Web aims at providing a common framework that allows data to be shared and reused across application boundaries. Standardisation for Semantic Web is under the care of World Wide Web Consortium (W3C). The W3C standards for the Semantic Web mainly include: XML, RDF(S), OWL and SPARQL. Figure 2.1 shows the semantic web stack and provides an overview of the standard technologies recommended by the W3C.

2.2.1 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a both human-readable and machine-readable format. An XML document consists of a properly nested set of open and close tags, where each tag can have a number of attribute-value pairs. Crucial to XML is that the vocabulary of the tags and their allowed combinations is not fixed, but can be defined per application of XML. In the Semantic Web context, XML is being used as a uniform data-exchange format thus providing a common syntax for exchange data across the web.

2.2.2 Resource Description Framework (RDF)

Resource Description Framework (RDF)[1] is a W3C recommendation originally designed as metadata model, it has being used as a general framework for modelling information. The basic construction in RDF is the triple <subject, preficate, object>. The subject denotes a resource and the predicate expresses a relationship between the subject and the object (which can be a value or another resource). For example, a way for representing the fact "The author of *War and Peace* is *Leo Tolstoy*" is

`:War_and_Peace :author :Leo_Tolstoy`

where `:War_and_Peace` and `:Leo_Tolstoy` are the Uniform Resource Identifiers (URIs) of two resources representing respectively the book titled "War and Peace" and the writer "Leo Tolstoy", and `:author` is the URI of the predicate "author" which is used to connect a book to its author. It is easy to see that an RDF model can be seen as a graph where nodes are values or resources and edges are properties. Several common serialisation formats of RDF are in use, including: TURTLE[2], RDF/XML[3], N-Triples[4].

[1]RDF, W3C Recommendation https://www.w3.org/TR/rdf11-concepts/
[2]TURTLE, https://www.w3.org/TR/turtle/
[3]RDF/XML, https://www.w3.org/TR/rdf-syntax-grammar/
[4]N-Triples, https://www.w3.org/TR/n-triples/

RDF Schema (RDFS)[5] provides a data-modelling vocabulary for RDF data. RDFS is an extension of RDF aims at providing basic elements for structuring RDF resources. It allows to define: Classes, Properties, Datatypes and Hierarchies for both classes and properties.

2.2.3 Web Ontology Language (OWL)

The Web Ontology Language (OWL)[6] is a semantic markup language for defining, publishing and sharing ontologies on the World Wide Web. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called *ontology*. OWL is part of the Semantic Web stack (see Figure 2.1) and it is complementary to XML, RDF and RDFS:

- *XML* provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents;
- *RDF* is a datamodel for resources and relations between them. It provides a simple semantics for this datamodel;
- *RDFS* is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalisation-hierarchies of such properties and classes;
- *OWL* adds constructs for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

[5]RDFs, W3C Recommendation https://www.w3.org/TR/rdf-schema/
[6]OWL, W3C Recommendation https://www.w3.org/TR/owl-ref/

2.2.4 SPARQL

SPARQL[7] is a query language for retrieving and manipulating data store in RDF format. Most forms of SPARQL queries contain a set of triple patterns called "basic graph pattern". Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable (denoted by a question mark). A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph can be substituted with the variables of the pattern. For example, the following SPARQL query retrieves pairs *book*s authored by Tolstoy.

```
SELECT ?book WHERE {?book :author :Leo_Tolstoy}
```

2.3 Ontologies

Historically ontology, listed as part of metaphysics, is the philosophical study of the nature of being, becoming, existence, or reality, as well as the basic categories of being and their relations. Ontology deals with questions concerning what entities exist or can be said to exist, and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences. While the term ontology has been rather confined to the philosophical sphere in the recent past, it has gained a specific role in a variety of fields of Computer Science, such as Artificial Intelligence, Computational Linguistics, and Database Theory and Semantic Web. In Computer Science the term loses part of its metaphysical background and, still keeping a general expectation that the features of the model in an ontology should closely resemble the real world, it is referred as a formal model consisting of a set of types, properties, and relationship types aimed at modeling objects in a certain domain or in the world. In early '90s Gruber [97] gave an initial and widely accepted definition:

> *An ontology is a formal, explicit specification of a shared conceptualization. An ontology is a description (like a formal specification of a*

[7]SPARQL, W3C Recommendation https://www.w3.org/TR/rdf-sparql-query/

program) of the concepts and relationships that can formally exist for an agent or a community of agents.

Accordingly, ontologies are used to encode a description of some world (actual, possible, counterfactual, impossible, desired, etc.), for some specific purpose.

In the Semantic Web, ontologies have been used as a formalism to define the logical backbone of the Web itself. The language used for designing ontologies in the Web of Data is the Web Ontology Language (OWL). In the last decade there has been a lot of research for investigating best practices for ontology design and re-use in the Web of Data. Among the others the EU-FP7 NeOn project[8] has provided sound principles and guidelines for designing complex knowledge networks called *ontology networks*. An ontology network is a set of interconnected ontologies. According to [4], the interconnections can be defined in a variety of ways, such as alignments, modularization based on `owl:import` axioms, and versioning. Ontology networks enable modular ontology design in which each module conceptualizes a specific domain and can be designed by using Ontology Design Patterns [88] and pattern-based ontology design methodologies, such as eXtreme Desing [33].

2.3.1 Knowledge Management Frameworks for Social Robots

Ontologies and Semantic Web technologies can support the development of robotic systems and applications that deal with knowledge representation, acquisition and reasoning. Furthermore, Semantic Web standards enable the interlinking of local robotic knowledge with available information and resources coming from the Web of Data. This trend has also led to the creation of the IEEE RAS Ontologies for Robotics and Automation Working Group (ORA WG), with the goal of developing a core ontology and an associated methodology for knowledge representation and reasoning in robotics and automation [171].

[8]http://www.neon-project.org/

In this direction, different frameworks have been proposed to model, manage and make available heterogeneous knowledge for robotic systems and applications. Focusing on service robots that operate in indoor environments through perception, planning and action, the ontology-based unified robot knowledge framework (OUR-K) [127] aims at supporting robot intelligence and inference methods by integrating low-level perceptual and behavioural data with high-level knowledge concerning objects, semantic maps, tasks, and contexts. An ontology-based approach is also adopted in the ORO knowledge management platform [123]. The platform stores and processes knowledge represented according to the OpenRobots Common Sense Ontology[9], an OWL ontology based on the OpenCyc upper ontology and extended with the definition of reference concepts for human-robot interaction. When deployed on a robot, the knowledge base can be instantiated with a priori common-sense knowledge and is then used as a "semantic blackboard" where the robotic modules (such as the perception module, the language processing module, the task planner and the execution controller) can store the knowledge they produce and query it back.

Along the same path, research projects and initiatives, such as KnowRob[10], RoboEarth[11] and RoboBrain[12], go beyond local knowledge bases and, also with the emergence of cloud-based robotics, propose Web-scale approaches. KnowRob [205, 206, 207] is a knowledge processing system and semantic framework for integrating information from different sources, including encyclopedic knowledge, common-sense knowledge, robot capabilities, task descriptions, environment models, and object descriptions. Knowledge is represented and formally modeled according to a reference upper ontology, defined using the Web Ontology Language (OWL). The system supports different reasoning capabilities and provides interfaces for accessing and querying the KnowRob ontology and knowledge base. Similarly, the RoboEarth framework [208] provides a web-based knowledge base for robots to access and share semantic representations of actions, object models and environments, augmented

[9]https://www.openrobots.org/wiki/oro-ontology
[10]http://knowrob.org/
[11]http://roboearth.ethz.ch/
[12]http://robobrain.me/

with rule-based learning and reasoning capabilities. The RoboEarth knowledge base relies on a reference ontology, as an extension of the KnowRob ontology to *(i)* represent actions and relate them in a temporal hierarchy; *(ii)* describe object models to support recognition and articulation; and *(iii)* represent map-based environments.

An HTTP-based API enables robots to access the knowledge base for uploading, searching and downloading information from and to their local knowledge bases. Along the same path, the RoboBrain knowledge engine [186] aims at learning and sharing knowledge gathered from different sources and existing knowledge bases, including linguistic resources, such as WordNet, image databases, such as ImageNet, and Wikipedia. Although the RoboBrain knowledge base does not explicitly adopt ontologies and Semantic Web technologies, knowledge is represented in a graph structure and stored in a graph database. A REST API enables robots to access RoboBrain as-a-service, to provide and retrieve knowledge on the basis of a specific query language.

The need to provide robots with a knowledge representation and management framework able to handle knowledge from different sources (including external data sources and knowledge bases) and support multiple tasks and applications has long been considered in robotics. However, it is only in recent years that the potential of ontology-based knowledge representation approaches and Semantic Web technologies has been considered to address the two aforementioned points in robotic platforms.

2.4 Pattern-based Ontology Design

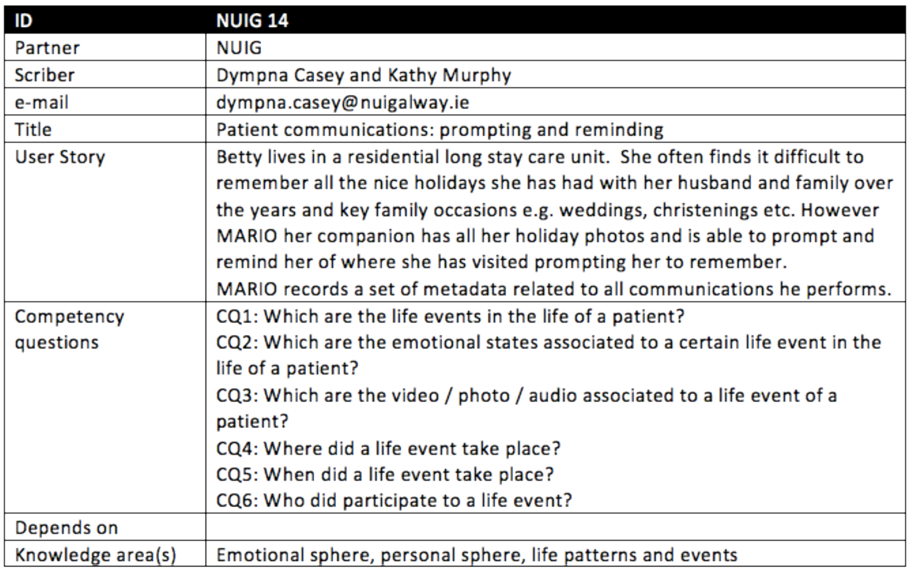
The notion of "pattern" has proved useful in design, as exemplified in diverse areas, such as software engineering. Under the assumption that there exist classes of problems that can be solved by applying common solutions (as has been experienced in software engineering), it is suggested to support reusability on the design side specifically. To this end Ontology Design Patterns (ODPs) have been proposed as modeling solutions to recurrent ontology design problems. ODPs are modeling components that can be used as basic building blocks of an *ontology network*. eXtreme Design (XD) is an ontology design methodology that supports the pattern-based approach. We adopted XD as methodology for designing the MARIO Ontology Network presented in Chapter 3 and we extensively reused ODPs. Sections 2.4.1 and 2.4.2 briefly introduce ODPs and XD, respectively.

2.4.1 Ontology Design Patterns

Ontology Design Patterns (ODPs) [88] is an emerging technology that favors the reuse of encoded experiences and good practices. ODPs are modeling solutions to solve recurrent ontology design problems. They can be of different types including: (i) *logical*, which typically provide solutions for solving problems of expressivity e.g., expressing n-ary relations in OWL; (ii) *architectural*, which describe the overall shape of the ontology (either internal or external) that is convenient with respect to a specific ontology-based task or application e.g. a certain DL family; (iii) *content*, which are small ontologies that address a specific modeling issue, and can be directly reused by importing them in the ontology under development e.g., representing roles that people can play during certain time periods; (iv) *presentation*, which provide good practices for e.g. naming conventions.

2.4.2 eXtreme Design

eXtreme Design (XD) [173, 33, 174] is a family of methods and associated tools, based on the application, exploitation, and definition of ontology design patterns (ODPs) for solving ontology development issues. XD principles are inspired by those of the agile software methodology called eXtreme Programming (XP). The main idea of agile software development is to be able to incorporate changes easily, in any stage of the development. Instead of using a waterfall-like method, where you first do all the analysis, then the design, the implementation and finally the testing, the idea is to cut this process into small pieces, each containing all those elements but only for a very small subset of the problem. XD is test-driven, and



ID	NUIG 14
Partner	NUIG
Scriber	Dympna Casey and Kathy Murphy
e-mail	dympna.casey@nuigalway.ie
Title	Patient communications: prompting and reminding
User Story	Betty lives in a residential long stay care unit. She often finds it difficult to remember all the nice holidays she has had with her husband and family over the years and key family occasions e.g. weddings, christenings etc. However MARIO her companion has all her holiday photos and is able to prompt and remind her of where she has visited prompting her to remember. MARIO records a set of metadata related to all communications he performs.
Competency questions	CQ1: Which are the life events in the life of a patient? CQ2: Which are the emotional states associated to a certain life event in the life of a patient? CQ3: Which are the video / photo / audio associated to a life event of a patient? CQ4: Where did a life event take place? CQ5: When did a life event take place? CQ6: Who did participate to a life event?
Depends on	
Knowledge area(s)	Emotional sphere, personal sphere, life patterns and events

Figure 2.2: An example of collected uses-story.

applies the divide-and-conquer approach as well as XP does. Also, XD adopts pair design, as opposed to pair programming. The main principles of the XD method can be summarised as follows:

- **Customer involvement and feedback.** The customer should be involved in the ontology development and its representative should be aware of all parts of the ontology project under development. Interaction with the customer representative is key for favoring the explicit expression of the domain knowledge.

- **Customer stories and Competency Questions.** The ontology requirements and its tasks are described in terms of small stories by the customer representative. Designers work on those small stories and, together with the customer, transform them in the form of Competency Questions [98] (CQs). CQs will be used through the whole development, and their definition is a key phase as the designers have the challenge to help the customer in making explicit as much implicit knowledge as possible. We asked all the partners involved in the case study to provide their own stories. The template for providing the stories is shown in Figure 2.2. The fields "*Partner*", "*Scriber*", "*e-mail*" were used for asking further clarification about the story. The *Title* field helped for a better understanding the main focus of the story. The "*Priority*" field was used to choose the stories to treat first. The allowed values were *High*, *Medium* and *Low*. "*Depends on*" allowed to specify a link between two stories. For example, if a story was too long, it could be split into two stories and this field allowed one to express the dependency. The last field "*Knowledge area(s)*" was used for associating the story with one or more knowledge areas which the story belonged to. The customer stories collected together with the resulting Competency Questions can be retrieved on-line[13]. Other competency questions have been extracted by analysing domain documents, such as those used for effectuating a Comprehensive Geriatric Assessment (CGA) of a patient.

- **Content Pattern (CP) reuse and modular design.** A development project is characterised by two main sets: (i) the *problem space* composed of the actual modelling issues that have to be addressed during the project which are called "Local Use Case" (LUC); (ii) the *solution space* made up of reusable modelling solutions, called "*Global Use Case*" (GUC), representing the problem that a certain ODP provides a solution for. If there is a CP's GUC that matches a LUC it has to be reused, otherwise a new module is created. An analysis of the possible strategies for reusing CP is provided by [174].

- **Collaboration and Integration.** Collaboration and constant sharing of knowledge is needed in a XD setting, in fact similar or even the same CQs and sentences can be defined for different stories. When this happens, it means that these stories can be modelled by reusing a set of shared CPs.

- **Task-oriented design.** The focus of the design is on that part of the domain of knowledge under investigation that is needed in order to address the user stories, and more generally, the tasks that the ontology is expected to address.

[13]http://etna.istc.cnr.it/mario/D5.1/.

- **Test-driven design.** A new story can be treated only when all unit tests associated with it have been passed. An ontology module developed for addressing a certain user story associated to a certain competency question, is tested e.g. (i) by encoding in the ontology a sample set of facts based on the user story, (ii) defining one or a set of SPARQL queries that formally encode the competency question, (iii) associating each SPARQL query with the expected result, and (i) running the SPARQL queries against the ontology and compare actual with expected results.

2.5 Ontology Matching

Among the various semantic technology proposed to handle heterogeneity Ontology Matching [191] has proved to be an effective solution to automate integration of distributed information sources. Ontology Matching (OM) finds correspondences between semantically related entities of ontologies. These correspondences enable several tasks such as ontology merging, query answering, or data translation. There have been proposed several formalization of the matching problem, we follow the formalization in [67] that provide a unified approach over the previous works. The matching problem is the problem of finding an alignment between two ontologies. An alignment is a set of 4-uple (e_1, e_2, r, n) where: (i) e_1 and e_2 are entities defined by the first and the second ontology, respectively; (ii) r is a relation holding between e_1 and e_2, e.g., equivalence, subsumption, disjointness; (iii) n is the confidence measuring the likelihood that the relation holds.

2.6 Linguistic Linked Open Data Resources

Many resources belonging to different domains are now being published on-line using Linked Data principles to provide easy access to structured data on web. This includes many linguistic resources that are already a part of Linked Data, but made available mainly for the purpose of being used by NLP applications.

Two of the most important linguistic linked open data resources are WordNet [145] and FrameNet [19]. They have already been formalised as semantic web resources, e.g. in OntoWordNet [85], WordNet RDF [16], FrameNet RDF [153], etc. FrameNet allows to represent textual resources in terms of Frame Semantics. The usefulness of FrameNet is limited by its limited coverage, and non-standard semantics. An evident solution would be to establish valid links between FrameNet and other lexical resources such as WordNet , VerbNet and BabelNet to create wide-coverage and multi-lingual extensions of FrameNet. By overcoming these limitations NLP-based applications such as question answering, machine reading and understanding, etc. would eventually be improved. Within MARIO these were important requirements, hence we developed Framester (presented in Chapter 3): a frame-based ontological resource acting as a hub between e.g. FrameNet, WordNet, VerbNet, BabelNet, DBpedia, Yago, DOLCE-Zero, and leveraging this wealth of links to create an interoperable *predicate space* formalised according to frame semantics [75], and semiotics [80]. Data designed according to the predicates in the predicate space created by Framester result to be more accessible and interoperable, modulo alignments between specific entities or facts.

The closest resources to Framester are FrameBase [184] and Predicate Matrix [116]. FrameBase aimed at aligning linked data to FrameNet frames, based on similar assumptions as Framester's: full-fledged formal semantics for frames, detour-based extension for frame coverage, and rule-based lenses over linked data. However, the coverage of FrameBase is limited to an automatically learnt extension (with resulting inaccuracies) of FrameNet-WordNet mappings, and the alignment to linked data schemas is performed manually. Anyway, Framester could be combined with FrameBase (de)reification rules so that the two projects can mutually benefit from their results.

Predicate Matrix is an alignment between predicates existing in FrameNet, VerbNet, WordNet, and PropBank. It does not assume a formal semantics, and its coverage is limited to a subset of lexical senses from those resources. A RDF version of Predicate Matrix has been created in order to add it to the Framester linked data cloud, and (ongoing work) to check if those equivalences can be reused in semantic web applications.

2.7 Common Sense Knowledge

Over the years, a number of projects aimed at generating common sense knowledge. Regardless specific settings, the results of these projects can be seen an ontology $O =< T, A >$ consisting of T, a T-box (i.e. terminology box, also called schema or vocabulary) and A is an A-box (assertion box). The outcome of these projects can be informally classified on the basis of following criteria. *(i)* the process used to generate knowledge (e.g. automatic or with humans in the loop); *(ii)* the breadth of the knowledge produced (i.e. universal or domain specific); *(iii)* the formalism used to represent knowledge; *(iv)* the density of the knowledge (i.e. the number of assertion per entity); *(v)* the richness (the variety of types and relations) and the depth (the number of inheritance relations) of the t-box; *(vi)* the level of interoperability with other datasets (i.e. the linkage of the ontology at both intensional and extensional level); *(vii)* the metadata (i.e. provenance and validity of stated facts).

Among existing projects we overview strengths, weaknesses and results of DBpedia, ConceptNet, NELL. DBpedia[14] [31] is a very popular dataset automatically obtained from Wikipedia infoboxes. DBpedia is the de-facto main hub of the Web of Data containing 4,58 milion entities of encyclopedic nature. However, some weaknesses of DBpedia are the scarcity of relations among entities and the limited depth of the vocabulary. In fact, the DBPedia ontology is induced from Wikipedia and is only partially aligned with existing formal theories (e.g. foundational ontologies). These drawbacks make hard answer queries such that "what knives are used for?", "give me all physical entities".

ConceptNet[15] [130, 197, 196] is a large scale multilingual semantic network that integrates knowledge from *(i)* Open Mind Common Sense project [193, 194] who ran

[14]DBpedia, http://wiki.dbpedia.org/

[15]ConceptNet, http://conceptnet.io/

a web site that collected common sense facts from users; *(ii)* existing datasets such as DBpedia, Wiktionary, Open Multilingual WordNet, OpenCyc [125] and Umbel; *(iii)* Verbosity, a game with a purpose that learns common sense knowledge from people's intuitive word associations. ConceptNet defines thirty lexical and common sense relations among its entities such as: "antonym", "synonym", "is used for" (that associates an object with what is used for, e.g. "bridge" and "cross water"), "at location" (that associates an object with its typical locations, "butter" and "refrigerator"), "capable of" (associating an object with what it can do, e.g. "knife" and "cut") etc. NELL[16] [45, 148] is an ongoing project aiming at learning large semantic network (similar to ConceptNet) with a never ending approach. The main drawback of ConceptNet and NELL is the inherent ambiguity of their concepts (e.g. "apple" is both a fruit and a computer company that can be used both for "eating" and for "computing", and it is controlled by "Steve Jobs"). ConceptNet and NELL provide information about provenance and confidence of facts, but they miss contextual conditions that make the facts true. Moreover, it is not clear how these projects select the resources to extract the knowledge from.

[16]NELL, http://rtw.ml.cmu.edu/rtw/

Chapter 3

An Ontology Network for Social Robots in Assistive Context

In order to interact with people showing human-like features, a social robot must be provided with a human-like background knowledge. Furthermore, when employed in socially assistive context, robots continuously perform knowledge-intensive tasks aimed at *(i)* assisting their users with their daily activities (e.g. drive the patients to a specific location or identifying searched objects); *(ii)* helping nurses, physician and familiars in the healthcare process of people with dementia (e.g. collecting information for assessing patient's cognitive status). Determining what kind of knowledge social robots need in socially assistive context and how to organize their knowledge is the goal related to research question RQ1 (cf. Section 1.1) which is investigated in this chapter. To this end, the chapter presents a set of interconnected and modularised ontologies, i.e. the MARIO Ontology Network (MON), which are meant to model all *knowledge areas* that are relevant for robots' activities in socially assistive contexts. This ontology network defines reference models for representing and structuring the knowledge processed by the robot. MON provides a robot with the means for creating, organizing, querying and reasoning over a background knowledge. The robot background knowledge consists of: lexical knowledge (e.g. natural language lexica and linguistic frames), domain knowledge (e.g. users related information), environmental knowledge (e.g. physical locations and maps), sensor knowledge (e.g. RFID, life measures), and metadata knowledge (e.g. entity tagging). The Ontology

Network, named MARIO Ontology Network (MON), is composed of several modularised ontologies that cover different knowledge areas that are relevant to the tasks of supporting people affected by dementia. The knowledge areas and the ontology modules were identified by analysing the use cases that emerged from the MARIO project (cf. Section 1.3), These uses cases mainly describe actions and behaviors featuring the MARIO robot. Nevertheless, they also provide a detailed descriptions about the nature of the knowledge that the robot has to deal with in order to behave. The MON consists of 53 modules covering 12 knowledge areas. The Ontology Network has been developed following the eXtreme Design methodology (introduced in Section 2.4.2) and by extensively reusing Ontology Design Patterns (cf. Section 2.4.1). The rest of the Section is organized as follows. Section 3.1 describes the ontology development process, Section 3.2 presents the MON's knowledge areas and Section 3.3 outlines the most innovative ontology modules of the Ontology Network.

3.1 Design Methodology

The MARIO Ontology Network (MON) has been designed by following best practices and pattern-based ontology engineering methods aimed at extensively re-using Ontology Design Patterns (ODPs) [88]. In particular, the MON has been developed following the eXtreme Design (XD) [33, 173] methodology (cf. Section 2.4.2). This methodology has been extended in order to identify the knowledge areas that are relevant to a companion robot (cf. 3.2) and to provide ontology engineers with guidelines for re-using existing ontologies (cf. 3.1.1). Section 3.1.2 illustrates how eXtreme Design has been configured in for the development of MON.

3.1.1 Guidelines for Ontology Re-use

Linked Data is rapidly increasing, especially in the public sector where opening data is becoming a consolidated institutional activity. However, the importance of providing Linked Data with a high quality ontology modeling is still far from being fully perceived. The result is that Linked Data are mostly modeled by directly reusing individual classes and properties defined in external ontologies, overlooking the possible risks caused by such a practice. Although ontology reuse is a recommended practice in most ontology design methodologies [192], a standardization of ontology reuse practices is still missing. Most literature on ontology reuse is focused on the challenging issue of ontology selection, while our perspective is on how to implement reuse once the selection finalized. This practice may compromise the level of semantic interoperability that can be achieved. Therefore, the need of clear guidelines for ontology reuse arise.

In [174] we provided a series of guidelines for ontology reuse in the context of ontology projects that exhibit these characteristics: *(i)* there is no ontology that addresses all or most of the requirements of the local ontology project; *(ii)* the ontology under development is meant to be used as a reference ontology for a certain domain; *(iii)* there is the willingness to comply with existing standards. These guidelines can be integrated into the tasks 7 and 8 of the XD workflow (cf. Figure 3.1 and Section 2.4.2).

Ontology re-use models can be classified based on *(i)* the type of reused ontology (e.g. foundational, top-level, ontology design patterns, domain ontologies); *(ii)* the type of reused ontology fragment (e.g. individual entities, modules, ontology design patterns, arbitrary fragments); *(iii)* the amount of reused axioms (e.g. import of all axioms, of only axioms in a given neighbourhood of an entity, of no axioms); *(iv)* the alignment policy (e.g. direct reuse of entities, reuse via equivalence or subsumption relations such as `owl:equivalentClass` and `rdfs:subClassOf`). The only characteristic that all these models share is to reuse entities with the same logical type as they were defined (e.g. an entity defined as `owl:Class` in an ontology is commonly reused as such).

We identify the following possible approaches to ontology reuse.

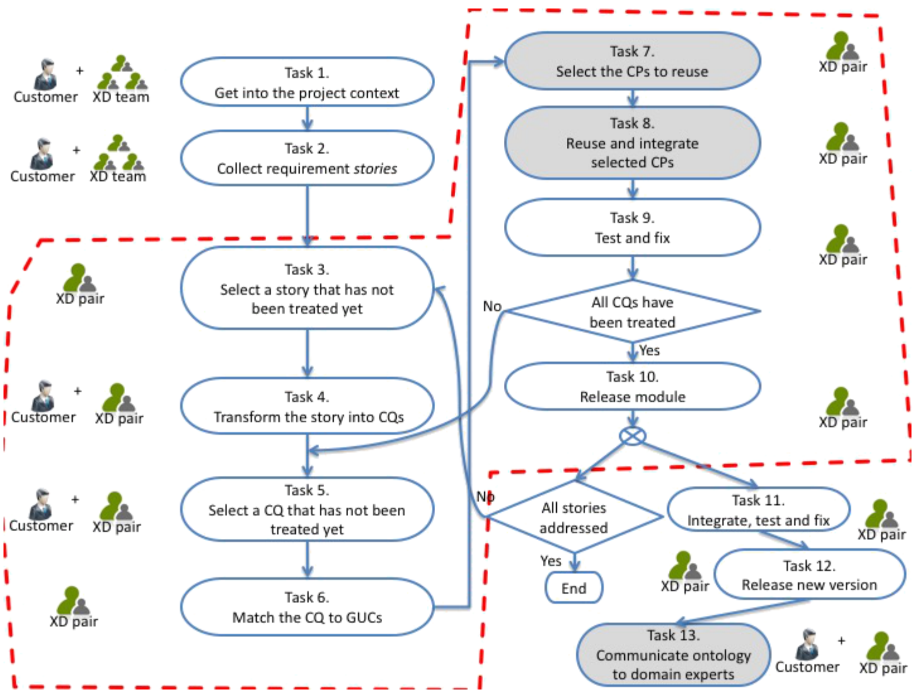

Customer + XD team
Task 1. Get into the project context
Customer + XD team
Task 2. Collect requirement stories
XD pair
Task 3. Select a story that has not been treated yet
Customer + XD pair
Task 4. Transform the story into CQs
Customer + XD pair
Task 5. Select a CQ that has not been treated yet
XD pair
Task 6. Match the CQ to GUCs
Task 7. Select the CPs to reuse
XD pair
Task 8. Reuse and integrate selected CPs
XD pair
Task 9. Test and fix
XD pair
No
All CQs have been treated
Yes
Task 10. Release module
XD pair
No
All stories addressed
Yes
End
Task 11. Integrate, test and fix
XD pair
XD pair
Task 12. Release new version
Task 13. Communicate ontology to domain experts
Customer + XD pair

Figure 3.1: The eXtreme Design workflow as extended in [174]. The highlighted tasks involve the guidelines for ontology reuse.

Direct Reuse of Individual Entities. This approach consists of directly introducing individual entities of external ontologies in local axioms. This practice is very common in the Linked Data community, however it is a routine, not a good practice, at all. It is essentially driven by the intuition of the semantics of concepts based on their names, instead of their axioms. In this case, the risk that the formal semantics of the reused entities is incompatible with the intended semantics to be represented is rather high. Moreover, with this practice a strong dependency of the local ontology with all the reused ontologies is created. This dependency may put at risk the sustainability and stability of the local ontology and its associated knowledge bases: if a change in the external ontology introduces incoherences in the local one, they must be dealt with a redesign process and consequential change in the ontology signature.

Indirect Reuse of Ontology Modules and Alignments. With this approach, the modeling of some concepts and relations, which are relevant for the domain but applicable to more general scopes, is delegated to external ontologies by means of ontology module reuse. An ontology module is a fragment that may be identified as providing a solution to one or more specific requirements of the local ontology. For example, let us consider an external ontology modeling the participation of an individual (e.g. through a property `ex:isInvolvedIn`) to an event (e.g. a class `ex:Event`). If the local ontology needs to specify a particular involvement in an event (e.g. `lo:hosted`) it should specialize (it indirectly reuses) the relation of the external one (i.e. `ex:isInvolvedIn`). The fragment of the external ontology identified as relevant for the local ontology may be communicated in some usage documentation provided with the ontology. Nevertheless, it is difficult to provide third parties with a formal indication of the fragment that was meant to be relevant. This may lead to high heterogeneity in the usage of external fragments in data modeled through the local ontology. As for ontology sustainability, when a change in the external ontology provokes possible incoherences, the redesign process would be easier dealt with as compared to the previous approach.

Direct Reuse of Ontology Design Patterns and Alignments. If the fragment is clearly and formally identified, since it is embedded in a dedicated ontology, some of the previous remarked issues can be mitigated. Let us consider that the earlier example class `ex:Event` is defined in an external ontology that implements a specific ODP. In this case, a scenario in which a redesign process must be undertaken may be less frequent. In fact, ODPs are developed for reuse purposes and thus they are unlikely to change. In the light of these observations, it is recommended to reuse ODPs in contrast to individual entities.

Indirect Reuse of Ontology Design Patterns and Alignments. ODPs are used as templates. This approach is an extension of the previous one. At the same time, the ontology guarantees interoperability by keeping the appropriate alignments with the external ODPs, and provides extensions that satisfy more specific requirements. The alignment axioms may be published separately from the core of the ontology. With this type of reuse, the potential impact of possible changes in the external ODP is minimised. In fact, even if incoherences show after a change in the external ODP (which is rather unlikely to happen) the redesign process would be very simple. The ontology signature and axioms would remain unchanged, as incoherences would be resolved by simply removing or revising the alignment axioms.

Reuse method	**Fragment**	**Advantages**	**Disadvantages**
Direct Re- use	Individual Entity	Linked data practise	Semantic ambiguity, difficulty in verifying the consistency among the diverse reused concepts, dependency on external ontolo- gies, instability and unsustainability
Indirect Re- use	Ontology Module	Stability and sustain- ability of domain re- lations and concepts, modularity, interoper- ability	Possible heterogeneity in module usage, de- pendency on external modules, instability and unsustainability limited to external modules

Continued on next page

Table 3.1: Advantages and Disadvantages of different approaches to ontology reuse.

Reuse method	**Fragment**	**Advantages**	**Disadvantages**
Direct Re-use	Ontology Design Pattern	Stability and sustainability of domain relations and concepts, modularity, interoperability, easier redesign in case of external changes	Dependency on external modules, mitigated risk of instability and unsustainability limited to external ODPs
Indirect Re-use	Ontology Design Pattern	Stability and sustainability of domain relations and concepts, modularity, interoperability, dependency on external modules limited to alignment axioms	Slightly increased design effort for moulding ODPs

Table 3.1: Advantages and Disadvantages of different approaches to ontology reuse.

Table 3.1 summarizes the advantages and disadvantages of the discussed four approaches. In general, among all of them, the recommended one is the fourth approach: in the situation of incoherence raised by a change in an external reused ontology, it guarantees the easiest maintenance. Besides the development of MON, we applied these guidelines in two Linked Open Data projects of the e-government sector. The first project was developed in the context of cultural heritage, in collaboration with the Italian Ministry of Cultural Heritage and Activities and Tourism [131]; the second was carried out within the agriculture domain, in collaboration with the Italian Ministry of Agriculture [165].

3.1.2 MARIO Ontology Network Development Process

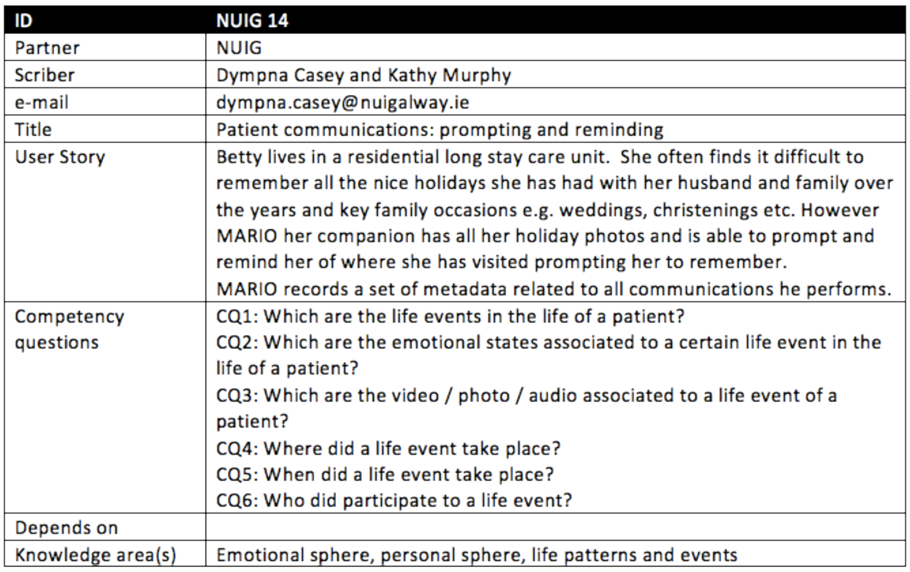
The first task of the ontology network development process was identifying the knowledge areas that the ontology has to cover. This task is ideally included in the first step of eXtreme Design process which aims at providing "the ontology design team with an overview of the problem from a domain expert perspective, its scope, and agree on initial terminology" (cf. [172]). The knowledge areas were identified by analyzing the use cases that emerged from the MARIO Project [29]. Section 3.2 describes the MON knowledge areas and their connection with MARIO's use cases.

As in the eXtreme Desing methodology, the ontology requirements were described in terms of small stories provided by the domain experts. The partners of the MARIO consortium acted as domain experts and provided the stories. The story template is shown in Figure 3.2. The fields "Partner", "Scriber", "e-mail" were used for asking possible further clarifications about the story. The "Title" field helped for a better understanding the main focus of the story. The "Priority" was used to choose the stories to treat first. The allowed priority values were "High", "Medium" and "Low". "Depends on" allowed to specify a link between two stories. For example, if a story was too long, it could be split into two stories and this field allowed one to express the dependency. The domain experts were recommended to keep the stories as short as possible and to use the dependency relation in case of long stories. The last field "Knowledge area(s)" was used for associating the story with one or more knowledge areas which the story belonged to.

Ontology Designers worked on those small stories and, together with the customer, transform them in the form of Competency Questions (CQs) [98]. Other competency questions have been extracted by analysing domain documents, such as those used for performing a Comprehensive Geriatric Assessment (CGA) of a patient. The customer stories collected together with the resulting Competency Questions can be retrieved on-line[1].

Following the eXtreme Design principles, the ontology requirements were implemented in a networked ontology consisting of 53 ontology modules covering

[1]MON customer stories, http://etna.istc.cnr.it/mario/D5.1/stories/.



ID	NUIG 14
Partner	NUIG
Scriber	Dympna Casey and Kathy Murphy
e-mail	dympna.casey@nuigalway.ie
Title	Patient communications: prompting and reminding
User Story	Betty lives in a residential long stay care unit. She often finds it difficult to remember all the nice holidays she has had with her husband and family over the years and key family occasions e.g. weddings, christenings etc. However MARIO her companion has all her holiday photos and is able to prompt and remind her of where she has visited prompting her to remember. MARIO records a set of metadata related to all communications he performs.
Competency questions	CQ1: Which are the life events in the life of a patient? CQ2: Which are the emotional states associated to a certain life event in the life of a patient? CQ3: Which are the video / photo / audio associated to a life event of a patient? CQ4: Where did a life event take place? CQ5: When did a life event take place? CQ6: Who did participate to a life event?
Depends on	
Knowledge area(s)	Emotional sphere, personal sphere, life patterns and events

Figure 3.2: The template of the user stories provided by the customer representatives.

12 knowledge areas. Knowledge areas and ontology modules were formalized in OWL ontology and are available on-line at [2]. Each ontology module has been developed in a iterative and incremental way. The use of these ontologies within the MARIO's abilities made emerge new requirements that were continuously addressed by the MARIO Ontology Network. This practice reflects the "Embracing Changes" paradigm which is one of the main principles of the Agile development on which eXtreme Design is based on.

Figure 3.3 provides an overview of the whole ontology network (detailed figures on portions of the network are provided in the next section). The nodes of the network are ontologies belonging to MON whereas the arrows represent `owl:imports` axioms holding among them. The entry point of the Ontology Network is the ontology `mario.owl`[3]. This ontology imports all the knowledge areas, which, in turn, import the ontology modules they contain.

For the development of the MON, we configured eXtreme Design in order to

[2]MARIO Ontology Network, http://www.ontologydesignpatterns.org/ont/mario/
[3]MON entry point, http://www.ontologydesignpatterns.org/ont/mario/mario.owl

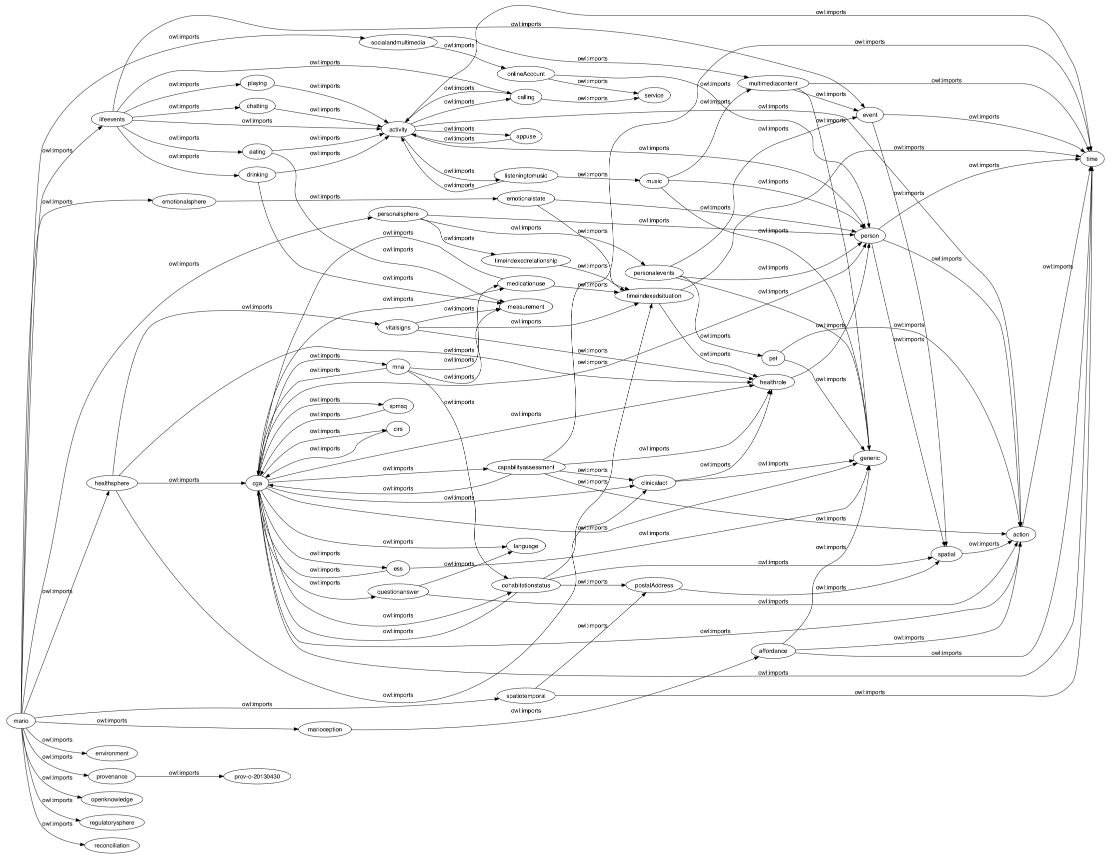


Figure 3.3: The network of ontologies constituting the MARIO Ontology Network.

have an *indirect re-use* of Ontology Design Patterns (ODPs) and alignments. Ontology Design Patterns are used as templates for designing the local ontology, and alignments axioms are provided in order to bind the local entities to the external ones. The alignment axioms are published separately from the core of the ontology[4]. At the same time, this strategy guarantees interoperability by defining appropriate alignments with the external ODPs, and allows the implementations of extensions for satisfying more specific requirements. With this type of re-use, the potential impact of possible changes in the external ODP is minimised. In fact, even if incoherences or changes occur in the external ODP (which is rather unlikely to happen) then the redesign process would be very simple. The ontology signature and axioms would remain unchanged, as incoherences or changes would be resolved by simply

[4] http://etna.istc.cnr.it/mario/ont/alignments.ttl

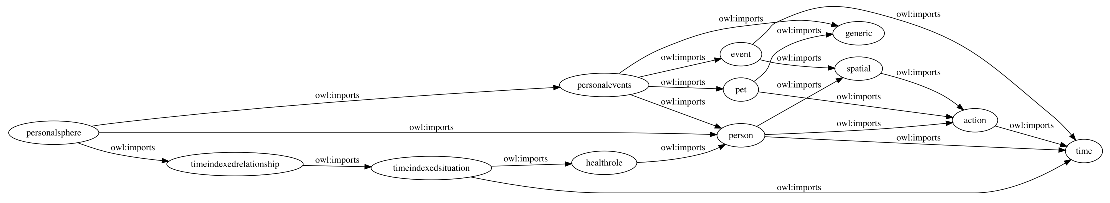
removing or revising the alignment axioms.

3.2 Knowledge Areas

The MARIO Ontology Network (MON) consists of several ontologies that cover different knowledge areas that are relevant for robot's tasks. The knowledge areas were identified by analysing the use cases that emerged from the system specification carried out in the context of the MARIO Project [29]. These uses cases describe actions and behaviors featuring the MARIO robot. Nevertheless, they also provide us with detailed description about the nature of knowledge the robot should deal with. Hence, we highlighted, for each use case, the knowledge domains required to address such a use case. This process was driven by the identification of the *competency questions* [98] from the textual descriptions of the use cases. Thus, the knowledge domains emerged from the competency questions we collected, i.e. the knowledge domains identify the topics involved by competency questions. Finally, we gathered a set of top-level knowledge areas by iteratively generalizing the knowledge domains. This method is similar to the Gronded Theory [199], which is often used in Social Sciences to extract relevant concepts from unstructured corpora of natural language resources (e.g., texts, interviews, or questionnaires). This method allowed us to identify 12 knowledge areas, listed below together with diagram representing the portion of MON included in the area.

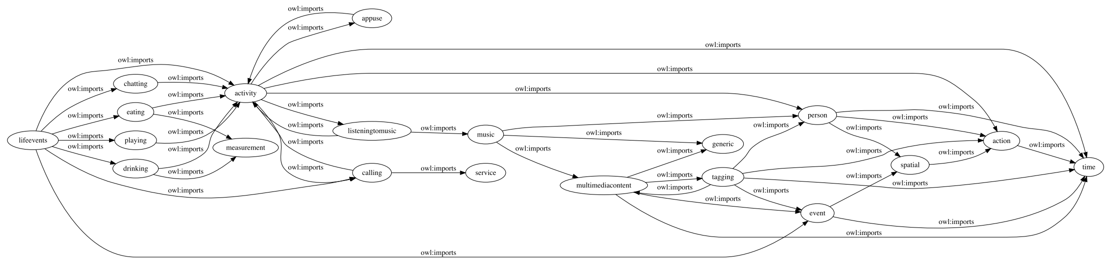
1. **Knowledge Area.** Personal Sphere.
   Description. People information, information about relationship among people, contacts etc.
   Use Cases. *UC3.1.1.5* Capture and load personal data for the use, *UC3.1.1.12* Set up users, *UC3.1.3.2* Assist the user with information about people, *UC3.1.6.2* CGA: Question User about Family.

2. **Knowledge Area.** Life events and patterns.



Description. Information about everyday events, memories, scheduling, plans etc.

Use Cases. *UC3.1.3.1* Add Events, *UC3.1.3.4* Help the user carry out a sequence of actions, *UC3.1.3.5* Inform the user about events, *UC3.1.3.6* Suggest things the user can do, *UC3.1.6.3* CGA: Question user about Daily Living activity, *UC3.1.6.8* Monitor the daily pattern of the user.



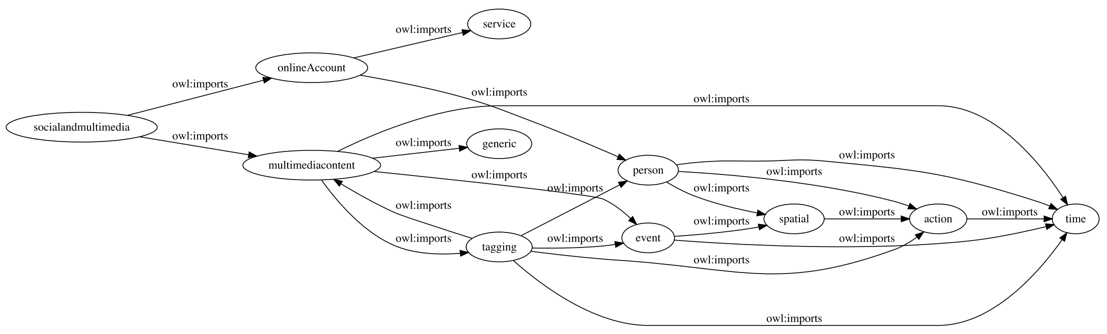
3. **Knowledge Area.** Social and multimedia content.

   Description. Online social network community, multimedia content such as photos, videos, movies, documents.

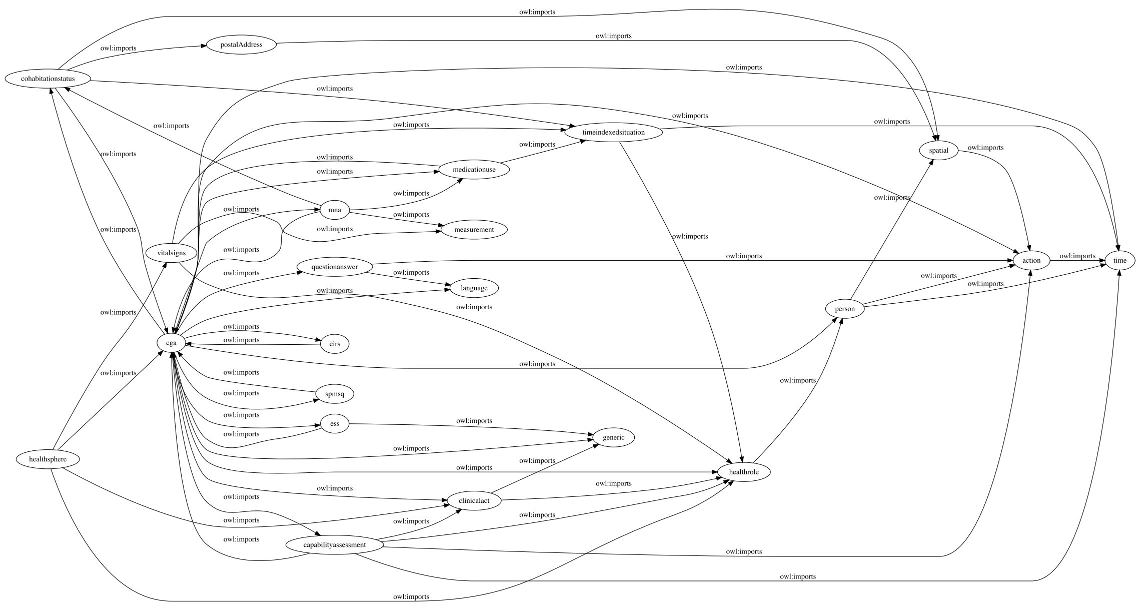
   Use Cases. *UC3.1.1.v1* Choose and pre-load Games for the User, *UC3.1.1.2* Choose and pre-load Music for the User, *UC3.1.1.3* Choose and pre-load Videos for the User, *UC3.1.2.2* Play Music for the User, *UC3.1.2.3* Play a Game with the User, *UC3.1.2.4* Read a text to the User, *UC3.1.2.5* Show a video to the User.

4. **Knowledge Area.** Health sphere.

   Description. Information about living patterns, health patterns, vital signs, anything related to CGA and MPI.



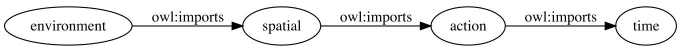
Use Cases. *UC3.1.6.1* CGA: Assess the user when using a Telephone, *UC3.1.6.2* CGA: Question User about Family, *UC3.1.6.3* CGA: Question user about Daily Living activity, *UC3.1.6.4* Question User to Establish Emotional State, *UC3.1.6.5* Carry out a CGA assessment on the user, *UC3.1.6.6* Generate Health reports for the care staff, *UC3.1.6.7* Monitor the Health of the user, *UC3.1.6.8* Monitor the daily pattern of the user, *UC3.1.6.9* Record where the user goes and what they do during the day, *UC3.1.8.1* Ask the user a series of questions to establish facts about them, or examine their heath or how they are feeling.



5. **Knowledge Area.** Environment.

Description. Information about rooms, furnitures, objects etc.

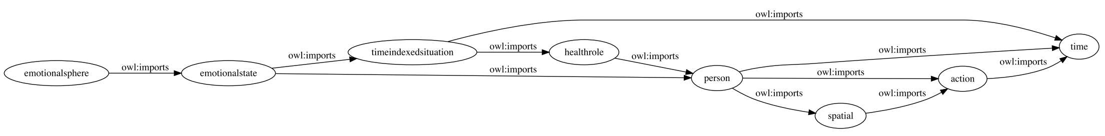
Use Cases. *UC3.1.1.8* Name Locations and rooms on the map, *UC3.1.1.11* Map Operating Environment, *UC3.1.4.1* Approach the user, *UC3.1.4.2* Identify the User in the Immediate Area, *UC3.1.4.3* Search for the User in the Operating Environment, *UC3.1.6.9* Record where the user goes and what they do during the day.



6. **Knowledge Area.** Emotional sphere.

   Description. Information about emotions, sentiments, interests, opinions related to people etc.

   Use Cases. *UC3.1.6.4* Question User to Establish Emotional State, *UC3.1.7.3* Identify and remember what the user likes, *UC3.1.7.4* Show the User some items from the generic reminiscence store that match their era, *UC3.1.7.5* Show the User some items from their personal reminiscence store, *UC3.1.8.1* Ask the user a series of questions to establish facts about them, or examine their heath or how they are feeling.



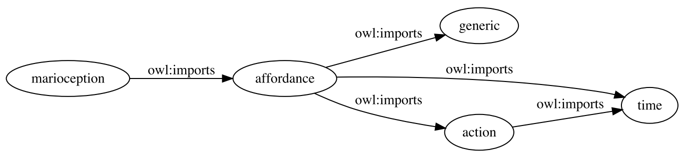
7. **Knowledge Area.** Open knowledge.

   Description. This area encloses unforeseen knowledge retrieved from web (e.g. web sites, news articles etc.) or extracted from the dialogue with users. This knowledge area provides the robot with the means for treating unexpected knowledge.

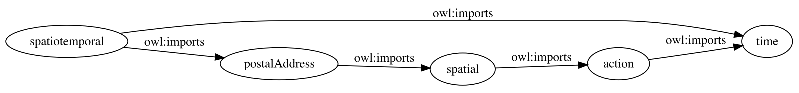
   Use Cases. *UC3.1.3.5* Inform the user about events, *UC3.1.3.6* Suggest things the user can do, *UC3.1.8.1* Ask the user a series of questions to establish facts about them, or examine their heath or how they are feeling.

8. **Knowledge Area.** Regulatory sphere.
   Description. Information about
   norms, rules, social habits etc. **Use Cases.** *UC3.1.3.4* Help the user carry out a sequence of actions, *UC3.1.3.6* Suggest things the user can do.

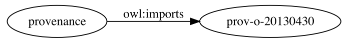
9. **Knowledge Area.** Proprioception.
   Description. Information about MARIO abilities, MARIO functionalities, applications MARIO is able to run, actions MARIO is able to do etc.
   Use Cases. This knowledge area is transversal to all the use cases



10. **Knowledge Area.** Spatio-temporal data.
    Description. Information about spatio-temporal data.
    Use Cases. This knowledge area is transversal to all the use cases



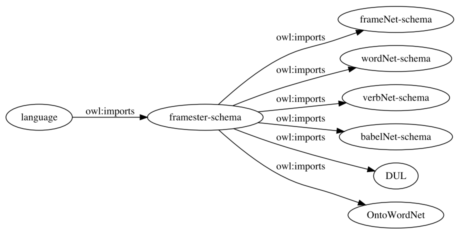
11. **Knowledge Area.** Provenance.
    Description. Information about provenance and validity of facts.
    Use Cases. This knowledge area is transversal to all the use cases



12. **Knowledge Area.** Lexical-linguistic data.

    Description. Data for natural language processing.

    Use Cases. This knowledge area is transversal to all the use cases



3.3 Ontology Modules

In this section we describe the ontology modules composing the MARIO Ontology Network.

3.3.1 Affordance Ontology

In the design of cognitive agents like robots, *behaviour selection* (also called behaviour arbitration) is the process of deciding which action to execute at each point of time. For the sake of simplicity, most implemented systems use a built-in fixed priority ordering of behaviours, i.e. the agent's control strategy is embedded into a collection of preprogrammed condition-action pairs. This strategy, called *purely reactive*, has proven effective for a variety of problems that can be completely specified at design-time [140]. However, it is inflexible at run-time due to its inability to store new information in order to adapt the robot's behaviour on the basis of its experience. Moreover, the burden of predicting all possible input states and choosing the corresponding output actions is completely left to the designer.

Behaviour-based approaches to action selection can be considered as an extension of purely reactive strategy. These approaches are related to the concept of *affordance*. The notion of affordance has been introduced by Gibson [91] who devised a theory of how animals perceive opportunities for action. Gibson called these opportunity affordance. He suggested that the environment offers the agents (people or animals) opportunities for action. For instance, a door can have the affordance of "openability". These action opportunities are latent in the environment and independent from individual's ability to recognize them, but affordances are always dependent on agent's capability. For example, to a thief an open window can afford the "steal" action, but not so to a waitress who may simply be afforded by the "close" action if outside the temperature is too cold.

In this section we present the Affordance Ontology Design Pattern (ODP) that extends the classical notion of affordance, which suggests that the physical objects (e.g., a door) offer the opportunity of performing an action (e.g., open). In fact, our ODP is designed by relying on the assumption that, not only physical objects, but also complex situations (e.g., the user want to listen to some music) afford actions (e.g., play music). A complex situation can be seen as the fullfilment at a certain time of certain conditions. These conditions may involve temporal aspects (e.g. lunchtime may afford the task remember the user to take the pills), the perception of certain physical objects, the receiving of a command (e.g. I want to listen to some music), or, even the existence of certain state-of-affairs (e.g. the situation the user is sitting on a chair for a long while may afford the task entertain the user).

Related Work. There exist few examples of ontologies conceptualising the idea of affordances. In literature, the notion of affordance has been seen either as the relation between the environment and an agent [198], or as qualities of objects in the environment taken with reference to an observer [159, 158, 190]. Our approach is closer to the characterisation proposed by Stoffregen [198], albeit we abstract the notion environment to a more general concept of situation as conceived by Gangemi and Mika [84]. Namely, a situation embeds all the environment's characteristics perceived by the robot and possibly other conditions (e.g. involving time, the receiving of a commands etc.).

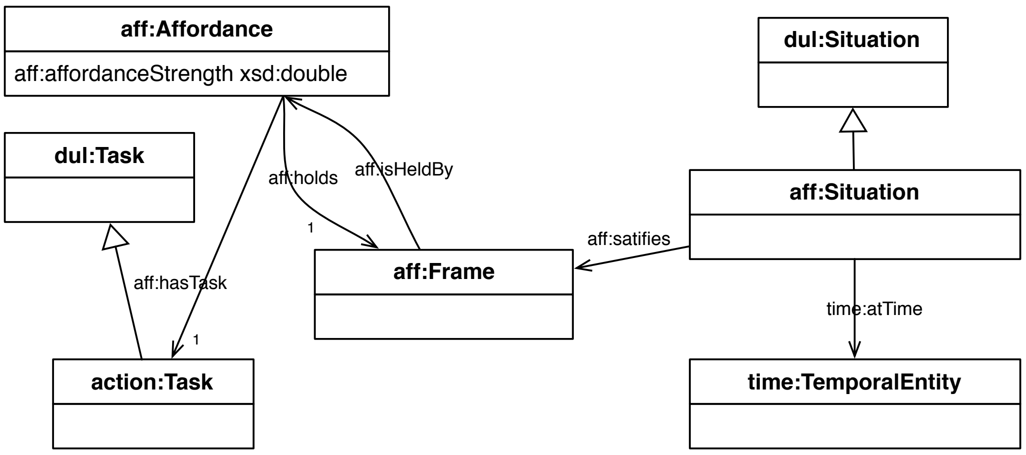


Figure 3.4: The diagram of the Affordance ontology.

ID	Competency question
CQ1	Which is the strength of an Affordance?
CQ2	Which tasks are afforded in a certain situation?
CQ3	How should an agent behave in a certain situation?
CQ4	Which are the parameters involved in certain task?

Table 3.2: Competency questions answered by the Affordance ODP.

Module overview. The proposed pattern relies on the Descriptions and Situations ODP[5] [84], combined with a frame-based representation scheme [153]. Table 3.2 reports the competency questions [98] that drove the design of the Affordance ODP. Figure 3.4 shows the UML class diagram of the ontology. The base namespace is associated with the value `http://www.ontologydesignpatterns.org/ont/mario/affordance.owl#`. The page of the proposed pattern as submitted to the ontologydesignpatterns.org portal is http://ontologydesignpatterns.org/wiki/Submissions:Affordance.

Affordances are represented as individuals of the class `Affordance`, which is modelled as a n-ary relation connecting:

- A class of situations that represents states of the world (i.e., any individual of

[5]Description and Situation ODP http://ontologydesignpatterns.org/wiki/Submissions:DescriptionAndSituation

the class `Frame`). This relation is expressed by means of the object property `holds`;

- An agent's behavior (aka a task), which is any individual of the class `action: Behavior`. This relation is expressed by the object property `hasBehavior`.
- A quantity that indicates how much a behavior is relevant for the occurrence of a certain frame. This relation is expressed by the datatype property `affordanceStrength`, whose range is `xsd:double`.

According to [89], the intended meaning of a frame (represented in our ODP by the class `Frame`) can be summarised as a small-sized and richly interconnected structure, used to organize our knowledge, as well as to interpret, process or anticipate information. Frames identify classes of situations and have been investigated in linguistics by Fillmore [75], in AI by Minsky [147], and more recenty in the Semantic Web [89, 153]. We modelled the class `Frame` as a sub-class of `framester:Frame`[6], which is a class re-used from the Framester Ontology [81]. `framester:Frame` extends the class `fn:Frame`[7] of the the OWL version [153] of FrameNet [19].

Situations are states of the world fulfilling certain conditions. These conditions may involve: temporal aspects, the perception of physical entities, the receiving of a command or the existence of certain state-of-affairs. Following the Description and Situation ODP we made a basic distinction, between a `Frame` (or description) and a `Situation`, which is a frame occurence. The class `Situation` is modelled as sub-class of the class `dul:Situation`[8] that is re-used from DOLCE Ultra-lite [83]. Any individual of `Situation` is modelled as a time indexed situation, i.e., a state of the world anchored to a certain time point (e.g. at 11am the user expresses the willingness to listen to jazz music). We re-used the time-indexed situation

[6]The prefix `framester:` stands for the namespace https://w3id.org/framester/schema/.

[7]The prefix `fn:` stands for the namespace http://www.ontologydesignpatterns.org/ont/framenet/tbox/.

[8]The prefix `dul:` stands for the namespace http://www.ontologydesignpatterns.org/ont/DUL.owl#.

ODP[9] for modelling time constraints for situations. Hence, a `Situation` is related to `time:TemporalEntity`[10] that allows to represent the notion of time either as a time interval (i.e., any individual of the class `time:TimeInterval`), which has a start and an end instant, or an instant itself (i.e., any individual of the class `time:Instant`) that is associated with temporal values by means of the datatype property `time:inXSDDataTime` whose range is `xsd:dateTime`[11].

Our ODP models agent's behaviours as tasks. Those tasks are represented as individuals of the class `action:Behavior` that can be parameterised by specific parameters represented as individuals of the class `BehaviorParameter`. The relarions between tasks and task parameters are expressed by the object property `hasParameter`. For example, a certain task "Play music" can be associated with a parameter "Jazz" that specifies the genre of the music to play.

Behaviours are always executed by actions. An action is represented as an individual of the class `action:Action` and can expect the execution of multiple tasks. The association between tasks and actions is represended by the object property `action:executes`. This design reflects the way actions and tasks are modelled in DOLCE. Hence, the classes `action:Action` and `action:Behavior` are represented as sub-classes of `dul:Action` and `dul:Task`, respectively.

Actions are performed by agents. An agent is represented as an individual of the class `action:Agent`, which in turn is designed to be sub-class of `dul:Agent`. The relation between an action and an agent is expressed by the object property `action:byAgent`.

Usage Scenario. The affordance, as introduced by Gibson [91], has been investigated in robotics in the context of behaviour-based approaches to action selection. We are experimenting with behaviour-based approaches in MARIO. To the best of our knowledge this is the first attempt to formalise the notion of affordance as an

[9]http://www.ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl.

[10]The prefix `time:` stands for the namespace http://www.ontologydesignpatterns.org/ont/mario/time.owl#.

[11]The prefix `xsd:` stands for the namespace http://www.w3.org/2001/XMLSchema#.

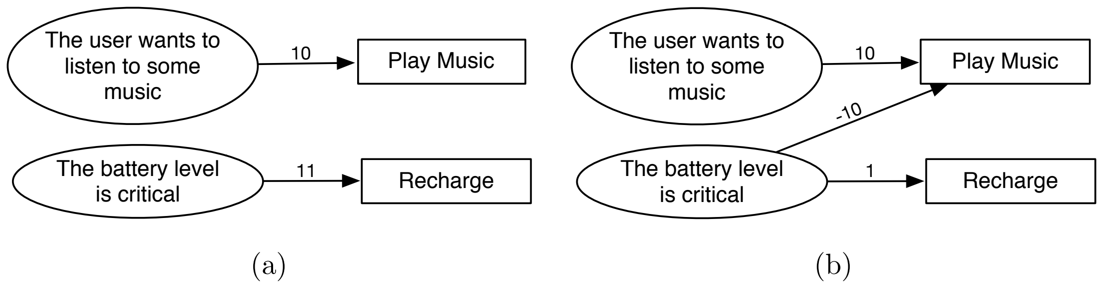

The user wants to listen to some music
10
Play Music
The battery level is critical
11
Recharge
(a)
The user wants to listen to some music
10
Play Music
-10
The battery level is critical
1
Recharge
(b)

Figure 3.5: Two equivalent action-selection schemes.

Ontology Design Pattern and to use it in the context of behaviour-based robotics. In fact, MARIO uses a behaviour-based approach to action selection which relies on both the notion of affordance devised by Gibson [91] and the proposal of Pattie Maes [136]. MARIO exploits the Affordance ODP for dynamically decide which action to perform in specific situations.

Figure 3.5 shows a simple example scenario. This scenario is about two alternative configurations of an affordance model (i.e., Figures 3.5a and 3.5b, respectively) of a cognitive agent (i.e., the MARIO robot in our case). Both configurations are composed of associations of known frames (represented as ovals) with some actions (represented as rectangles) that the agent can perform to react to such situation descriptions. The associations are weighted relations that convey affordance strengths. A configuration is determined by the recognition of some situation (i.e., a frame occurence) by the agent that satisfies a known frame. The first configuration (i.e., Figure 3.5a) comes from the recognition of two distinct and concurrent situations, e.g., *(i)* the patient asks the robot to play her favourite music; *(ii)* and the robot battery status is at 1%. In this configuration the robot is caused to prefer to recharge its battery instead of playing some music. In fact, the affordance strength with value 11 and associated with the battery level frame is greater than the affordance strength with value 10 associated with the play music action. It is worth noting that the second configuration (i.e., Figure 3.5b) leads to the same effect, namely, the agent react by performing the recharge action. In fact, when both situations contemporary hold, the negative affordance strength of the second situation on play music nullifies the affordance strength of the first one. As a consequence, the agent chooses to perform recharge which results to be the action with highest affordance value. The following RDF triples (in Frame 3.1), serialised as TURTLE, model the configuration represented in Figure 3.5b according to the Affordance ODP.

```
:UserWantsToListenToSomeMusic a aff:Frame;
fn:hasFrameElement :genre, :user.

:genre a fn:FrameElement, aff:BehaviorParameter .

:BatteryInCritalLevel a aff:Frame;
fn:hasFrameElement :batteryLevel, :agent .

:batteryLevel a fn:FrameElement .

:agent a fn:FrameElement .

:user a fn:FrameElement .

:PlayMusic a action:Behavior;
aff:hasParameter :genre .

:Recharge a aff:Behavior.

:affordancePlayMusicBatteryCritical a aff:Affodance ;
aff:affordanceStrength "-10"^^xsd:double ;
aff:holds :BatteryInCritalLevel ;
aff:hasBehavior :PlayMusic  .

:affordancePlayMusicUserWantsToListenToSomeMusic a aff:Affodance ;
aff:affordanceStrength "10"^^xsd:double ;
aff:holds :UserWantsToListenToSomeMusic ;
aff:hasBehavior :PlayMusic  .

:affordanceRechargeBatteryCritical a aff:Affodance ;
aff:affordanceStrength "1"^^xsd:double ;
```

```
aff:holds :BatteryInCritalLevel ;
aff:hasBehavior :Recharge .

:sit1 a aff:Situation ;
:user :Freddy ;
:genre :Jazz ;
time:atTime :Time1 ;
dul:satisfies :UserWantsToListenToSomeMusic .

:sit2 a aff:Situation ;
:batteryLevel “1''^^xsd:integer ;
:agent :MARIO ;
time:atTime :Time1 ;
dul:satisfies :BatteryInCritalLevel .

:actRechargeAtTime1 a action:Action ;
action:byAgent :MARIO ;
action:executes :BatteryInCritalLevel ;
time:atTime :Time1 .
```

Frame 3.1: An example of usage of the Affordance ontology serialized in TURTLE language

`:UserWantsToListenToSomeMusic` represents the frame where a `:user` request to listen to some music of a particular `:genre`. `:BatteryInCritalLevel` represents the frame where the `:batteryLevel` of a certain `:agent` is critical. `:affordance-PlayMusicBatteryCritical`, `:affordancePlayMusicUserWantsToListenToSome-Music` and `:affordanceRechargeBatteryCritical` represent the three affordance relations depicted in figure 3.5b as arrows. `:sit1` and `:sit2` represent the situation that hold at time 1. `:actRechargeAtTime1` is the action carried out by `:MARIO` to cope with the situations `:sit1` and `sit2`.

3.3.2 Comprehensive Geriatric Assessment Ontology

The evaluation of elderly patients represents a complex universe difficult to evaluate and manage as a unique body. The Comprehensive Geriatric Assessment (CGA) represents one of the most used and validated approaches to evaluate the elderly subjects. It is defined as a "multidimensional interdisciplinary diagnostic process focused on determining a frail older person's medical, psychological and functional capability in order to develop a coordinated and integrated plan for treatment and long term follow up" [185]. It can be roughly viewed as a set of tests aiming at assessing the medical, psychological and functional capability of a person.

One of the objectives of the MARIO project was to take advantage of the continuous monitor of a an elderly provided by a social robot so to facilitate the assessment of the status of the elderly patient. The continuous monitoring activity aims also to improve the communication inside the team who is in charge of the care of the subject and to manage and check rehabilitation plans. In the context of MARIO project a customized CGA has been defined with the following instruments: i) *Activities of Daily Living* (ADL) [109] and *Instrumental Activities of Daily Living* (IADL) [119] for evaluating functional disabilities in the daily living; ii) *Short-Portable Mental Status Questionnaire* (SPMSQ) [166] for assessing the cognitive status for dementia screening; iii) *Mini-Nutritional Assessment* (MNA) [212] for assessing the nutritional status; iv) *Exton-Smith scale* (ESS) [32] for evaluating the risk of pressure sores in patients at high risk of immobilization or bed-ridden; v) *Comorbidity Illness Rating Scale* (CIRS) [51] for carefully evaluating the the comorbidities; vi) Evaluation of medication use for assessing the appropriateness of prescriptions, and the risk for adverse drug reactions.

The contribution of the CGA ontology is twofold. On the one hand, the ontology supports the execution of the assessment by providing a reference model for storing test information (such as questions, expected answer etc.). On the other hand, it allows to store the data resulting from the patient's assessments.

Related Work. Medicine is one of the first fields that employed ontologies in knowledge-base systems [170]. Ontologies have been used to create an unified medical language [34, 87], to build a computer based patient record[12], to allow the representation of clinical narratives [204], to support professional decisions in the life-cycle of home care treatments [177], to an ontology-driven adaptive medical questionnaire [35] and so on. To the best of our knowledge it does not exist any ontology able to represents the results of an execution of our customization of the CGA. Some ontologies have been proposed for supporting the (general) medical assessment process [36, 25]. This ontologies define high-level concepts for representing medical assessment. The CGA ontology follow their approach and specializes the high-level concepts where needed.

Module overview. The choice of customizing the Comprehensive Geriatric Assessment impedes the re-use of an off-the-shelf ontology for the CGA. The requirements of the ontology have been directly derived from the template[13] used by physicians during the assessment of an elderly patient. The competency questions extracted by analyzing these documents can be found at[14].

The ontology is modular, meaning that it includes a different module for each test in the MARIO's customization of the CGA. The modules composing the CGA ontology (together with the namespace specification) can be found in Table 3.3.

The submodules addressing specific tests specialize the CGA ontology on the basis of the specific requirements of the test. For example, the CGA ontology defines the class `cga:GeriatricAssessment`, encompassing all geriatric assessment performed by an agent, and the ontology addressing ADL and IADL specializes that class with `ca:CapabilityAssessment`, representing only the assessment aimed at evaluating the capabilities of elderly patients.

The Figure 3.6 shows the UML class diagram of the CGA ontology. As in [25],

[12]CPRO, http://ontohub.org/bioportal/CPRO.owl

[13]MARIO's settings for the customized Comprehensive Geriatric Assessment, http://etna.istc.cnr.it/mario/D5.1/CGA.pdf

[14]http://etna.istc.cnr.it/mario/D5.1/CQ-CGA.pdf.

Namespace prefix	Namespace
`cga`	http://www.ontologydesignpatterns.org/ont/mario/cga.owl#
`coh`	http://www.ontologydesignpatterns.org/ont/mario/cohabitationstatus.owl#
`ca`	http://www.ontologydesignpatterns.org/ont/mario/capabilityassessment.owl#
`spmsq`	http://www.ontologydesignpatterns.org/ont/mario/spmsq.owl#
`ess`	http://www.ontologydesignpatterns.org/ont/mario/ess.owl#
`cirs`	http://www.ontologydesignpatterns.org/ont/mario/cirs.owl#
`mna`	http://www.ontologydesignpatterns.org/ont/mario/mna.owl#
`action`	http://www.ontologydesignpatterns.org/ont/mario/action.owl#
`clinicalact`	http://www.ontologydesignpatterns.org/ont/mario/clinicalact.owl#
`time`	http://www.ontologydesignpatterns.org/ont/mario/time.owl#

Table 3.3: Ontology modules imported/reused by the CGA ontology.

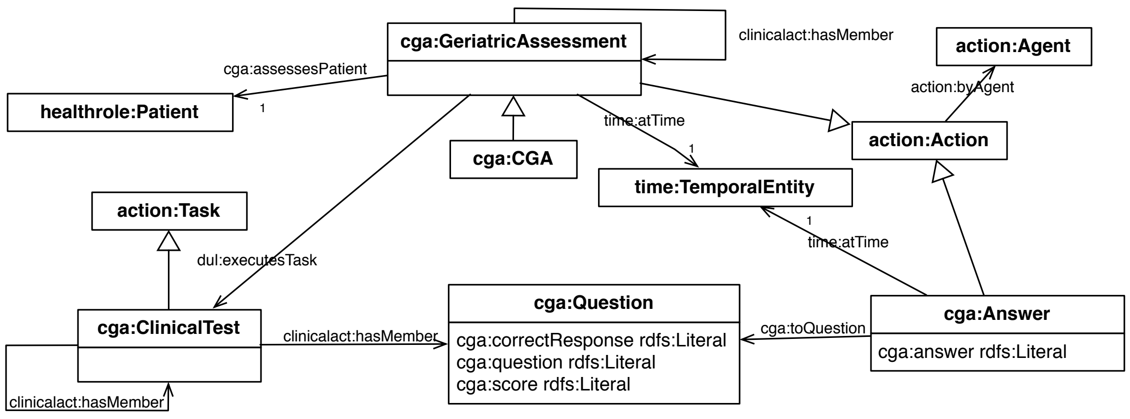


Figure 3.6: The UML class diagram of CGA ontology.

a patient assessment (i.e. `cga:GeriatricAssessment`) is an action having as participant the assessed `healthrole:Patient` and an `action:Agent`[15] who make the assessment. The agent making the assessment can be either a `healthrole:Physician` or another kind of agent (e.g. MARIO). In order to represent the *description* of how the assessment is to be executed, we implemented the Ontology Design Pattern *Task Execution*[16]. The action `cga:GeriatricAssessment` executes a `cga:ClinicalTest` which provides a "*description*" of how the assessment has to be executed. A `cga:ClinicalTest` can be composed of other clinical tests or some `cga:Question`.

[15]Since `cga:GeriatricAssessment` specializes the class `action:Action`

[16]Task execution ODP http://www.ontologydesignpatterns.org/cp/owl/taskexecution.owl

Furthermore the CGA ontology allows to store information about the answers (i.e. `cga:Answer`) a patient provides to reply a question.

Usage Scenario. The Frame 3.2 shows an example of usage of the CGA ontology. The resource `:CGA` represents the CGA (intended as "*test*"), whereas the resource `:CGA-20160617` represents the actual execution of `:CGA`, performed on 17 June 2016, for assessing the patient `:Freddy`. In this example `:CGA` is composed only of the test `:SPSMQ` which represents a questionnaire containing only of the question (i.e. `:Q1-SPMSQ`) "Who is the president now?". `:Q1-SPMSQ` defines the conditions under which the answer provided by the patient has to be considered correct (i.e. "Requires only the correct last name") and the score to be given if the patient properly responds.

The actual execution of the CGA effectuated by the agent `:DrRossi` for assessing the patient `:Freddy` is represented by `:CGA-20160617`. Within `:CGA-20160617` the Short Portable Mental Status Questionnaire is performed, i.e. `:SPMSQ-20160616`. Freddy's answer (i.e. `:Answer-Freddy-Q1-20160616`) to question Q1 is "Mattarella". The answer has been considered correct by the `:DrRossi` who gave the score "1" to it. The answer's assessment effectuated by the `:DrRossi` is represented by `:Answer-Freddy-Q1-20160616-Assessment`.

```
:CGA a  cga:ClinicalTest ;
  clinicalact:hasMember cga:SPMSQ .

:SPMSQ a cga:ClinicalTest ;
  clinicalact:hasMember cga:Q1-SPMSQ .

:Q1-SPMSQ a cga:Question ;
  cga:correctResponse
    "Requires only the correct last name"@en ;
  cga:question "Who is the president now?"@en ;
  cga:score "1" .

:CGA-20160617
```

```
  a cga:ComprehensiveGeriatricAssessment ;
  action:byAgent :DrRossi ;
  cga:executesTask :CGA ;
  clinicalact:hasMember :SPMSQ-20160616 ;
  cga:assessesPatient :Freddy .

:SPMSQ-20160616
  a spmsq:ShortPortableMentalStatusQuestionnaire ;
  action:byAgent :DrRossi ;
  cga:executesTask :SPMSQ ;
  clinicalact:hasMember
    :Answer-Freddy-Q1-20160616-Assessment ;
  cga:assessesPatient :Freddy .

:Answer-Freddy-Q1-20160616 a cga:Answer ;
  action:byAgent :Freddy ;
  cga:toQuestion :Q1-SPMSQ ;
  cga:answer “Mattarella'' .

:Answer-Freddy-Q1-20160616-Assessment
  a spsmq:AnswerAssessment ;
  score “1'';
  action:byAgent :DrRossi ;
  cga:derivedFrom :Answer-Freddy-Q1-20160616 .
```

Frame 3.2: An example of usage of the CGA ontology and the SPMSQ ontology serialized in TURTLE language.

3.3.2.1 CGA Ontology Modules

In this section we provide an overview of the ontology modules composing the CGA ontology.

Co-Habitation Status. The aim of the “*Co-Habitation Status*” ontology is to provide a reference model for representing the habitation status of a patient so to allow to make an assessment on it. The ontology, shown in Figure 3.7, answers to the Competency Question in Table 3.4. The namespace prefix `coh` is associated with `http://www.ontologydesignpatterns.org/ont/mario/cohabitationstatus.owl#`. The ontology implements the ODP *Time Indexed Situation*[17] to

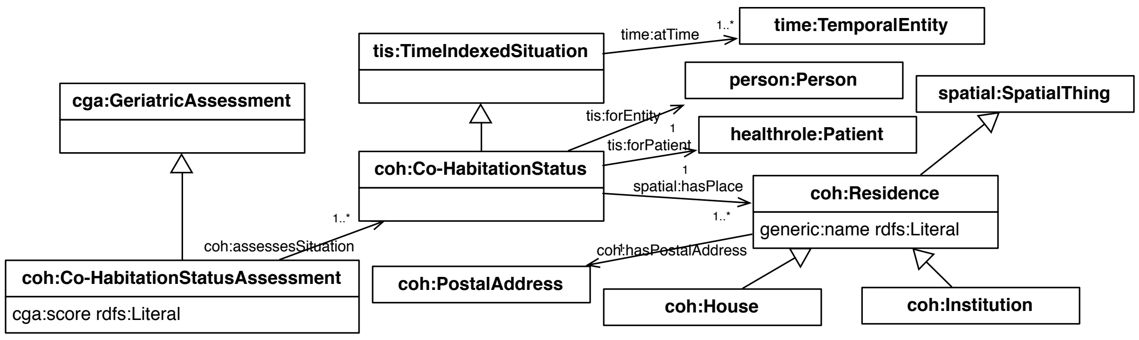


Figure 3.7: The UML class diagram of the Co-Habitation status ontology.

CQ1	Does the patient live alone or with relatives/nurse or in an institution?
CQ2	Which is the address of the place where a patient lives?
CQ3	Which is the name of the place where a patient lives?
CQ4	Which is the assessment for a certain co-habitation status of a certain patient?

Table 3.4: Competency questions answered through the Co-Habitation status ontology.

represent the `coh:Co-HabitationStatus` of a `healthrole:Patient` at given time. `coh:Co-HabitationStatus` can be seen as a n-ary relation connecting (i) a patient, (ii) its `coh:Residence` (an `coh:House` or an `coh:Institution` characterized by a name and a `coh:PostalAddress`), (iii) and possibly some cohabitants. A `coh:Co-HabitationStatus` is assessed by a `coh:Co-HabitationStatusAssessment` which assigns a score to it.

Medication Use. The ontology module "*Medication Use*" enables to keep track the medication use of a patient thus allowing to make an assessment on it. The on-

[17]Aldo Gangemi, Time Indexed Situation ODP, http://ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl

CQ1	Which is the assessment for a certain medication use of a certain patient?
CQ2	How many drugs does a patient use?

Table 3.5: Competency questions answered through the Medication Use ontology.

tology, shown in Figure 3.8, answers to the Competency Question in Table 3.5. The namespace prefix `medicationuse` is associated with `http://www.ontologydesignpatterns.org/ont/mario/medicationuse.owl#`. The *Medication Use* is a *Time*

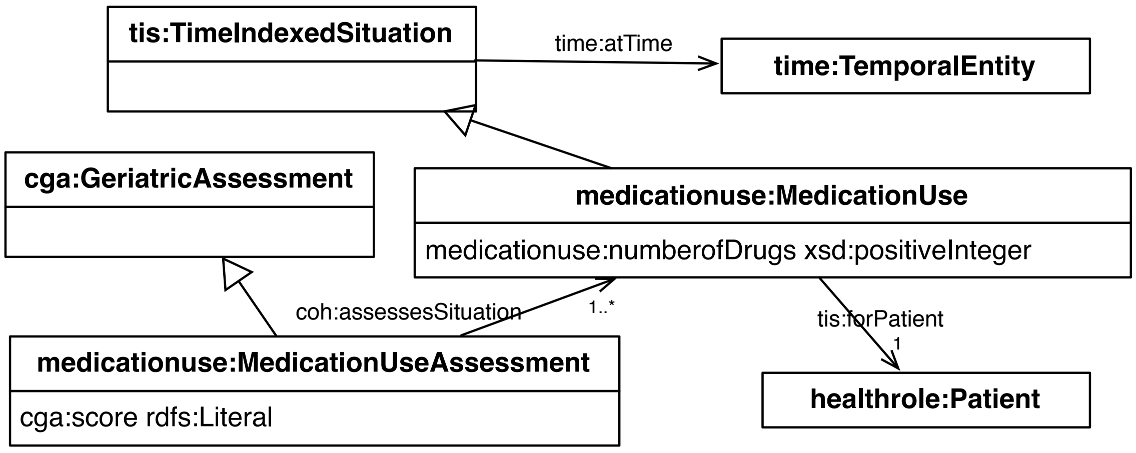


Figure 3.8: The UML class diagram of the Medication Use ontology.

Indexed Situation [17] involving i) the Patient targeted of certain treatments; ii) the number of medications used by him; iii) and the time period in which he takes the medications. A `medicationuse:MedicationUse` is assessed by a `medicationuse:MedicationUseAssessment` which assigns a score to it.

Capability assessment. The ontology module "*Capability assessment*" allows to store into a knowledge base the results of an execution of both *Activities of Daily Living* (**ADL**) and *Instrumental Activities of Daily Living* (**IADL**). The Figure 3.9 shows the UML class diagram of the Capability Assessment ontology. The ontology allows to answer the competency questions listed in Table 3.6. The namespace prefix `ca` is associated with `http://www.ontologydesignpatterns.org/ont/mario/capabilityassessment.owl#`. The *Capability Assessment* on-

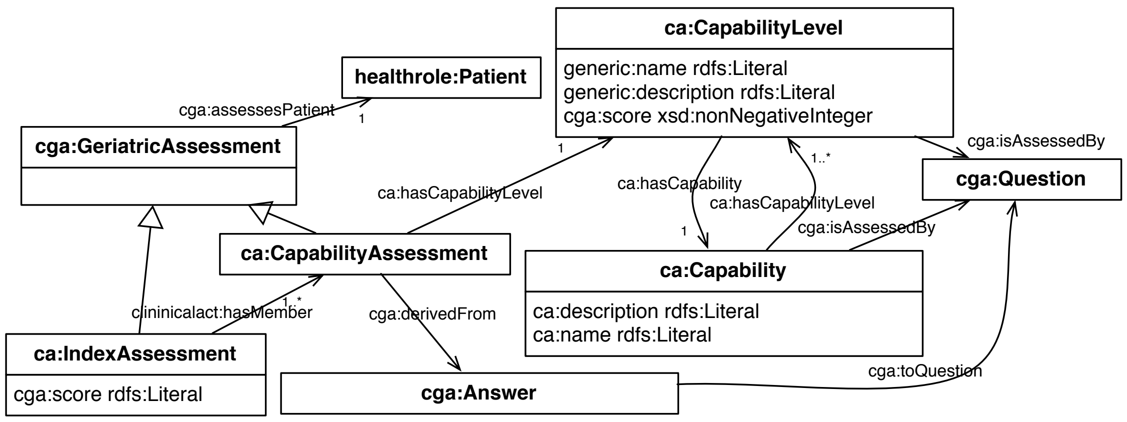


Figure 3.9: The UML class diagram of the Capability Assessment ontology.

tology defines two `cga:GeriatricAssessment`, i.e. the `ca:IndexAssessment` and `ca:CapabilityAssessment`. The former allows to represent the assessment (e.g. an execution of ADL or IADl) made on the basis of a set of capability assessments (e.g. Bathing, Dressing etc.). Therefore a `ca:IndexAssessment` specifies the set of `ca:CapabilityAssessment` on which the the assessment is made and the resulted total score (`cga:score`). The latter, `ca:CapabilityAssessment` is used to evaluate a patient's capability in performing the activities of daily living. A `ca:Capa bilityAssessment` is an action used to associate a certain patient with a certain `ca:CapabilityLevel` at a certain time. The `ca:CapabilityAssessment` could be derived from a `cga:Answer` that the patient provide to reply to a `cga:Question` (e.g. *Do you need assistance for bathing? - No*). `ca:CapabilityLevel` is used to define the capability levels for a certain capability to assess. Each `ca:CapabilityLevel` is characterized by the `ca:Capability` (e.g. *Bathing*) it refers to, its `generic:name` (e.g. *Receives no assistance*), a `generic:description` (e.g. *gets in and out of tub by self if tub is usual means of bathing*) and a `cga:score` to give if the patient being assessed shows that level of capability (e.g. 1).

Short Portable Mental Status Questionnaire (SPMSQ). The *SPMSQ ontology* allows to store the results of a *Short Portable Mental Status Questionnaire* into the Knowledge Base. The Figure 3.10 shows the UML diagram of the ontology.

CQ1	Which are the capabilities needed to assess the ADL/IADL?
CQ2	Which is the ADL/IADL question used to assess the capability level of a certain patient?
CQ3	Which is the description of a certain capability level?
CQ4	Which is the score in ADL/IADL associated with a certain capability level?
CQ5	Which is the score of a correct answer of a certain question?
CQ6	Which is the total score of the ADL/IADL questionare of a certain patient at a certain time?
CQ7	Which was the level of capability of the activities of daily living (such as, bathing, dressing, toileting, transferring, controlling urination and bowel movement, feeding) of a patient at a certain time?
CQ8	Which was the level of capability of the instrumental activities of the daily living (such as, using a telephone, shopping, preparing food, housekeeping, laundering, getting means of transportation, having responsibility of own medications, managing finances) of a patient at a certain time?

Table 3.6: Competency questions answered through the Capability Assessment ontology.

The CQs answered by the SPMSQ ontology are listed in Table 3.7. The namespace prefix `spmsq` is associated with `http://www.ontologydesignpatterns.org/ont/mario/spmsq.owl#`. An individual of type `spmsq:ShortPortableMentalStatus-`

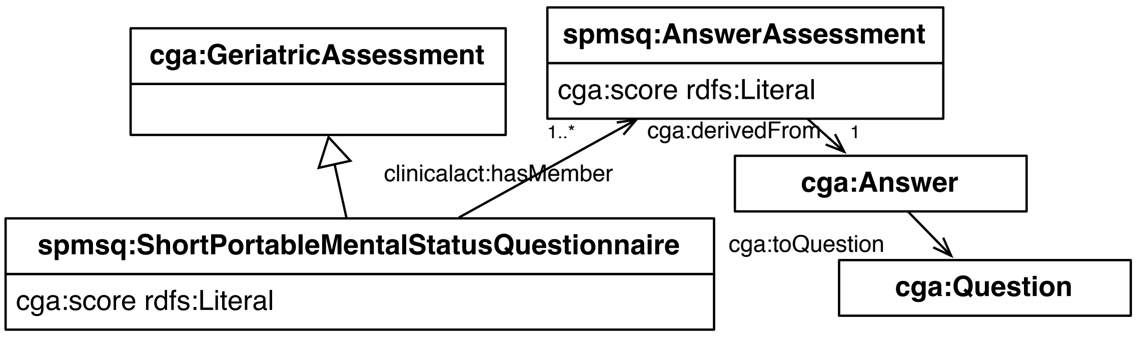


Figure 3.10: The UML class diagram of the SPMSQ ontology.

CQ1	Which is the total score of the SPMSQ at a certain time?
CQ2	Which is the score associated with an answer of the SPMSQ?

Table 3.7: Competency questions answered through the SPMSQ ontology.

`Questionnaire` is created when the questionnaire for assessing the mental status of a patient is terminated. This instance provides the total result of the SPMSQ (i.e. `cga:score`) scored by the assessed Patient. Each `cga:Answer` of the Patient being assessed is associated with a `spmsq:AnswerAssessment` providing the evaluation of that answer. `spmsq:ShortPortableMentalStatusQuestionnaire` aggregates all the `spmsq:AnswerAssessment` and provides the sum scored answering to the individual questions. An example of usage of the SPMSQ ontology is shown in the Frame 3.2.

Exton-Smith Scale (ESS). The *ESS ontology* allows to store into the KB the results of an evaluation of a Patient through the Exton-Smith Scale. The UML class diagram of the ESS ontology is shown in Figure 3.11. The CQs answered through the ESS ontology are listed in Figure 3.8. The namespace prefix `ess` is associated with the value `http://www.ontologydesignpatterns.org/ont/mario/ess.owl#`. The *Exton-Smith Scale* aims at evaluating the pressure sores risk of a Patient. `ess:Exton-SmithScaleAssessment` represents an execution of this assessment. It associates the Patient that is being assessed with a `cga:score` and with the observed patient conditions that induced to this score. Moreover, `ess:Exton-SmithScaleAssessment` associates (through `ess:hasPressureSoresRisk`) a Patient with a `ess:PressureSoresRisk`. An individual of type `ess:PressureSoresRisk` represents a level of pressure sores risk on the ESS scale (e.g. "*Score 16-20: minimum risk*"). A `ess:PressureSoresRisk` is characterized by a `generic:name` (e.g. "*minimum risk*") and the interval (i.e. `cga:scoreMin` and `cga:scoreMax`) of values associated with the pressure sores risk (e.g. *16-20*). The object property `ess:hasCondition` is used to associate an `ess:Exton-SmithScaleAssessment` with the current `ess:PatientCondtion` observed in the assessed Patient. The onto-

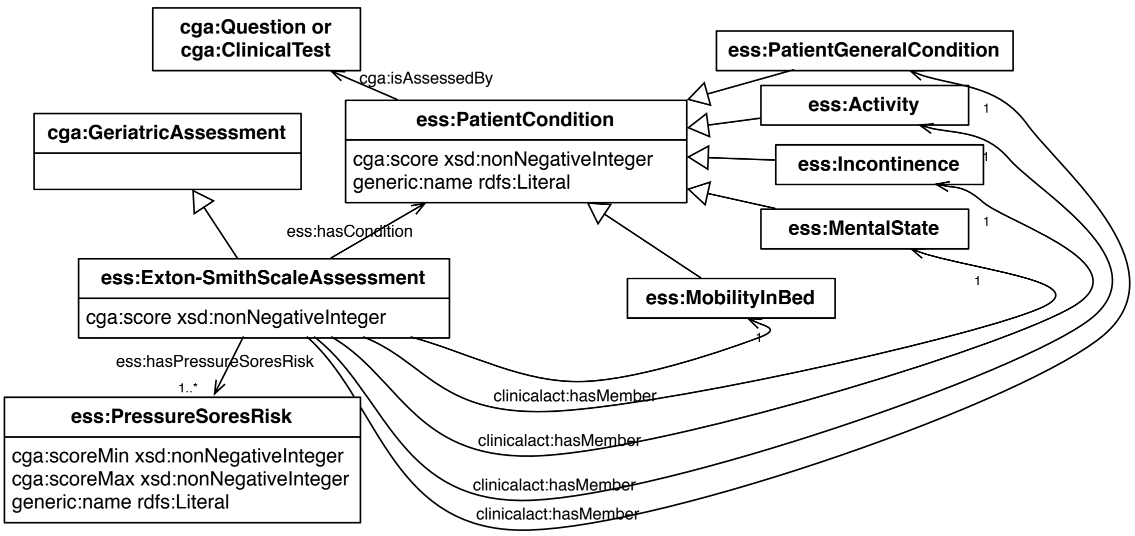

cga:Question or cga:ClinicalTest
cga:isAssessedBy
ess:PatientGeneralCondition
ess:Activity
ess:PatientCondition
cga:score xsd:nonNegativeInteger
generic:name rdfs:Literal
cga:GeriatricAssessment
ess:Incontinence
ess:MentalState
ess:hasCondition
ess:Exton-SmithScaleAssessment
cga:score xsd:nonNegativeInteger
ess:MobilityInBed
ess:hasPressureSoresRisk
1..*
ess:PressureSoresRisk
cga:scoreMin xsd:nonNegativeInteger
cga:scoreMax xsd:nonNegativeInteger
generic:name rdfs:Literal
clinicalact:hasMember
clinicalact:hasMember
clinicalact:hasMember
clinicalact:hasMember
1
1
1
1
1

Figure 3.11: The UML class diagram of the ESS ontology.

logy defines five different types of patient condition that are evaluated by the ESS: the `ess:PatientGeneralCondition`, the `ess:MentalState`, the `ess:Activity`, the `ess:Incontinence` and the `ess:MobilityInBed`. Each `ess:PatientCondtion` has a `generic:name` (e.g. "*Doubly incontinent*") and a `cga:score` (e.g. 1) that can contribute to the result of the `ess:Exton-SmithScaleAssessment`.

Cumulative Illness Rating Scale (CIRS). The *CIRS ontology* allows to store the results of an evaluation of a patient with the *Cumulative Illness Rating Scale.* The UML class diagram of the CIRS ontology is shown in Figure 3.12. The CQs answered through the ESS ontology are listed in Figure 3.9. The namespace prefix `ess` is associated with the value `http://www.ontologydesignpatterns.org/ont/mario/cirs.owl#`. The ontology aims at evaluating the illness severity of the patient's biological system (e.g. *cardiovascular system*, *respiratory system* etc.). The `cirs:BiologicalSystems` are characterized a `generic:name` (e.g.*respiratory system*) and a `generic:description` (e.g. *lungs, bronchi, trachea*). `cirs:BiologicalSystemAssessment` represents an action aiming at assessing a `cirs:BiologicalSystem`. `cirs:BiologicalSystemAssessment` assigns at certain time a `cirs:CIRSRating` (e.g. *Moderate*,3) to a `cirs:BiologicalSystem`. A `cirs:CIRSRating` has

CQ1	How was the condition of patient at a certain time? Was it Bad, Poor, Fair or Good?
CQ2	How was the mental state of a patient at a certain time? Was it Stuporosous, Confused, Apathetic or Alert?
CQ3	Which is the range of ESS scores associated with the high/medium/low risk of sores?
CQ4	Which is the score associated with a certain patient activity level/condition/mental state/incontinence level/mobility?
CQ5	Which was the patient's activity level at a certain time? Did s/he stay in the bed all the day? Did s/he need of a chairfast? Did s/he walk with help? Or, was s/he ambulant?
CQ6	Which was the patient's incontinence level at a certain time? Was s/he double incontinent? Was s/he usually incontinent of urine? Was s/he occasionally incontinent? Or, wasn't s/he incontinent?
CQ7	Which was the patient's mobility in bed at a certain time? Was s/he immobile, very limited, slightly limited, or full?
CQ8	Which was the patient's score at the exton-smith scale (ESS) at a certain time?

Table 3.8: Competency questions answered through the ESS ontology.

a `generic:name` (e.g. *Moderate*) and a `cga:score` (e.g. 3) which may contribute to the illness severity score and to the comorbidity index. An instance of `cirs:CIRSAssessment` evaluates the illness severity score and the comorbidity index of a Patient at certain time. It indicates (through `clinicalact:hasMember`) the `cirs:BiologicalSystemAssessment` used to compute these two scores.

Mini Nutritional Assessment (MNA). The *Mini Nutritional Assessment* (MNA) aims at evaluating the nutritional status of a patient at a certain time. The *MNA ontology* allows to store the results of the Mini Nutritional Assessment into the Knowledge Base. The UML class diagram of the MNA ontology is shown in Fig-

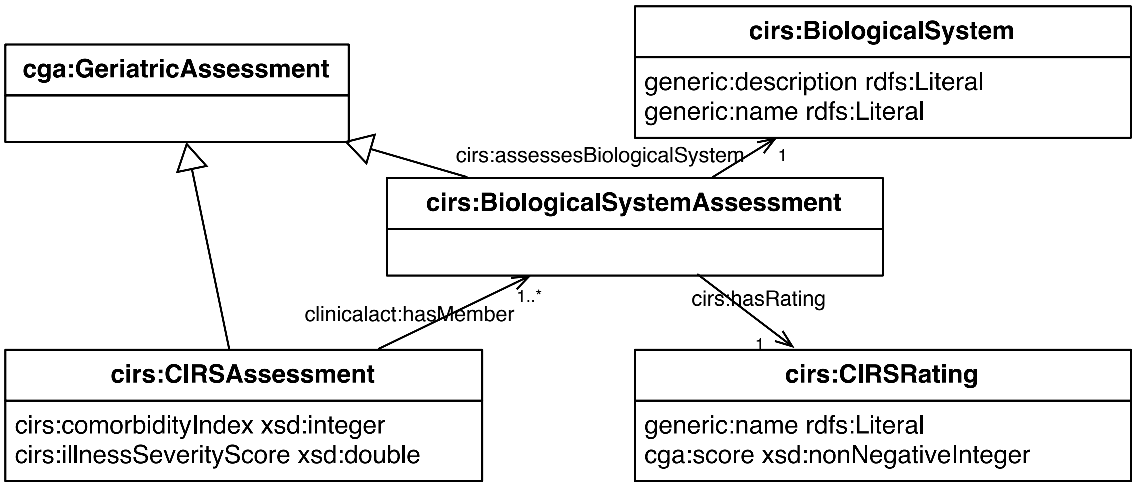

cga:GeriatricAssessment
cirs:BiologicalSystem
generic:description rdfs:Literal
generic:name rdfs:Literal
cirs:assessesBiologicalSystem
1
cirs:BiologicalSystemAssessment
clinicalact:hasMember
1..*
cirs:hasRating
1
cirs:CIRSAssessment
cirs:comorbidityIndex xsd:integer
cirs:illnessSeverityScore xsd:double
cirs:CIRSRating
generic:name rdfs:Literal
cga:score xsd:nonNegativeInteger

Figure 3.12: The UML class diagram of the CIRS ontology.

CQ1	Which is the name of a rating in the CUMULATIVE ILLNESS RATING SCALE (C.I.R.S.)?
CQ2	Which was the patient's illness rating for a biological system at a certain time?
CQ3	Which was the patient's COMORBIDITY INDEX (CIRS-CI) at a certain time?
CQ4	Which was the patient's ILLNESS SEVERITY SCORE (CIRS-IS) at a certain time?

Table 3.9: Competency questions answered through the CIRS ontology.

ure 3.13. Refer to the OWL file for the CQs. The namespace prefix `mna` is associated with the value `http://www.ontologydesignpatterns.org/ont/mario/mna.owl#`. The result of a Mini Nutritional Assessment is represented by `mna:MiniNutritionalAssessment` that associates a certain Patient with the score (i.e. `cga:score`) of the assessment and the resulting Malnutrition Indicator Score (represented by `mna:MNARating`). An `mna:MNARating` has a `generic:name` (e.g. *well-nourished*) and a quantitative value indicating the range of MNA score it refers to (e.g. a Patient is considered well-nourished if her score in the MNA is greater than 24). A

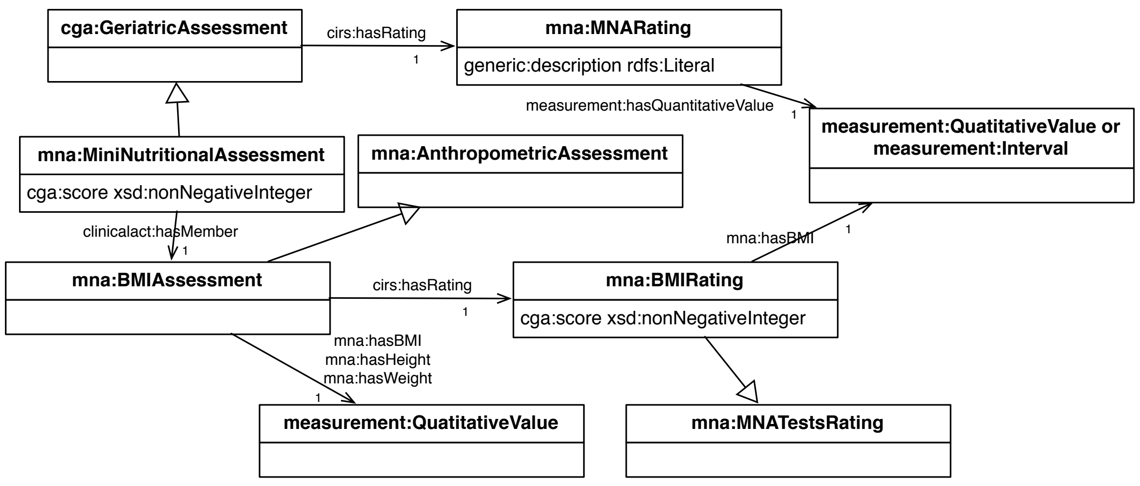

cga:GeriatricAssessment
cirs:hasRating
1
mna:MNARating
generic:description rdfs:Literal
measurement:hasQuantitativeValue
1
measurement:QuatitativeValue or measurement:Interval
mna:MiniNutritionalAssessment
cga:score xsd:nonNegativeInteger
mna:AnthropometricAssessment
clinicalact:hasMember
1
mna:hasBMI
1
mna:BMIAssessment
cirs:hasRating
1
mna:BMIRating
cga:score xsd:nonNegativeInteger
mna:hasBMI
mna:hasHeight
mna:hasWeight
1
measurement:QuatitativeValue
mna:MNATestsRating

Figure 3.13: The UML class diagram of the MNA ontology.

`mna:MiniNutritionalAssessment` is made on the basis of other four assessments: the `mna:AnthropometricAssessment`, the `mna:GeneralAssessment`, the `mna:DietaryAssessment` and the `mna:SelfAssessment`. Each of these assessment is made evaluating other assessments. For instance, the `mna:AnthropometricAssessment` is made on the basis of the `mna:BMIAsssessment`, the `mna:CalfCircumferenceAssessment`, the `mna:MACAssessment` and the `mna:WeightLossAssessment`. Each assessment defines its rating scale, e.g. the rating scale of `mna:BMIAsssessment` is `mna:BMIRating`.

3.3.3 Tagging Ontology

Tagging has become a key feature of the todays social media. A tag is a label (precisely, a free-word keyword) that is attached to someone or something for many purposes: identifying, categorizing, commenting, voting, reacting etc.

The aim of the tagging ontology is to represent a tagging action, i.e the action performed by an agent that attaches a label or something with a well-defined semantics (eg. a concept or a frame etc.) to some entity. In the context of the MARIO project tags will be associated with multimedia contents (e.g. photos, videos, audios and so on.), events (e.g. festivals, birthdays, daily events etc.) or personal memories

CQ1	*Which is the tag associated with a certain photo or events?*
CQ2	*Who has given a tag to a certain entity?*
CQ3	*Which is the entity associated to a certain tag?*

Table 3.10: Competency questions answered through the Tagging ontology.

(which can be considered as a particular kind of events in a person's life). Hence, the tagging ontology provides a reference model for representing tags in MARIO and can be used for interacting with patients in a variety of tasks that requires remembrances, e.g., stimulating the patient's memory, entertaining the patient with multimedia contents associated with particular moments of the her life, etc.

Related Work. Several ontologies[133, 115, 128] and Ontology Design Patterns[18] have been proposed to conceptualize the tagging action so far. [112] surveyed the state of the art in the tagging ontologies. In the most of them the tagging concept is represented as reified n-ary relationship between the tagger (the agent who gives the tag), the tag (often a keyword taken from a folksonomy), the entity tagged, and the date when the action happened. Some of them tried to associate the tag with its meaning [133], to express the polarity of the tagging [18] [115] or to attempt to treat tagging as a vote [19].

Module overview. The ontology we propose does not deviate from the state of the art significantly. However, our ontology allows MARIO to use not only simple free-word keywords, but also individuals with a well defined semantics or even complex named graphs. More in general, the designed ontology allows to answer the competency questions of the table 3.10.

The figure 3.14 shows the ontology diagram. The class `tagging:Tagging`[20] is a reification of the relationship between the agent who made the tag, the tag itself and

[18]Aldo Gangemi, http://ontologydesignpatterns.org/cp/owl/tagging.owl

[19]http://info.slis.indiana.edu/ dingying/uto.owl

[20]The prefix `tagging` is associated with the namespace http://www.ontologydesignpatterns.org/ont/mario/tagging.owl. Refer to Table 3.11 for the namespaces of the imported ontology modules.

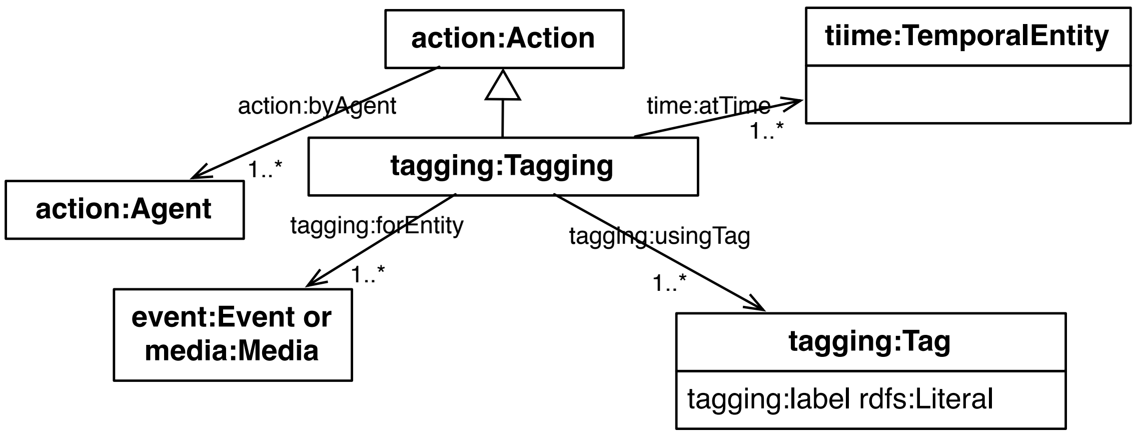


Figure 3.14: The UML class diagram of tagging ontology.

the entity tagged. It can be also viewed as an action whose agent is the tagger and the patient is the entity tagged. An individual of `tagging:Tagging` has to define *at least* an agent that performs the action (`action:byAgent` property), an entity target for tagging action (`action:forEntity` property), the tag used (`tagging:usingTag` property) and the date time when this action is performed (`time:atTime` property).

The class `tagging:Tag` represents any object that can be used to identify, categorize, describe or comment the entity being tagged. Being object of a the predicate `tagging:hasTag` implies to be a Tag, therefore a richer description such as a frame, a named graph, or a FRED graph can be used as tag for an entity. Furthermore, the datatype property "`rdfs:label`" can be used to associate an individual of Tag with the text it represents.

Currently, an individual of `tagging:Tagging` can be connected either to an individual of the class `event:Event` or to an individual of the class `media:Media` (such as photos, videos, audios etc.). A multimedia content (i.e., an individual of the class `media:Media`) can be associated with an event by the object property `event:hasEvent`, e.g. a video taken at a birthday party. Finally, we defined a property chain to ensure that the tags of an event are inherited by all multimedia content connected to it, e.g. if the birthday party has the tag "Birthday", then the video associated to it inherits the tag "Birthday".

Namespace prefix	**Namespace**
`action`	http://www.ontologydesignpatterns.org/ont/mario/action.owl#
`time`	http://www.ontologydesignpatterns.org/ont/mario/time.owl#
`event`	http://www.ontologydesignpatterns.org/ont/mario/event.owl#
`media`	http://www.ontologydesignpatterns.org/ont/mario/media.owl#

Table 3.11: Ontology modules imported/reused by the Tagging ontology.

```
:January_20_1971 a time:TemporalEntity .
:John_51st_birthday a event:Event ;
  spatial:hasPlace :Piper_Club ;
  time:atTime :January_20_1971 .
:John_picture a media:Image .
:Piper_Club a spatial:SpatialThing .
:tag_John_51st_birthday a tagging:Tag .
:tagging_John_51st_birthday a tagging:Tagging ;
  tagging:forEntity :John_51st_birthday , :John_picture;
  tagging:usingTag :tag_John_51st_birthday .
```

Frame 3.3: An example of usage of the tagging ontology serialized in TURTLE language

Usage Scenario MARIO might use the tagging ontology for annotating a picture about the birthday of his patient John. This picture was takes at John's 51st birthday on January 20 1971. Thus, the tagging ontology can be used for tagging the picture with this knowledge. Frame 3.3 shows the RDF graph resulting from the usage of tagging ontology for the previous example.

3.3.4 Other Modules

This Section briefly overviews the remaining modules of the ontology network. These modules have been derived from existing ontologies or well-known ontology design patterns. These ontologies have been adapted in order to fulfill the requirements of the case study of this thesis.

Action. The MON's Action module[21] is meant to implement the Task Execution Ontology Design Pattern[22]. This module allows to keep track of the actions performed either by the robot or by its users. An example of action performed by the robot is the assessment of the patient Freddy using the CGA framework. Actions execute and are classified by tasks. An example of task is assessing a patient through the CGA. The class `action:Task` aims at collecting all the *behaviors* of the robot. The Action module also defines a class named `action:App` which specializes Task and encloses all the apps made available by the robot.

Activity. The MON's Activity module[23] is a specialization of the Action module aimed at tracing activity performed by people. A person's activity usually involve many actions. For example, an activity of type `listeningtomusic:ListeningToMusic` may includes the actions of type `listeningtomusic:ListeningToSong` or `listeningtomusic:SkipSong`.

Calling, Chatting and Playing. The MON's Calling[24], Chatting[25], and Playing[26] module specialize the Activity module in order to provide a vocabulary to keep track of calling, chatting and playing activity of the users. The Calling ontology distinguishes `calling:VideoCalling` from `calling:VoiceCalling`. Video calling allows to trace the calling activity of a user using a service providing the functionality of video calling, whereas voice calls use services having voice calling functionality.

Clinical Act. The MON's Clinical Act module[27] defines a simple vocabulary that enable the robot to store the actions (i.e. `clinicalact:ClinicalAct`) performed by

[21]Action Module, http://ontologydesignpatterns.org/ont/mario/action.owl
[22]Task Execution,http://ontologydesignpatterns.org/wiki/Submissions:TaskExecution
[23]Activity Module, http://ontologydesignpatterns.org/ont/mario/activity.owl
[24]Calling Module, http://ontologydesignpatterns.org/ont/mario/calling.owl
[25]Chatting Module, http://ontologydesignpatterns.org/ont/mario/chatting.owl
[26]Playing Module, http://ontologydesignpatterns.org/ont/mario/playing.owl
[27]Clinical Act Module, http://www.ontologydesignpatterns.org/ont/mario/clinicalact.owl

physicians that involve a patient. These actions are described by the data property `generic:description`.

Drinking and Eating. The dietary assessment in elderly is an important task that avoid the risk malnutrition. Assistive social robots allow to continuously monitor the food and liquid intake of elderly. MON's Drinking[28] and Eating[29] modules enable robots to keep track of food and liquid intakes. In particular, `eating:Eating` and `drinking:Drinking` are two activities performed by a person that involve a as patients `eating:Course` and `drinking:Drink`, respectively. The classes `eating:Course` and `drinking:Drink` allow to specify the quantity and the substances the person intakes.

Emotional State. The MON's Emotional State module[30] allows the robot to trace the emotional state of a person over time. An emotional state is a time-indexed situation defined as a ternary relation connecting *(i)* the person who expresses the emotion, *(ii)* the emotion expressed (i.e. an individual of type `emotionalstate:Emotion`), *(iii)* and, the time interval in which the person shows the emotion.

Event. The aim of the MON's Event module[31] is to enable the robot to trace the relevant events it is involved in. An `event:Event` relates a number of agents and entities involved in the event, the place where the event takes place and the time when the event occurs.

Generic. The MON's Generic module[32] provide the robot with the data properties for naming (i.e. `generic:name`) and describing (i.e. `generic:description`) things.

[28]Drinking Module http://www.ontologydesignpatterns.org/ont/mario/drinking.owl

[29]Eating Module, http://www.ontologydesignpatterns.org/ont/mario/eating.owl

[30]Emotional State Module, http://www.ontologydesignpatterns.org/ont/mario/emotionalstate.owl

[31]Event Module, http://www.ontologydesignpatterns.org/ont/mario/event.owl

[32]Generic Module, http://www.ontologydesignpatterns.org/ont/mario/generic.owl

Health Role. The Health Role module[33] provides a vocabulary that defines the main health professional roles involved in the assistive context. The health roles are defined as individuals of the class `healthrole:HealthRole`. The ontology also defines for each role a class aimed at enclosing the persons having the role. For example, for the health role `healthrole:PhysicianRole`, the ontology defines the class `healthrole:Physician` which is specified as the class of all the persons having `healthrole:PhysicianRole` as value for the property `healthrole:hasHealth Role`.

Language. The MON's Language module[34] allows to associate a text fragment with the language the text belongs to. It is a specialization of the more general Literal Reification Ontology Design Pattern[35]. A `language:Text` is an individual having a `language:content` and a language (through the object property `language: hasLanguage`). A language (i.e. `language:LinguisticSystem`) is defined as a system of "signs, symbols, sounds, gestures, or rules used in communication".

Multimedia Content. The MON's Multimedia Content module[36] enables the robot to create an archive of multimedia files. The main class of the ontology is `multimediacontent:Media` that subsumes other media types like `multimedia content:Image`, `multimediacontent:Video` and `multimediacontent:Audio`. A Media is characterized by a `multimediacontent:url`, a name, a description, a time in which it is taken, and (possibly) an event of the knowledge base which the media refers to.

[33]Health Role Module, http://www.ontologydesignpatterns.org/ont/mario/healthrole.owl

[34]Language Module, http://www.ontologydesignpatterns.org/ont/mario/language.owl

[35]Literal Reification ODP,http://ontologydesignpatterns.org/wiki/Submissions:Literal_Reification

[36]Multimedia Content Module, http://www.ontologydesignpatterns.org/ont/mario/multimediacontent.owl

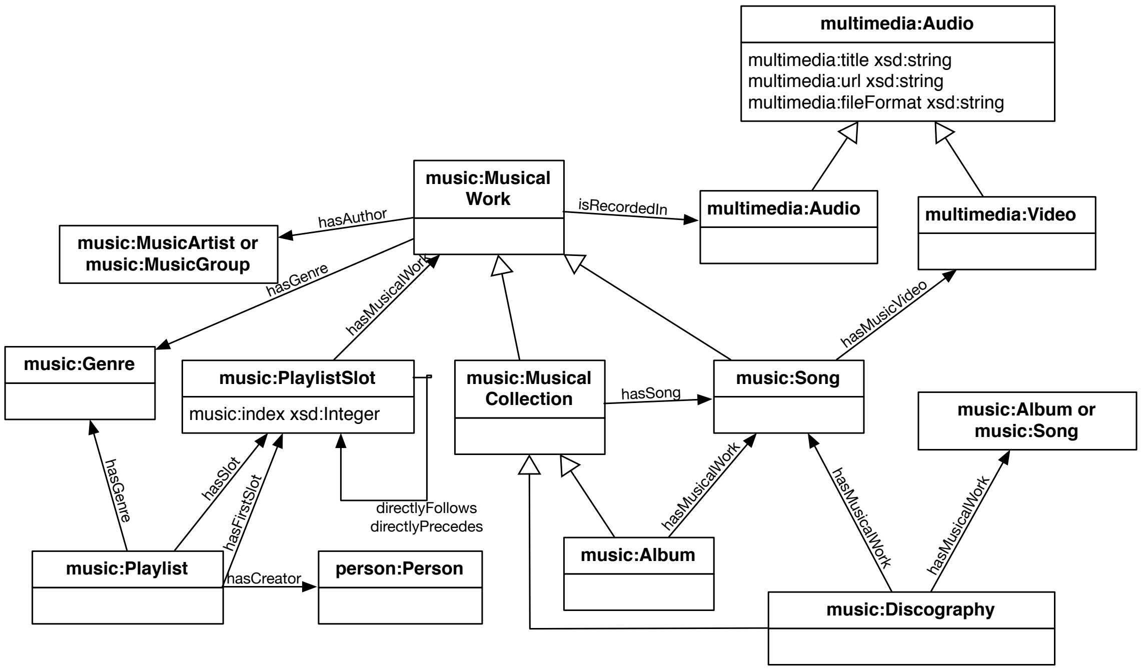


Figure 3.15: The UML class diagram of Music ontology.

Measurement. The MON's Measurement module[37] provides a vocabulary for expressing quantities and intervals. The module defines the class `measurement:QuantityValue` to denote quantities (e.g. 42) and `measurement:Interval` to represent intervals (e.g. from 3 to 5). Quantities and Intervals may have unit of measure (e.g. kilogram). The unit of measure is identified by the class `measurement:MeasurementUnit` which allows to specify the unit symbol through the data property `measurement:unitSymbol`.

Music. The MON's Music module[38] enables the robot to store information about the music it can reproduce. Figure 3.15 depicts the UML class diagram for the Music ontology module. The class `music:MusicalWork` is aimed at including all the musical works the robot is able to reproduce, namely songs (i.e. `music:Song`) and musical collections (i.e. `music:Album` and `music:Discography`). Musical works have an author (either a `music:MusicArtist` or a `music:MusicGroup`), a genre

[37]Measurement Module, http://www.ontologydesignpatterns.org/ont/mario/measurement.owl
[38]Music Module, http://www.ontologydesignpatterns.org/ont/mario/music.owl

(i.e. `music:Genre`), and are recorded in audio files. The ontology also allows to define playlists, that is, ordered collections of items (i.e. `music:PlaylistSlot`). An ordered collection is realized by applying the List Ontology Design Pattern[39]. The items of the list are the playlist slots. Each playlist slot is associated with the slot it directly precedes/follows and is connected to the musical work that has to be reproduced. All the slots are grouped by an individual of the class `music:Playlist` through the property `music:hasSlot`. The first slot of the playlist is identified using the `music:hasFirstSlot`. The playlist is also associated with the `music:Genre` it belongs to and the person who created the playlist.

Online Account. The MON's Online Account module[41] is aimed at storing the information about the account owned by the robot's user. An `onlineAccount: OnlineAccount` is are associated with the service that provides the account, and, with the unique identifier of the account. The object property `onlineAccount:has OnlineAccount` associates the person with the account s/he holds.

Person. The information related to persons are modeled following the MON's Person module[42]. A `person:Person` is defined as an agent having a gender (i.e. an individual of the class `person:Sex`), a residence (i.e. a `spatial:Residence`), a birth place (i.e. a `spatial:City`), and an hometown (i.e. a `spatial:City`). Persons are specialized in `person:Male` (defined as the persons having `person:Male` as value of the property `person:hasGender`[43]) and `person:Female` (defined as the persons having `person:Female` as value of the property `person:hasGender`). In this way, the ontology gives the opportunity of defining other genders. The ontology also defines a hierarchy of relations among persons[44]. The top property of this hierarchy

[39]List ODP,[40]

[41]Online Account Module, http://www.ontologydesignpatterns.org/ont/mario/onlineAccount.owl

[42]Person Module, http://www.ontologydesignpatterns.org/ont/mario/person.owl

[43]`person:Male` and `person:Female` are defined both as classes and as individuals of the class `person:Gender`.

[44]The hierarchy is inspired to http://vocab.org/relationship/.

is `person:hasRelationshipWith` that relates two persons having any kind of relationship. This property is specialized by forty sub-properties (e.g. `person:childOf`, `person:worksWith`). The domain and range of the sub-properties is also specialized where needed. For example the domain of `person:brotherOf` is `person:Male`, whereas the domain of `person:sisterOf` is `person:Female`. The ontology also defines proper inverse relation among the properties of the hierarchy. For example, `person:grandparentOf` is defined as the inverse of `person:grandchildOf`.

Personal Events. The MON's Personal Events module[45] provides a vocabulary for storing information about the main events of a person's life (e.g. Marriage, School attendance etc.). A `personalevents:PersonalEvent` is defined as an event related to a person. Following the Neo-Davidsonian[58] event semantics, this class subsumes six event types: *(i)* `personalevents:Employment` the fact of having a paid work; *(ii)* `personalevents:Marriage` the fact of being married with some one; *(iii)* `personalevents:SchoolAttendance` the fact of having attended a school; *(iv)* `personalevents:PetOwnership` the fact of owning a pet; *(v)* `personalevents:LivingInAPlace` the fact of living in a place; *(vi)* `personalevents:VisitingAPlace` the fact of visiting a place. These six event types support the application for delivering the reminiscence therapy presented in Section 7.4.1.2. Additional event types can be also gathered from Framester's Frames (cf. Section 4.1). In fact, a frame is a stereotyped situation that could used to model information about events of a person's life.

Pet. The MON's Pet module[46] is aimed at modeling all the information about pets owned by the robot's users. A `pet:Pet` is a defined as an agent having a certain `pet:PetType`.

[45]Personal Events Module, http://www.ontologydesignpatterns.org/ont/mario/personalevents.owl

[46]Pet Module, http://www.ontologydesignpatterns.org/ont/mario/pet.owl

Postal Address. The MON's Postal Address module[47] provides a model for the addresses that is compliant with the specifications of the EU INSPIRE directive [48].

Question Answer. The MON's Question Answer module[49] allows to specify a set of questions and tracing what someone says in reaction to the question. A `question answer:Question` is (a reification of) a sentence in interrogative form, whereas a `questionanswer:Answer` represents the reaction of a Person to a question. The relation between answer and question can be specified with the object property `questionanswer:toQuestion`. Both questions and answers have a textual content that can be specified through the data property `questionanswer:textualContent`.

Service. The MON's Service module[50] defines a vocabulary for specifying the services (e.g. on line messaging services or social networks) users have an account on. A `service:Service` is defined as "an intangible commodity", and it is specialized by the class `service:OnlineService` which represents the services available on the web. The ontology allows to specify for each service the functionalities it provides (e.g. voice calling, video calling etc.).

Spatial. The MON's Spatial module[51] provides classes and properties for specifying the knowledge about spatial things (i.e. anything with a spatial extent) and places (e.g. cities). These concepts are used for specifying the generic locations for physical objects. This ontology module is an implementation of the Ontology Design Pattern Place[52].

[47]Postal Address Module, http://www.ontologydesignpatterns.org/ont/mario/postalAddress.owl

[48]https://inspire.ec.europa.eu/Data-Models/Data-Specifications/2892

[49]Question Answer Module, http://www.ontologydesignpatterns.org/ont/mario/questionanswer.owl

[50]Service Module, http://www.ontologydesignpatterns.org/ont/mario/service.owl

[51]Spatial Module, http://www.ontologydesignpatterns.org/ont/mario/service.owl

[52]Place Ontology Design Pattern http://ontologydesignpatterns.org/wiki/Submissions:Place

Time. The MON's Time module[53] defines a vocabulary for specifying time-related entities (such as instants and intervals) and for relating entities to time. This ontology has been derived from the W3C's Time ontology[54]. As in the W3C's ontology time related entities are subsumed by the class `time:TemporalEntity`. This class subsumes `time:Interval` which, in turn, (differently from the W3C's ontology) it subsumes `time:Instant`. In fact, for the continuous nature of time and its infinite divisibility, an instant it can be seen as a very short interval. Within the context of knowledge base systems, an instant is usually defined as the unit of the shortest granularity available on the system. The MON's Time module allows to define an instant as an `xsd:datetime` which gives to instants the granularity of milliseconds.

Time-indexed Situation. The MON's Time-indexed Situation module[55] allows to represent situations that are explicitly indexed at some time and that involve an entity. This ontology module implements the homonymous Ontology Design Pattern[56].

Time-indexed Relationship. The MON's Time-indexed Relationship module[57] allows to specify the fact that two persons have a relationship for a certain time. A `timeindexedrelationship:TimeIndexedRelationship` is a time-indexed situation involving at least two persons, an entity of type `timeindexedrelationship:Relationship`, and a temporal entity which indicates the beginning and the end of the relationship of the two persons.

[53]Time Module, http://www.ontologydesignpatterns.org/ont/mario/time.owl

[54]W3C's Time ontology, https://www.w3.org/TR/owl-time/

[55]Time-indexed Situation Module, http://www.ontologydesignpatterns.org/ont/mario/timeindexedrelationship.owl

[56]Time-indexed Situation Ontology Desing Pattern, http://ontologydesignpatterns.org/wiki/Submissions:TimeIndexedSituation

[57]Time-indexed Relationship Module, http://www.ontologydesignpatterns.org/ont/mario/timeindexedrelationship.owl

Vital Signs. The MON's Vital Signs module[58] allows to keep track of the vital signs of a patient over time. A `vitalsigns:VitalSignsMeasurementInTime` is a time-indexed situation representing a measurement (specified with a `measurement:QuantitativeValue`) of a certain vital sign (i.e. `vitalsigns:VitalSign`) at a certain time for a certain patient.

3.4 Discussion

The study reported in this chapter aimed at determining what kind of knowledge social robots need in order to operate in socially assistive context (cf. RQ1, Section 1.1). To this end, the chapter presented a proof-of-concept, called MARIO Ontology Network (MON). MON defines a set of interconnected and modularized ontologies for representing and structuring the knowledge processed by a social robot in socially assistive context. The ontology modules were identified by analyzing and generalizing the use cases that emerged from the MARIO project (cf. Section 1.3).

The most developed knowledge areas are those related to the medicine domain, social and multimedia contents, and user-related information (user data, user's life events etc.). The competency questions falling in the area of multimedia contents, user related information and transversal knowledge (e.g. temporal and spatial information) were mostly addressed by reusing and adapting existing ontologies. Instead, considerable effort was spent to meet the requirements related to the medicine domain. In particular, there were no ontologies able to support a robot in performing a Comprehensive Geriatric Assessment of its patient. MON filled this gap with the CGA Ontology (cf. Section 3.3.2) which enabled the robot to autonomously perform a Comprehensive Geriatric Assessment (cf. Section 7.4.1.1).

Another innovative module of MON is the Affordance ontology (cf. Section 3.3.1) which enables a novel mechanism for arbitrating robot's actions. This model allows to define a series of situations a robot should react to and the most appropriate actions to perform in each situation. In this way the choice of the action to perform

[58]Vital Signs, http://www.ontologydesignpatterns.org/ont/mario/vitalsigns.owl

is faced at the knowledge level [151] and the robot is able to take the decision with all the knowledge it has. This ontology provides a simple and effective strategy for designing robot's personality and social rules it must follow in any situation. However, this ontology provides only an high level model for supporting this mechanism and considerable work needs to be done in order to facilitate the definition of the affordance relation. This could be done by defining a vocabulary of common situations social robots should reacts to. This gap is partially fulfilled by projects like FrameNet [19] which provides a repository of stereotyped situations (called frames). Nevertheless, FrameNet's frames are too generic to be used for this mechanism and need specialization in the social robotics domain.

Concerning the "sociality" of a robot, much work still to be done to investigate to what extent the robot's knowledge impacts on acceptability, empathy and trustability of robots. Although Social Ontology [188] (i.e. the study of the nature and properties of the social world) is a developed field in philosophy, the results of this research area are closed off to robots. Therefore, future work should focus on encoding social theories in a machine-interpretable format.

Chapter 4

Providing Linked Open Data as Background Knowledge for Social Robots

While Chapter 3 focused on defining the intensional level of the robot's knowledge, this Chapter investigates the possibility of populating the extensional level with data retrieved from the web (cf. Section 1.1, RQ2). To this end two lines of research have been carried out in parallel focusing on linguistic and common sense knowledge respectively. Regarding the first line of research this chapter presents Framester, a huge linguistic knowledge graph integrating lexical, linguistic, ontological and encyclopedic data. The method for building Framester is described in Section 4.1. This Chapter also introduces a novel empirical method for assessing foundational distinctions over Linked Open Data entities from a common sense perspective (e.g. deciding if an entity inherently represents a class or an instance from a common sense perspective). This method realizes the first step of a more general procedure meant to automatically generate common sense knowledge from Linked Open Data. Section 4.2 overviews methods and experiments aimed at assessing to what extent Linked Open Data could be a source of common sense knowledge.

4.1 Framester: a Linguistic Data Hub

When dealing with robot understanding, we need to integrate knowledge about the robot's physical context, linguistic knowledge, and knowledge about the world in general. Designing a knowledge base for a socially assistive robot can be therefore considered a direct application of the Linked Data paradigm, which provides interoperability across existing Linked Open Data (world's background knowledge), linguistic knowledge, user's knowledge and robot's sensor knowledge. In order to create an adequate amount of background knowledge, and to make it accessible to the robot, Framester[1] [81], a large "cloud" of linguistic and factual data has been created which stands on the shoulders of a flexible and cognitively-sound theory of human sense making, Frame Semantics [75].

Framester is intended to work as a knowledge graph/linked data hub to connect lexical resources, NLP results, linked data, and ontologies. It is bootstrapped from existing resources, notably the RDF versions of FrameNet [153], WordNet [145], VerbNet [187], and BabelNet [150], by interpreting their semantics as a subset of (a formal version of) Fillmore's frame semantics [75], and semiotics [80], and by reusing or linking to off-the-shelf ontological resources including OntoWordNet [85], DOLCE [138] and DOLCE-Zero [86], Yago [201], DBpedia [31].

4.1.1 Framester Overview

Framester uses the D&S (i.e. Descriptions and Situations) knowledge pattern [84], which allows to distinguish the reification of the intension of a predicate (a *description*) from the reification of the extensional denotation of a predicate (a *situation*). A description d can define or reuse *concepts* $c_1 ... c_n$ that can be used to *classify* entities $e_1 ... e_m$ involved in a situation s that is expected to be compatible with d. For those reasons, D&S perfectly fits the core assumptions of Fillmore's frame semantics, by which a frame is a schema for conceptualising the interpretation of a natural language text (and beyond), its denotation (a frame occurrence) is a situation, and the elements (or semantic roles) of a frame are aspects of a frame, which can be either obligatory, optional, inherited, reused, etc. Furthermore, D&S takes into account a semiotic theory to integrate linguistic and formal semantics. It can therefore support additional frame semantics assumptions such as *evocation* and *semantic typing*.

[1]Framester, https://w3id.org/framester

As described in [153, 184], several recipes can be designed to interpret FrameNet frames and frame elements as OWL classes, object properties, or punned individuals. Both FrameBase and Framester make use of the basic recipe that interprets frames as classes and frame elements as properties. However, Framester goes deeper in providing a two-layered (intensional-extensional) semantics for frames, semantic roles, semantic types, selectional restrictions, and the other creatures that populate the world of lexical resources. The two-layered representation is based on the Descriptions and Situations pattern framework [84], and exploits OWL2 punning, so enabling both (intensional) navigation in the linked lexical datasets, and the reuse of lexical predicates as extensional classes or properties. The Framester's schema is included in the MARIO Ontology Network and it is available online at[2]. The main assumptions for Framester knowledge graphs are as follows:

1. A frame is a *multigrade intensional predicate* [156] $f(e, x_1, ..., x_n)$, where f is a first-order relation, e is a (Neo-Davidsonian) variable for any *eventuality* or *state of affairs* described by the frame, and x_i is a variable for any *argument place*, which could admit several *positions* in case multiple entities are expected to be classified in a place.

2. OWL2 punning allows to represent a frame as either a class $f \sqsubseteq$ `framester:Situation` (a subclass of the `framester:Situation` class, having situations as instances) or as an individual $f \in$ `framester:Frame` (an instance of the `framester:Frame` class).

3. Any word or multiword can evoke a frame: this is represented by means of a property chain that connects a word entity to a (punned) frame.

4. *Frame Projections* include any projections of a frame relation. Assuming frame semantics, each meaning consists of activated frames, whose formal counterparts are multigrade intensional predicates. When only some aspect of that frame is considered, it can be formalized as a (typically unary or binary) pro-

[2]Framester's schema, https://w3id.org/framester/schema

jection of a frame relation. Semantic role as well as co-participation relation are binary projections of a frame.

5. A *frame occurrence* (a situation denoted by text or data) $s \in f$ is an instance of f and the entities $\{e, x_1...x_n\}$ involved in a situation are individuals.

Due to the expressivity limitations of OWL, some refactoring was needed to represent frame semantics: frames are represented as both classes and individuals, semantic roles and co-participation relations as both (object or datatype) properties and individuals, selectional restrictions and semantic types as both classes and individuals, situations and their entities as individuals. Frames and other predicates are represented as individuals when a schema-level relation is needed (e.g. between a frame and its roles, or between two frames), which cannot be represented by means of an OWL schema axiom (e.g. subclass, subproperty, domain, range, etc.).

Example. Figure 4.1 shows how the D&S pattern framework (descriptions, situations, classification patterns) is at the basis of Framester representation of the example predicate `G_suit`. Related notions from WordNet (`wn:`), BabelNet (`bn:`), FrameNet (`fe:`), DBpedia (`dbr:`), and DOLCE-Zero (`dul:`) make linguistic and factual data linked by using Framester ontology (`fschema:`) and data (`framester:`), and OWL logic (`owl:`, `rdfs:`). In practice, this automated integration allows to represent any data coming from different resources (i.e. not only those depicted, but all those that are associated with them in the Linked Open Data cloud) in a homogeneous and logically rigorous way. That would include instances of G-suits, knowledge about G-suits, places where G-suits are produced or used, the frames (e.g. Clothing) that include G_suit as a participant, multilingual versions of the predicates and entities associated with G_suits, etc.

4.1.2 Semi-automatic Generation of Framester

The process for generating Framester includes several steps including conversion of linguistic and factual resources in RDF, automatic generation and manual refinement

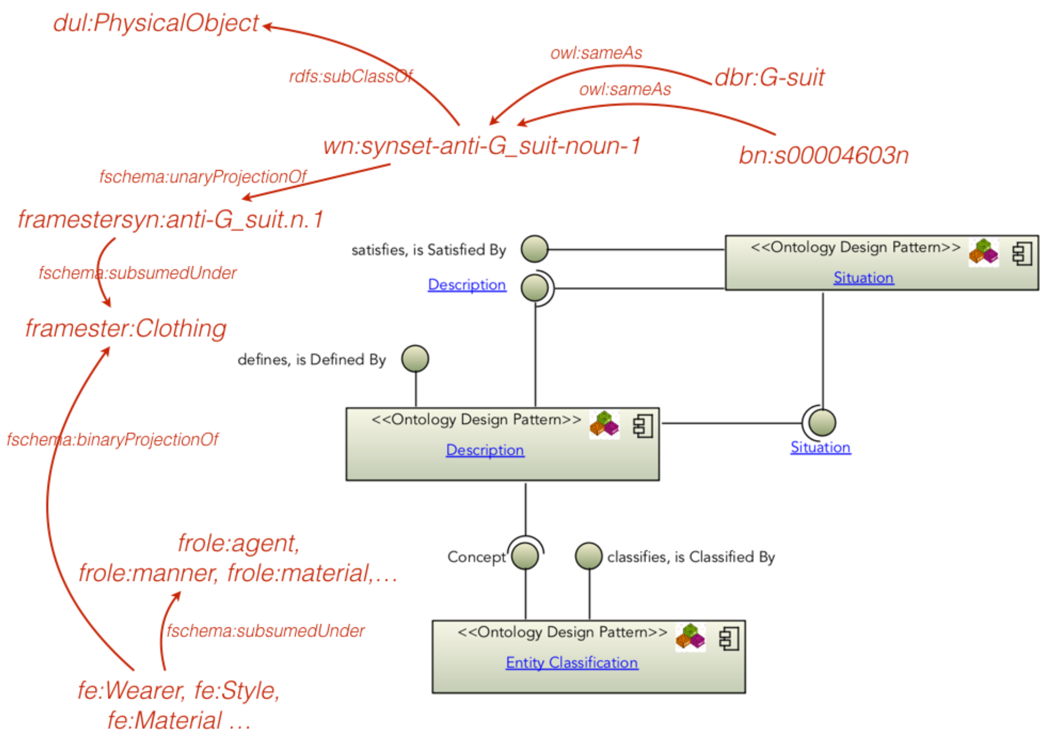

dul:PhysicalObject
rdfs:subClassOf
owl:sameAs
owl:sameAs
dbr:G-suit
wn:synset-anti-G_suit-noun-1
bn:s00004603n
fschema:unaryProjectionOf
framestersyn:anti-G_suit.n.1
fschema:subsumedUnder
framester:Clothing
fschema:binaryProjectionOf
frole:agent, frole:manner, frole:material,…
fschema:subsumedUnder
fe:Wearer, fe:Style, fe:Material …
satisfies, is Satisfied By
Description
<<Ontology Design Pattern>>
Situation
defines, is Defined By
<<Ontology Design Pattern>>
Description
Situation
Concept
classifies, is Classified By
<<Ontology Design Pattern>>
Entity Classification

Figure 4.1: An example of Framester representation for the concept `G_suit`.

of mapping among RDF individuals and assessment of integrity constraints. This process guarantees the sustainability of Framester over time and follows the Extract, Transform and Load (ETL) pattern.

4.1.2.1 Conversion of Input Resources in RDF

The input of the process is a collection of linguistic, factual, encyclopedic and ontological resources. Most of the factual, encyclopedic and ontological resources (such as DBpedia, YAGO, DOLCE etc.) are already available in RDF format, but this is not the case for most of the linguistic resources (such as FrameNet, WordNet or VerbNet) which are often distributed in non-standard format. In the last two decades several projects (among others [16, 153, 65, 142, 184, 52]) aimed at publishing particular versions of these datasets following the Linked Open Data principles. As a result most of these datasets are now available as Linked Open Data (cf. Section 2.6). Whenever a dataset (or one of its version) is not already available in Linked Open Data, one of the following conditions holds. *(i)* The resource is distributed in a standard syntactic format (e.g. XML or CSV), then, it can be converted in RDF by using general purpose triplification tools (such as Semion [154], D2RQ [30] or Any23[3]). The schema of the generated RDF dataset is autonomously learned from the input dataset and could be refined by defining some hand-crafted rules. *(ii)* For particular datasets (e.g. FrameNet, WordNet) ad-hoc tools have been designed for managing the transformation to RDF (e.g. WN-RDF[4] [142] or Premon [52]). These tools guarantee that future versions of the dataset can be easily converted to RDF (unless substantial changes of the format of resource occur). *(iii)* For the remaining datasets a tool for performing the conversion to RDF needs to be developed. For example, this is the case of the Paraphrase Database (PPDB) [163] which is distributed in non-standard tabular format. In order to convert PPDB in RDF we have developed PPDB2RDF, a tool, available online at[5], which converts the PPDB dataset in RDF following the schema defined in the PPDB ontology[6].

4.1.2.2 Normalization of the Input Resources

The second step of the process is meant to normalize the input datesets in order to be compliant with the Framester ontology, existing rules and alignments. For each imported dataset the Framester's ontology provides a reference schema resulting from existing work. For example, the reference schema for WordNet is defined by OntoWordNet [85], the FrameNet's reference schema has been proposed in [153]. These reference schemas are also used to define integration and integrity rules. As we will see later, to integrate the input datasets and to assessing the integrity of the data, the process uses a set handcrafted rules and alignments among individuals belonging to different resources (e.g. mapping between WordNet synsets and DOLCE

[3]Any23, https://any23.apache.org/

[4]WN-RDF, https://github.com/jmccrae/wn-rdf

[5]PPDB2RDF, https://github.com/luigi-asprino/ppdb2rdf

[6]PPDB ontology, https://w3id.org/ppdb/ontology

classes provided by OntoWordNet [85]). Developing rules and alignments is a costly activity, and, therefore it is desirable to reuse them as much as possible.

The normalization step involves both T-box and A-box of the new version of the resource. The A-box resulting from the conversion step is transformed to make it compliant to the T-box of the former version. For example, frames (i.e. the FrameNet's A-box) defined in FrameNet 1.7 are refactored following the ontology defined for FrameNet [153]. In this way, rules (involving onlu the T-box) defined for the former versions (i.e. FrameNet 1.5) can be applied on the new one (i.e FrameNet 1.7). The other normalization action involves the entities within the A-box. If the semantics of the entities has not changed, then a mapping between the entities of the new version and the corresponding entities of the former one can be created. For example, the synsets defined in WordNet 3.1 are mapped to the synsets with the same semantics belonging to the former versions. This kind of mapping is often provided by the resource creator (e.g. FrameNet provides for each release the differences with the previous one). The mapping among entities of the various versions guarantees the validity of the handcrafted alignments.

The variation of the semantics of an entity across two versions of a resource is called semantic drift. The process is not able to automatically detect a semantic drift, therefore, it could be only managed when the resource creator provides information about the entities that drifted their semantics. When this happens it is possible to tracing the drift through by generating an individual for each meaning of the entity, and, then connecting these individuals through an n-ary relation which indicates the contextual information related to the drift (e.g. the version of the resources where the entities come from).

4.1.2.3 Linking Entities of Different Resources

At the end of the normalization step all the input datasets are converted in RDF, then, a series of actions can be undertaken to link the entities of these imported datasets. Some of links among the entities already exist. BabelNet synsets are linked to WordNet synsets and DBpedia indivduals. Classes defined in YAGO classify DBPedia individuals and are aligned with WordNet's synsets. OntoWordNet connects WordNet synsets to DOLCE classes. Predicate Matrix [116] integrates WordNet, FrameNet, VerbNet and PropBank. Several projects aimed at aligning FrameNet's frame with WordNet's synsets (among others eXtended WordFrameNet [117] and FrameBase [184]). PPDB has been used to increase the linguistic coverage of FrameNet [163].

These alignments are mostly automatically generated and are used for bootstrapping the linking among the imported datasets. The process also allows humans to intervene for refining these alignments. This cleansing activity can be supported by providing humans with additional information on the entities involved in the alignment. This information can be retrieved from the imported datasets (e.g. descriptions of the entities). As a concrete example of this kind of activity we describe the cleansing of the mapping between WordNet's synsets and FrameNet's frames.

WordNet's synsets are equivalence classes of word senses whose words can evoke one or more frames. Alignments between WordNet's synsets and corresponding evoked FrameNet's frames are provided by eXtended WordFramenet [117], Predicate Matrix [116], FrameBase [184], and, more recently BabelNet [150] started publishing the links between its synsets and FrameNet's frames. The assessment of a relation WordNet's synset s and a FrameNet's frame f can be supported by the following information: *(i)* Informal description of the semantics of s and f (i.e. glosses); *(ii)* Hyponyms and hypernyms of s with their glosses, and, frames subsumed by f and frames from which f inherits with their glosses; *(iii)* Semantic Roles (i.e. Frame Elements) and Semantic Types of f with their glosses; *(iv)* The DOLCE class that OntoWordNet associates with the synset s; *(v)* A set of sentences from WordNet containing a word annotated with the synset s, and, a set of sentences from FrameNet containing a word annotated with the frame f; *(vi)* Alternative synsets (i.e. WordNet synsets containing at least a lexical unit of f) and alternative frames (i.e. FrameNet's frames having a lexical unit contained in the s). This information helps a human in understanding the semantics of entities and shows some sentences exemplifying the use of these entities in natural language. This information can be retrieved from the imported datasets by firing some queries. Providing this information in single graphical user interface could facilitate the cleansing activities and relieve the human from the burden of submitting several queries.

A deep manual revision of the mapping between FrameNet's frames and WordNet's synset is fundamental for the Framester generation process. Similar revisions could be carried out for other kind of mappings such as the mapping between YAGO's classes and WordNet's synsets. These activities would require considerable human effort and therefore crowdsourcing this task will be considered as a possible solution for refining other mappings.

Novel Mappings. Novel mappings among the imported datasets have been also created with a two-step approach. The first step is aimed at heuristically creating a set of candidate links which are manually curated in a second step. The revision is similar to the activity carried out for the existing mappings (as described above). An example is the mapping between DBpedia classes and WordNet's synsets. Candidate synsets for mapping a DBpedia class are the synsets containing a word that matches the class name. The refinement of this mapping is supported by providing a human with: Hypernyms/Hyponyms of the synset, instances of the DBpedia class, and DOLCE classes associated with the synsets in [85, 82] and with DBpedia classes in [162]. A similar procedure has been carried out for mapping FrameNet's Semantic Types on WordNet's synsets.

4.1.2.4 Heuristic Methods for Extending the Mapping

Manually curated mapping and links resulting from existing works (such as OntoWordNet [85, 82], YAGO [201], BabelNet [150] and more recently the work in [162]) constitute the base of the Framester's linking structure. Additional links are generated by applying rules and heuristics.

Extending WordNet and FrameNet mapping. Further extensions of the WordNet-FrameNet mapping were automatically performed based on: *(i)* WordNet hyponymy relations between noun and verb synsets, where each frame is extended with direct hyponyms of the noun or verb synsets already mapped to frames; *(ii)* "Instance-of" relations between WordNet noun synsets; *(iii)* Adjective synset similarity; *(iv)* Same verb groups including verb synsets; *(v)* Pertainymy relations between adverb synsets and noun or adjective synsets; *(vi)* Participle relations between adjective and verb synsets; *(vii)* Morphosemantic links between adjective and verb synsets; *(viii)* Transitive WordNet hyponymy relations; *(ix)* Unmapped siblings of mapped noun or verb synsets; *(x)* Derivational links between different kinds of synsets.

Mapping DBpedia Entities on Frames. Following [85, 82], Framester interprets WordNet synsets as classes and unary projections of frames. Therefore, an individual belonging to a WordNet synset is either an occurrence or a participant of a certain frame. In both cases the occurrence of the individual within an sentence activates the frame. For example, "The Mayweather-McGregor boxing match"[7] is an event belonging to the synset "boxing match" (i.e. `wn31:07481100-n`[8]), and, it is an occurrence of the synset frame "boxing match" (since the synset is one of the frame's unary projections). Floyd Mayweather Jr.[9] is a boxer (and belongs to the synset `wn31:09889614-n`[10]) which is a participant (i.e. a Semantic Type) of the frame "Boxing match". The occurrence of Floyd Mayweather also evokes, in a *weaker* way than its match with Mc Gregor, the frame "boxing match". For expliciting the fact that an individual activates a frame we defined the object property `fschema:playsRoleIn` which connects an individual to the frame it can activate. The alignment between classes of the DBpedia Ontology and WordNet's synset are then used compute the `fschema:playsRoleIn` relation: DBPedia entities belonging to a class mapped on a WordSynset s_1 are aligned with the synset frame having s_1 as unary projection.

[7]https://en.wikipedia.org/wiki/Floyd_Mayweather_Jr._vs._Conor_McGregor
[8]http://wordnet-rdf.princeton.edu/id/07481100-n
[9]https://en.wikipedia.org/wiki/Floyd_Mayweather_Jr.
[10]http://wordnet-rdf.princeton.edu/id/09889614-n

```
ASK {
  ?i ex:subsumedBy+ ?i
}

SELECT ?i ?i1 WHERE {
 ?i ex:subsumedBy+ ?i1 .
?i1 ex:subsumedBy+ ?i .
}
```

Frame 4.1: The SPARQL queries used for detecting a cycle within a hierarchy.

4.1.2.5 Assessing Integrity Constraints

A list of integrity rules have been developed in order to assess the consistency of the dataset and to detect anomalies. Anomalies could have been generated during the execution of an automatized task of the process or could come from the imported datasets. The integrity rules are implemented as SPARQL queries: ASK queries are used to check the presence of anomalies, and SELECT queries are used to retrieve entities and relations involved in the anomalies. Following a recent W3C's recommendation, we plan to re-implement there rules using SHACL language[11]. When an anomaly is detected it can be addressed with human intervention. Similarly to the cleansing activities discussed in Section 4.1.2.3, a human can be assisted in this task with the provision of side information retrieved from the dataset. For example, rules can defined to avoid cycles within hierarchies. The queries showed in Frame 4.1 can be used to detect a cycle in a hierarchy defined using `ex:subsumedBy` as subsumption relation. The ASK query checks if there is an individual ?i that transitively subsumes from itself, and the SELECT query returns all the edges involved in the cycle. A manual inspection of these edges is needed to identify the incorrect edge(s) and to fix the hierarchy.

[11]W3C's Recommendation https://www.w3.org/TR/shacl/

4.2 Assessing Foundational Distinctions in Linked Open Data

The Web and its Semantic extension (i.e. Linked Open Data) contain open global-scale knowledge and make it available to potentially intelligent machines that want to benefit from it. Nevertheless, most of Linked Open Data lack ontological distinctions and have sparse axiomatisation. For example, distinctions such as whether an entity is *inherently* a class or an individual, or whether it is a physical object or not, are hardly expressed in the data, although they have been largely studied and formalised by foundational ontologies (e.g. DOLCE, SUMO). These distinctions belong to common sense too, which is relevant for many artificial intelligence tasks such as natural language understanding, scene recognition, and the like. There is a gap between foundational ontologies, that often formalise or are inspired by pre-existing philosophical theories and are developed with a top-down approach, and Linked Open Data that mostly derive from existing databases or crowd-based effort (e.g. DBpedia, Wikidata). In this Section we investigate whether machines can learn foundational distinctions over Linked Open Data entities, and if they match common sense. We want to answer questions such as "does the DBpedia entity for *dog* refer to a class or to an instance?".

Common Sense Knowledge and Linked Open Data. Common Sense Knowledge is knowledge about the world, shared by all people. Common sense reasoning is also at the core of many Artificial Intelligence (AI) tasks such as natural language understanding, object and action recognition, and behavior arbitration [59], but it is difficult to teach to those systems. Its importance was assessed back in 1989 by [100] who argued that AI needs a "formalization of a sizable portion of commonsense knowledge about the everyday physical world" (cit.), which, he says, must have three main characteristics: uniformity, density, and breadth. The Semantic Web effort has partly addressed this requirement with Linked Open Data

(LOD): ∼200 billion linked facts[12], formally and uniformly represented in RDF and OWL, and openly available on the Web. Nevertheless, LOD still fails in addressing density (high ratio of facts about concepts) and breadth (large coverage of physical phenomena). In fact, it is very rich for domains such as geography, linguistics, life sciences and scholarly publications, as well as for cross-domain knowledge, but it mostly encodes this knowledge from an encyclopaedic perspective. The goal of the methodologies presented in this Section is to enrich LOD with common sense knowledge, going beyond the encyclopaedic view. We claim that an important gap to be filled towards this goal is: assessing foundational distinctions over LOD entities, that is to distinguish and formally assert whether a LOD entity inherently refers to e.g. a class or an individual, a physical object or not, a location, a social object, etc., from a common sense perspective.

Foundational Distinctions. We use DOLCE+DnS UltraLite (DUL)[13] as reference foundational ontology to select the targets of our experiments. We start by focusing on two very basic but diverse distinctions, which need to be addressed before approaching all the others: whether a LOD entity e.g. `dbr:Rome`[14], *(i)* inherently refers to a class or an instance, and whether it *(ii)* refers to a physical object or not. The first distinction (class vs. instance) is fundamental in formal ontology, as evidenced by upper-level ontologies (e.g. SUMO and DOLCE), and showed its practical importance in modelling and meta-modelling approaches in computer science, e.g. the class/classifier distinction in Meta Object Facility[15]. It is also at the basis of LOD knowledge representation formalisms (RDF and OWL) for supporting taxonomic reasoning (e.g. inheritance). Automatically learning whether a LOD entity is a class or an instance – from a common sense perspective – impacts

[12]https://lod-cloud.net/, accessed on April 5th 2019

[13]DOLCE+DnS UltraLite (DUL) http://www.ontologydesignpatterns.org/ont/dul/DUL.owl is derived from DOLCE. DUL inherits most of DOLCE's distinctions by providing a more intuitive terminology and simplified axiomatisation. It has been widely adopted by the Semantic Web community.

[14]`dbr:` stands for http://dbpedia.org/resource/

[15]https://www.omg.org/spec/MOF/

on the behaviour of practical applications relying on LOD as common sense background knowledge. Examples include: question answering, knowledge extraction, and more broadly human-machine interaction. In fact, many LOD datasets that are commonly used for supporting these tasks (especially general purpose datasets e.g. DBpedia, Wikidata, BabelNet) only partially, and often incorrectly, assert whether their entities are classes or instances, and this has been proved to be a source of many inconsistencies and error patterns [162]. It is worth noting that the notion of what is a class and what is an instance is highly context dependent for information systems' ontologies and therefore one might argue that this distinction is meaningless. However, we assume a commonsense perspective which is a broad although specified context and the high agreement in the results of experiments confirm that it is reasonable to target this context.

The second distinction (physical object or not) is essential to represent the physical world. In fact, only physical objects can move in space or be the subject of axioms expressing their expected (naive) physical behaviour (e.g. gravity). We refer to the definition of *Physical Object* provided by DUL: *"Any Object that has a proper space region. The prototypical physical object has also an associated mass, but the nature of its mass can greatly vary based on the epistemological status of the object (scientifically measured, subjectively possible, imaginary)"*.

In the reminder of this Section we describe an automated way of making these distinctions emerge empirically. In summary, we present:

- a novel method that leverages supervised machine learning and crowdsourcing to automatically assess foundational distinctions over LOD entities (cf. Section 4.2.2), according to common sense;
- four reusable datasets, based on a sample of DBpedia, separately annotated by experts and by the crowd with class/instance and physical object classification, for each entity (cf. Section 4.2.5). The crowdsourced task designs are on their turn reusable;

- a set of reproducible experiments targeting two foundational distinctions (*class vs. instance* and *physical object vs. not a physical object*) (cf. Section 4.2.6).

4.2.1 Related Work

Only a few studies focus on typing entities based on **foundational distinctions**. To the best of our knowledge, our research is the first to test the hypothesis that machines can learn foundational distinctions that match common sense, by using web resources. The closest work to ours in approach and scale is [86], which produced a dataset of DBpedia entities annotated with DUL classes, using ontology learning. We reuse this dataset and compare our results against it. The work by Miller and Hristea [146] addresses the problem of distinguishing classes from instances in WordNet [145] synsets, through purely manual annotation. This approach is inappropriate to test our research questions due to its lack of scalability. A work by Zirn et al. [216] faces the problem of distinguishing classes and instances within the Wikipedia Category Taxonomy. However, the proposed method is hardly generalizable for other foundational distinctions and for any LOD entities. This Chapter presents a methodology that can assess any foundational distinction on any LOD entity. A recent work [161] proposes a method for detecting classes among Wikipedia articles by using simple linguistic features and heuristics. Similar assumptions, like checking particular linguistic patterns and features, are investigated in this work.

4.2.2 Methods

We are interested in observing whether the Linked Open Data (LOD) and its related Web resources provide an empirical basis for making foundational distinctions over entities represented in the LOD according to the commonsense intuition. Our objective is to test all the distinctions formalised in DUL. Nevertheless, for each of them we need to create a set of reference datasets (cf. Section 4.2.5) in order to train and evaluate the proposed method, which requires a significant amount of work. For this reason, we start focusing on two distinctions: between class and instance, and between what is a physical object and what is not. These are two of the most basic, but very diverse distinctions in knowledge representation and formal ontology. The former applies at a very high level, and is usually modelled by means of logical language primitives (e.g. `rdf:type, rdfs:subClassOf`). The latter concerns the identification of those entities that constitute the physical world, hence highly relevant and primitive as far as common sense about physics is concerned. We argue that by investigating these two distinctions, given their diverse character, we can assess the feasibility of a larger study based on the proposed method, and get an indication of its generalisability.

We use DBpedia (release 2016-10) in our study as most LOD datasets link to it. We approach this problem as a classification task, using two classification approaches: *alignment-based* (cf. Section 4.2.3) and *machine learning-based* (cf. Section 4.2.4). Since no established procedure exists, we tested different families of methods in an exploratory way. This led us to reuse – or compare to – existing work, which provides us with a baseline, which includes Típalo [86] as well as other relevant alignments between DBpedia and lexical resources (cf. Section 4.2.3).

4.2.3 Alignment-based Classification

Alignment-based methods exploit the linking structure of LOD, in particular the alignments between DBpedia, foundational ontologies, and lexical linked data, i.e. LOD datasets that encode lexical/linguistic knowledge. The advantage of these methods is their inherent unsupervised nature. Their main disadvantages are the need of studying the data models for designing suitable queries, and the potential limited coverage and errors that may accompany the alignments. We have developed **SENECA** (Selecting Entities Exploiting Linguistic Alignments), which relies on existing alignments in LOD, to make an automatic assessment of the foundational distinctions asserted over DBpedia entities. A graphical description of SENECA is depicted in Figure 4.2.

Class vs. Instance. As far as this distinction is concerned, SENECA works based on the hypothesis that common nouns are mainly classes and they are expected to be found in dictionaries, while it is less the case for proper nouns, that usually denote instances. This hypothesis was suggested by [146], who manually annotated instances in WordNet, information that SENECA reuses when available. A good quality alignment between the main LOD lexical resources and DBpedia is provided by BabelNet [150][16]. SENECA exploits these alignments and selects all the DBpedia entities that are linked to an entity in WordNet[17], Wiktionary[18] or OmegaWiki[19]. With this approach, 63,620 candidate classes have been identified, as opposed to WordNet annotations that only provide 38,701 classes. In order to further increase the potential coverage, SENECA leverages the typing axioms of Tipalo [86], broadening it to 431,254 total candidate classes. All the other DBpedia entities are assumed to be candidate instances. SENECA criteria for selecting candidate classes among DBpedia entities are depicted in Figure 4.2a.

Physical Object. Almost 600,000 DBpedia entities are only typed as `owl:Thing` or not typed at all. However, each DBpedia entity belongs to at least one Wikipedia category. Wikipedia categories have been formalised as a taxonomy of classes (i.e. by means of `rdfs:subClassOf`) and aligned to WordNet synsets in YAGO [201][20]. WordNet synsets are in turn formalised as an OWL ontology in OntoWordNet [85][21]. OntoWordNet is based on DUL, hence it is possible to navigate the taxonomy up to the DUL class for Physical Object. SENECA looks up the Wikipedia category of a DBpedia entity and follows these alignments. Additionally, it uses Típalo, which includes type axioms of DBpedia entities based on DUL classes. SENECA uses these paths of alignments and taxonomical relations, as well as the automated inferences

[16]We use BabelNet 3.6, which is aligned to WordNet 3.1

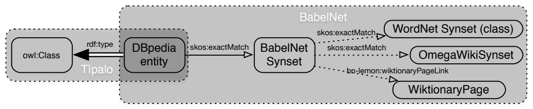
[17]http://wordnet-rdf.princeton.edu/, we use WordNet 3.0 and its alignments to WordNet 3.1, to ensure interoperability with the other resources

[18]https://www.wiktionary.org/

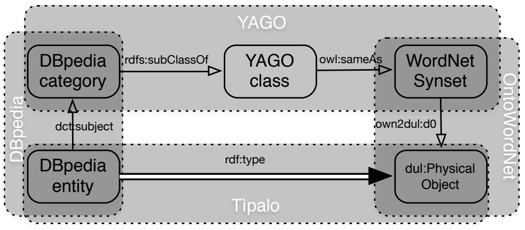
[19]http://www.omegawiki.org/

[20]We use YAGO 3, aligned to WordNet 3.1

[21]OntoWordNet is aligned to WordNet 3.0



(a) The alignment paths followed by SENECA for selecting candidate classes among DBpedia entities. It identifies as classes all DBpedia entities aligned via BabelNet to a WordNet synset, an OmegaWiki synset or a Wiktionary page, and all DBpedia entities typed as `owl:Class` in Típalo.



(b) The alignment paths used by SENECA for identifying candidate Physical Objects among DBpedia entities. It navigates the YAGO taxonomy that via OntoWordNet links DBpedia entities to `dul:PhysicalObject` or Típalo that links DBpedia entities to `dul:PhysicalObject`.

Figure 4.2: SENECA approach for assessing whether a DBpedia entity is a class or an instance (Figure 4.2a) and whether it is a physical object or not (Figure 4.2b).

that enable to assess whether a DBpedia entity is a Physical Object or not. With this approach, graphically summarised in Figure 4.2b, 67,005 entities were selected as candidate physical objects.

4.2.4 Machine Learning-based Classification

Within machine learning, *classification* is the problem of predicting which category an entity belongs to, given a set of examples, i.e. a training set. The training set is processed by an algorithm in order to learn a predictive model based on the observation of a number of *features*, which can be categorical, ordinal, integer-valued or real-valued. We have designed our target distinctions in the form of two binary classifications. We have experimented with eight classification algorithms: J48, Random Forest, REPTree, Naive Bayes, Multinomial Naive Bayes, Support Vector Machines, Logistic Regression, and K-nearest neighbours classifier. We have used WEKA[22] for their implementation.

[22]https://www.cs.waikato.ac.nz/ml/weka/

4.2.4.1 Features

The classifiers were trained using the following four features.

Abstract. Considering that DBpedia entities are all associated with an abstract providing a definitional text, our assumption is that these texts encode useful distinctive patterns. Hence, we retrieve DBpedia entity abstracts, and represent them as 0-1 vectors (bags of words). We built a dictionary containing the 1000 most frequent tokens found in all the abstracts of the dataset. The dictionary is case-sensitive, since the tokens are not normalised. The resulting vector has a value 1 for each token mentioned in the abstract, 0 for the others. By inspecting a good amount of abstracts, we noticed that very frequent words, such as conjunctions and determiners, are used in a way that can be informative for this type of classifications. For example, most of class definitions begin with "A" ("A knife is a tool..."). For this reason, we did not remove stop-words.

URI. We notice that the ID part of URIs is often as informative as a label, and often follows conventions that may be discriminating especially for the class vs. instance classification. In DBpedia, the ID of a URI reflects an entity name (it is common practice in order to make the URI more human-readable), and it always starts with an upper case letter. If the entity's name is a compound term and the entity denotes an instance, each of its components starts with a capital letter. We have also noticed that DBpedia entity names are always mentioned at the beginning of their abstract and, for most of the instance entities, they have the same capitalisation pattern as the URI ID. Moreover, instances tend to have more terms in their ID than classes. These observations were captured by three numerical features: *(i)* number of terms in the ID starting with a capital letter, *(ii)* number of terms in the ID that are also found in the abstract, and *(iii)* number of terms in the ID.

Incoming and Outgoing Properties. As part of our exploratory approach, we want to test the ability of LOD to show relevant patterns leading to foundational distinctions. Given that triples are the core tool of LOD, we model a feature based on ongoing and outgoing properties of a DBpedia entity. An outgoing property of a DBpedia entity is a property of a triple having the entity as subject. On the contrary, an incoming property is a property of a triple having the entity as object. For example, considering the triple `dbr:Rome :locatedIn dbr:Italy`, the property `:locatedIn` is an outgoing property for `dbr:Rome` and an incoming property for `dbr:Italy`. For each DBpedia entity, we count its incoming and outgoing properties, per type. For example, properties such as `dbo:birthPlace` or `dbo:birthDate` are common outgoing properties of an individual person, hence their presence suggests that the entity is an individual.

Outcome of SENECA. Following an exploratory approach, we decided to use the output of SENECA as a binomial feature (taking value “yes” or “no”) for the classifiers (excluding Multinomial Naive Bayes classifier).

4.2.5 Reference Datasets

In order to perform our experiments and evaluate the results of our approach (cf. Section 4.2.2), we have created two datasets for each of the foundational distinctions under study: one annotated by experts, and another one annotated by the crowd. In this way we can get an indication whether foundational distinctions match common sense. The resulting datasets include a sample of annotated DBpedia entities and are available online [23].

Selecting DBpedia Entities. The first step to build the datasets is to select a sample of DBpedia entities to be submitted for annotations. It is not straightforward to select a balanced number of classes and instances from DBpedia. A random selection would cause a strong unbalance towards instances because DBpedia contains a larger number of named individuals – e.g. Rome or Barack Obama – than

[23]https://w3id.org/fox

concepts[24]. A possible solution could be to manually select a sufficient equal number of DBpedia instances and classes, however this may inject a bias in the datasets. We have opted for a compromise solution by following the intuition that if entities are represented as vectors, their neighbour vectors would include classes and instances with a more balanced ratio than the random choice. As for minimising the bias, we only manually select an arbitrarily small (i.e. 20) number of seeds (equally distributed). We leverage NASARI [42], a resource of vector representations for BabelNet synsets and Wikipedia entities. For each vector corresponding to a seed entity, we retrieve its 100 nearest neighbours[25]. After cleaning off duplicated entities (e.g. Wikipedia redirects), entities without abstracts, disambiguation pages, etc., we still assessed (through expert annotations) an unbalance towards instances (∼35 classes, ∼65 instances). In the light of a broader usage of the same dataset to also annotate the distinction between physical objects and not physical objects, we enriched the sample with class entities representing physical locations (a good source of physical objects). In order to select only physical location classes from DBpedia, we used the corresponding DBpedia category `dbc:Places` and the SUN database[26], a computer vision-oriented dataset containing a collection of annotated images, covering a large variety of environmental scenes, places, and the objects within. We retrieved DBpedia entities whose labels match SUN locations or that belong to the category `dbc:Places`, and added a number of them to the sample that would suffice to improve the balance. As a result, a total number of 4502 entities were collected in the newly created dataset.

$\mathbf{CI_E}$ Dataset: Class vs. Instance Annotation Performed by Experts. Two authors of the paper have manually and independently annotated all entities by indicating whether they were instances or classes, using the associated DBpedia ab-

[24]A rough estimation made on a random sample of DBpedia showed that ∼90 of entities are instances.

[25]We observed that by picking 100 nearest neighbour entities the cosine similarity is always above a threshold of 0.6

[26]https://groups.csail.mit.edu/vision/SUN/

stract as reference description. Their judgements showed an agreement of 93,6%: they only disagreed on 286 entities. From a joint second inspection, they agreed on additional 281 entities that were initially misclassified by one of the two. Examples of misclassified entities are: `dbr:Select_Comfort` (a U.S. manufacturer) that was erroneously annotated as class; `dbr:Catawba_Valley_Pottery` (a kind of pottery) annotated as instance instead of class. Among the remaining entities, five are *polysemous* cases, where the entity and its description point to both types of referents, e.g. `dbr:Slide_Away_Bed` is a trademark commonly used also to refer to a type of beds. The authors decided to annotate these entities as classes. As a result, the CI_E annotated dataset contains 1983 classes and 2519 instances, which is reasonable balanced (44% classes, 56% instances).

CI_C Dataset: Class vs. Instance Annotation Performed by Crowd. The same dataset was then crowdsourced: each worker was asked to indicate whether an entity is a class or an instance based on its name, abstract, and its corresponding Wikipedia article. We want to assess the agreement between the experts and the crowd, which indicates whether foundational distinctions match common sense or not. The task was executed on Figure Eight [27] by English speakers with high trustworthiness. The quality of the contributors has been assessed with 51 test questions with a tolerance of only 20% of errors. We collected 22,510 judgments from 117 contributors: each entity was annotated by at least 5 different workers. For each entity e, we computed the level of agreement on each class c, weighted by the trustworthiness scores

$$agreement(e,c) = \frac{SumOfTrust(e,c)}{SumOfTrustOfWorkers(e)} \tag{4.1}$$

where $SumOfTrust(e,c)$ is the sum of the trustworthiness scores of the workers that annotated entity e with class c; and $SumOfTrustOfWorkers(e)$ is the sum of the trustworthiness scores of all the workers that annotated the entity e. Table 4.1 reports the results of the task indicating the distribution of classes and instances per level of agreement. The average agreement of the crowd is 95.76% . We compared

[27] https://www.figure-eight.com/

Agreement	# Class	# Instance	Total
≥ 0.5	1934	2568	4502
≥ 0.6	1884	2495	4379
≥ 0.8	1631	2330	3961

Table 4.1: CI_C dataset crowd-based annotated dataset of classes and instances. The table provides an insight of the dataset per level of agreement. Agreement values computed according to Formula 4.1.

crowd's annotations (with agreement greater than 0.5) against experts' ones. The judgements of the crowd workers diverge from the experts' only on 193 entities, i.e. agreement is 95,7%, suggesting that *the instance vs. class foundational distinction matches common sense* (cf. RQ1 applied to this distinction). Some of those entities (61) also caused a disagreement between experts, hence denoting ambiguous cases. Examples include polysemic entities such as `dbr:Zeke_the_Wonder_Dog` or music genres (e.g. `dbr:Ragga`).

PO_E Dataset: Physical Object Annotation Performed by Experts. Two authors of the paper further annotated (independently) the dataset by indicating for each entity whether it referred to a physical object (PO) or not (NPO), using its DBpedia abstract as reference description. They only disagreed on 272 entities, showing an agreement of 93,9%. By means of a joint second inspection, they agreed that the disagreement was caused by errors in the classification, some of which were borderline cases e.g.: *communities* (e.g. `dbr:Desert_Lake,_California`), wrongly interpreted as society instead of neighbourhood, *trademarked materials* (e.g. `dbr:Waxtite`) and entities with complex description (e.g. `dbr:Cabaña_pasiega`). The resulting PO_E annotated dataset contains 3055 POs and 1447 NPOs.

PO_C Dataset: Physical Object Annotation Performed by Crowd. We also crowdsourced the annotations of physical objects vs. not a physical object: the

Agreement	# Physical Object	# ¬ Physical Object	Total
≥ 0.5	3601	901	4502
≥ 0.6	3448	641	4089
≥ 0.8	2989	335	3324

Table 4.2: PO_C dataset: crowd-based annotated dataset of physical objects. The table provides an insight of the dataset per level of agreement. Agreement values computed according to Formula 4.1.

workers were asked to perform this task by using the entity's name, its abstract, and Wikipedia page as reference descriptions. The quality of the workers has been assessed with 49 test questions, used to exclude contributors that scored an accuracy lower than 80%[28]. We collected 25,776 judgments from 173 workers. Each entity has been annotated by at least 5 different English speakers. Table 4.2 summarises the level of agreement associated with the distribution of PO vs. NPO annotations. The average agreement of the crowd's annotations is 85.48% . The agreement between the crowd and the experts is 85,69%, suggesting that *the PO vs NPO foundational distinction also matches common sense.*

4.2.6 Evaluation

This Section presents a set of reproducible experiments evaluating the methods presented in Chapter 4 for assessing the two foundational distinctions (*class vs. instance* and *physical object vs. not a physical object*). The results of the performed experiments are expressed in terms of precision, recall and F1 measure, computed for each classification and for each target class (class vs. instance and physical object vs. ¬ physical object). The average F1 score is also provided. We compare the results of the methods, described in Section 4.2.2, against the reference datasets CI_E, CI_C, PO_E and PO_C, described in Section 4.2.5. As for CI_C and PO_C, we only

[28]We follow quality strategies recommended by the crowdsourcing platform team

Dataset	$\mathbf{P^C}$	$\mathbf{R^C}$	$\mathbf{F_1^C}$	$\mathbf{P^I}$	$\mathbf{R^I}$	$\mathbf{F_1^I}$	$\overline{\mathbf{F_1}}$
CI_E	.919	.693	.796	.753	.939	.836	.813
CI_C	**.935**	**.731**	**.818**	**.778**	**.945**	**.853**	**.836**
Dataset	$\mathbf{P^{PO}}$	$\mathbf{R^{PO}}$	$\mathbf{F_1^{PO}}$	$\mathbf{P^{NPO}}$	$\mathbf{R^{NPO}}$	$\mathbf{F_1^{NPO}}$	$\overline{\mathbf{F_1}}$
PO_E	.877	.247	.385	.561	.965	.713	.548
PO_C	**.954**	**.247**	**.393**	**.567**	**.988**	**.721**	**.557**

Table 4.3: Results of SENECA on the Class vs. Instance and Physical Object classifications compared against the reference datasets described in Section 4.2.5. P^*, R^* and F_1^* indicate precision, recall and F1 measure on Class (C), Instance (I), Physical Object (PO) and complement of Physical Object (NPO). $\overline{F_1}$ is the average of the F1 measures.

include the annotations having agreement $\geq 80\%$.

4.2.7 Alignment-based Methods: SENECA

Class vs. Instance. Table 4.3 shows SENECA's performance on the class vs. instance classification, by comparing its results with CI_E and CI_C. SENECA shows very good performance with best avg F1 = .836, when compared with CI_C. Considering that SENECA is unsupervised, and is based on existing alignments in LOD, this result suggests that LOD may better reflect common sense than the expert's perspective, an interesting hint for further investigation on this specific matter.

Physical Object. Table 4.3 shows the performance of SENECA on the Physical Object classification task computed by comparing its results with PO_E and PO_C (cf. Section 4.2.5). We observe a significant drop in the best average F1 score (.557) as compared to the class vs. instance classification task (.836). On one hand, this may suggest that the task is harder. On the other hand, the alignment paths followed in the two cases are different, since for classifying Physical Objects more alignment steps are required. In the first case (class vs. instance), BabelNet directly provides the final alignment step (cf. Figure 4.2a), while in the second case (PO vs. NPO), three more alignment steps are required: DBpedia Category → YAGO → WordNet (cf. Figure 4.2b). It is reasonable to think that this implies a higher potential of error propagation along the flow. This hypothesis is partly supported by [86], who report a similar drop when they add an automatic disambiguation step followed by an alignment step to DUL classes (including Physical Object). Also for this distinction, SENECA better matches the judgements of the crowd than the experts'.

4.2.8 Machine Learning Methods

We performed a set of experiments with eight classifiers: J48, Random Forest, REPTree, Naive Bayes, Multinomial Naive Bayes, Support Vector Machines, Logistic Regression, and K-Nearest Neighbors (cf. Section 4.2.4). We used a 10-fold cross validation strategy using the reference datasets (cf. Section 4.2.5). Before training the classifiers, the datasets were adjusted in order to balance the samples of the two classes. The $\mathrm{CI_E}$ and $\mathrm{PO_E}$ datasets were balanced by randomly removing a set of annotated entities. $\mathrm{CI_C}$ and $\mathrm{PO_C}$ were balanced by removing entities associated with lower agreement (which constitute weak examples for the classifiers). Each classifier was trained and tested with all four features, described in Section 4.2.4, both individually and in all possible combinations, with and without performing feature selection. We found that performing feature selection makes the results worse. Having two datasets for each classification (i.e. annotated by the experts and by the crowd) enables multiple configurations of the training set. When we train the classifiers with samples from $\mathrm{CI_E}$ and $\mathrm{PO_E}$, they all have the same weight = 1. Differently, when the samples come from $\mathrm{CI_C}$ and $\mathrm{PO_C}$, they are weighted according to their associated agreement score *agreement(e, c)*, computed with Formula 4.1 (cf. Section 4.2.5). As previously studied by [6], this diverse weighing allows a classifier to learn richer information, including ambiguity and consequent entities that may belong to an "unknown" class, which better represent human cognitive behaviour.

We only report the results of the best performing algorithm[29], which is Support Vector Machine, without feature selection, trained and tested on samples associated with an agreement score $\geq 80\%$. We report on all combinations of features, but **D** alone (i.e. SENECA's output).

Class vs. Instance. Table 4.4 shows the results of Support Vector Machine, trained on and tested against $\mathrm{CI_E}$ and $\mathrm{CI_C}$. The best average performance is obtained with $\mathrm{CI_C}$ by combining all features. Combining all features is also the best configuration for each individual classification (i.e. Class (C) and Instance (I)). When $\mathrm{CI_E}$ is used there is a slight drop in performance, although the quality of the classification remains high. A possible cause of this result may be the agreement-based weighing provided by $\mathrm{CI_C}$.

Physical Object. Table 4.5 also shows the results of the Support Vector Machine algorithm trained on and tested against $\mathrm{PO_E}$ and $\mathrm{PO_C}$. Similarly to the behaviour of SENECA, statistical approaches worsen their overall performance as compared to the case of the class vs. instance classification. We also observe a different behaviour of the individual features. The best average performance with $\mathrm{PO_E}$ is achieved by combining all the feature, while the best average performance with $\mathrm{PO_C}$ is achieved by combining the abstract (**A**), the outgoing/incoming properties (**E**), and SENECA output (**D**). In a sense, this confirms that conventions used for creating URI IDs are informative mainly for the class vs. instance distinction.

4.3 Discussion

This Chapter presented two different lines of research that investigate the possibility of providing Linked Open Data as background knowledge for supporting daily tasks. The two studies focused on linguistic and common sense knowledge respectively.

[29]The results of all experiments are available online at https://github.com/fdistinctions/ijcai18

				Results compared against $\mathrm{CI_E}$							*Results compared against* $\mathrm{CI_C}$						
A	**U**	**E**	**D**	$\mathbf{P^C}$	$\mathbf{R^C}$	$\mathbf{F_1^C}$	$\mathbf{P^I}$	$\mathbf{R^I}$	$\mathbf{F_1^I}$	$\overline{\mathbf{F_1}}$	$\mathbf{P^C}$	$\mathbf{R^C}$	$\mathbf{F_1^C}$	$\mathbf{P^I}$	$\mathbf{R^I}$	$\mathbf{F_1^I}$	$\overline{\mathbf{F_1}}$
✓				.927	.921	.924	.921	.927	.924	.924	.958	.965	.961	.965	.957	.961	.961
	✓			.881	.933	.906	.929	.873	.909	.903	.908	.970	.938	.967	.902	.933	.936
		✓		.854	.975	.911	.971	.834	.897	.904	.886	.983	.932	.981	.874	.924	.928
✓	✓			.928	.935	.932	.935	.928	.931	.932	.966	.971	.968	.971	.966	.968	.968
✓		✓		.939	.943	.941	.943	.939	.941	.941	.971	.976	.974	.976	.971	.973	.974
✓			✓	.934	.927	.939	.928	.934	.931	.931	.966	.964	.965	.964	.966	.965	.965
	✓	✓		.919	.968	.943	.966	.914	.939	.941	.961	.982	.971	.981	.963	.979	.971
	✓		✓	.881	.939	.909	.935	.873	.903	.906	.908	.973	.939	.971	.902	.935	.937
		✓	✓	.859	**.978**	.915	**.975**	.846	.903	.909	.889	**.987**	.935	**.985**	.877	.928	.932
✓	✓	✓		.942	.946	.944	.945	.942	.944	.944	.973	.981	.976	.980	.973	.976	.976
✓	✓		✓	.939	.933	.936	.934	.939	.937	.936	.968	.969	.968	.969	.968	.968	.968
✓		✓	✓	.945	.949	.943	.941	**.946**	.943	.943	.973	.976	.975	.976	.973	.975	.975
	✓	✓	✓	.926	.967	.946	.966	.922	.944	.945	.964	.981	.973	.981	.964	.972	.973
✓	✓	✓	✓	**.946**	.949	**.947**	.948	**.946**	**.947**	**.947**	**.981**	.982	**.982**	.982	**.981**	**.982**	**.982**

Table 4.4: Results of the Support Vector Machine classifier on Class vs. Instance task against the reference datasets described in Section 4.2.5. The first four columns indicate the features used by the classifier: A is the abstract, U is the URI, E are incoming and outgoing properties, D are the results of the alignment-based methods. $\mathrm{P^*}$, $\mathrm{R^*}$, $\mathrm{F_1^*}$ indicate precision, recall and F1 measure on Class (C) and Instance (I). $\overline{\mathrm{F_1}}$ is the average of the F1 measures.

Providing Linguistic Knowledge. The first line of research focuses on linguistic, factual and encyclopedic data that has been retrieved from the web and integrated in a unique knowledge graph, called Framester. The Framester's backbone is a huge linguistic sense inventory, mostly encoded by using semantic frames, which integrates resources like FrameNet, WordNet and BabelNet. This feature makes Framester as an ideal entry for the robot's knowledge when the interaction with users is through natural language. In fact, Framester allows to easily retrieves all the knowledge related to the words meanings of an interaction.

The main Framster drawback is its coverage and ongoing work is about integrating and linking Framester's linguistic information with other kinds of knowledge, so

				Results compared against $\mathrm{PO_E}$							*Results compared against* $\mathrm{PO_C}$						
A	**U**	**E**	**D**	$\mathbf{P^{PO}}$	$\mathbf{R^{PO}}$	$\mathbf{F_1^{PO}}$	$\mathbf{P^{NPO}}$	$\mathbf{R^{NPO}}$	$\mathbf{F_1^{NPO}}$	$\overline{\mathbf{F_1}}$	$\mathbf{P^{PO}}$	$\mathbf{R^{PO}}$	$\mathbf{F_1^{PO}}$	$\mathbf{P^{NPO}}$	$\mathbf{R^{NPO}}$	$\mathbf{F_1^{NPO}}$	$\overline{\mathbf{F_1}}$
✓				.828	.814	.821	.817	.832	.824	.823	.879	.837	.858	.844	.884	.864	.861
	✓			.615	.822	.703	.732	.485	.584	.644	.596	.863	.705	.751	.413	.533	.619
		✓		.786	.865	.824	.857	.764	.805	.814	.782	.886	.831	.868	.752	.806	.818
✓	✓			.831	.829	.838	.833	.832	.831	.831	.851	.811	.831	.819	.857	.838	.834
✓		✓		.869	.867	.868	.867	.870	.868	.868	.912	.853	.882	.862	.918	.889	.885
✓			✓	.849	.829	.835	.831	.843	.837	.836	.865	.834	.849	.839	.869	.854	.852
	✓	✓		.816	.852	.833	.845	.808	.826	.832	.802	**.889**	.842	.875	.777	.823	.833
	✓		✓	.659	.761	.707	.718	.607	.658	.682	.951	.243	.387	.565	**.987**	.719	.553
		✓	✓	.928	.735	.826	.788	.943	.854	.837	**.966**	.762	.852	.803	.973	.886	.866
✓	✓	✓		.865	.866	.866	.866	.865	.865	.865	.927	.847	.885	.859	.933	.894	.891
✓	✓		✓	.831	.824	.828	.826	.833	.829	.828	.878	.831	.855	.838	.876	.856	.853
✓		✓	✓	.867	.862	.864	.863	.868	.865	.865	.922	.879	**.899**	**.884**	.923	**.903**	**.901**
	✓	✓	✓	**.933**	.736	.823	.782	**.947**	.857	.843	.962	.759	.849	.801	.975	.877	.863
✓	✓	✓	✓	.871	**.877**	**.871**	**.879**	.872	**.871**	**.871**	.905	.866	.886	.872	.909	.895	.888

Table 4.5: Results of the Support Vector Machine classifier on Physical Object classification task against the reference datasets described in Section 4.2.5. The first four columns indicate the features used by the classifier: A is the abstract, U is the URI, E are incoming and outgoing properties, D are the results of the alignment-based methods. P^*, R^*, F_1^* indicate precision, recall and F1 measure on Physical Object (PO) and the complement of Physical Object (NPO). $\overline{F_1}$ is the average of the F1 measures.

to provide robots with a richer human-like knowledge base. Another line of research is on improving linguistic coverage of Framester's frames in cataloging and describing situations. Methods like [176, 163] could be used to extend the lexical units associated with frames, but the main lack of FrameNet-based datasets is the scarcity of semantic roles and semantic types associated with frames. A possible solution could be analyzing statistical correlation (by using tools such as sense embedding [42, 41]) between occurrence of frames and ontology classes within a corpus. We hypothesize that classes statistically correlated with a frame **f** are the semantic types involved in situations described by **f**. Further work (e.g. involving crowdsourcing techniques) is needed to determine semantic roles that entities of the discovered semantic types play in situations described by **f**.

Providing Common Sense Knowledge. The second line of research presented in this Chapter is concerned with the provision of Common Sense Knowledge to social robots. This study reports a set of experiments for assessing whether Linked Open Data provides an empirical basis to extract foundational distinctions, and if they match common sense. We claim that the performed experiments show promising results as far as our research questions are concerned. RQ2 of Section 1.1 has been further refined in the questions discussed below (in line with the research goals introduced in Section 4.2). Given the diversity and the basic nature of the two distinctions that we have analysed, and the positive results obtained in both cases by applying the same methods with the same configurations, we claim that the proposed methods can be generalised to other foundational distinctions.

Do foundational distinctions match common sense? As anticipated in Section 4.2.5 the high agreement observed among workers that participated in the crowdsourcing tasks, as well as the high agreement between the crowd and the experts, suggest that the foundational distinctions that we have tested do actually match common sense.

Does the (Semantic) Web provide an empirical basis for supporting foundational distinctions over LOD entities, according to common sense? We claim that the high average value of F1 measure associated with all experiments indicates that the Web, and in particular LOD, implicitly encodes foundational distinctions. We also think that, more in general, this is a hint that the Web is a good source for common sense knowledge extraction. We find particularly interesting to observe that the feature **E** (i.e. ongoing/incoming properties) has always a positive impact, in all features combinations, on the classifier's performance (cf. Table 4.4 and 4.5), for both tasks. This motivates us in conducting further investigations *(i)* towards identifying and testing additional features based on LOD, e.g. more sophisticated use of assertions and axioms from LOD as well as *(ii)* to analyse LOD at a much larger scale (e.g. by using LOD Laundromat [26, 73]) with an *empirical science* perspective: looking for emerging patterns that may encode relevant pieces of common sense knowledge [89]. Our promising results open a number of possible research directions: besides replicating these experiments at a larger scale, we plan a follow up study concerning the application of the same approach to distinguishing physical objects that can act as locations for other physical objects. This is particularly relevant in order to extract knowledge about where things are usually located in, whether a location is appropriate for an object in terms of its size, etc. Another relevant distinction to be investigated with priority is the one between physical and social objects (e.g. organisations), which is often prone to *systematic polysemy* [175], i.e. objects that have a same linguistic reference, but different (disjoint) types of referents. For example, the term *National Library* is used to refer both to an organisation (a social object) taking care of the library's collections, and of the related administrative and organisational issues, and to the buildings (physical objects) where the organisational staff works and the collections are located in. Besides covering foundational distinctions, we aim to extend our approach to learn or discover relational knowledge such as the one modelled and encoded in terms of frames [75, 81].

What ensemble of features, resources, and methods works best to make machines learn foundational distinctions over LOD entities? According to our results, statistical methods perform better than alignment-based methods. We use supervised learning and crowdsourcing to test two very diverse foundational distinctions, both very basic in knowledge representation and foundational ontologies. It emerges that two features show the same ability to positively impact on the methods' performance, for both distinctions: **A** (a text describing the entity) and **E** (entity's outgoing and incoming properties). Both features convey the semantic description of an entity: the former by means of natural language, which characterises a huge portion of the web, the latter by means of LOD triples, which characterise the semantic web. Based on these observations, we argue that the method can be generalised, even if each specific distinction may benefit from a specialised extension of the feature set. In our case, the **U** feature (i.e. URI ID) clearly shows effectiveness for the class vs. instance rather than for PO vs. NPO. A question is whether DBpedia text is special because of its “standardised” style of writing. Our experiments and results do not cover this issue, which needs to be assessed in order to provide a stronger support to our claim of generalisability. A similar doubt can be raised as far as outgoing and incoming links are concerned. DBpedia properties mainly come from infoboxes, which also follow, and are influenced by the standardised way of writing Wikipedia pages. Nevertheless, for this feature we argue that the doubt does not apply, since incoming and outgoing properties include links to and from LOD datasets that are outside DBpedia, hence independent from the standardised content of Wikipedia.

Chapter 5

Accessing Background Knowledge using Lizard

Nowadays a growing number of robotic frameworks relies on RDF and triple stores for managing robots' knowledge. Examples of such a framework are KnowRob [206], RoboBrain [186], RACE [182]. Providing robotic applications with a simple interface to access robot's knowledge (cf. RQ3 Section 1.1) is a tricky task for two reasons: *i* the wide variety of programming languages used to implement robotic applications; *ii* the perceived difficulty of adopting Semantic Web technologies in software development [111]. In recent years a number of frameworks have been proposed to foster the adoption of Semantic Web technologies in software development. Examples are Apache Jena[1] [46], OWL API[2] [102] or RDF4J[3] (previously known as Sesame [40]). These frameworks provide the basic facilities for manipulating/querying and reasoning on RDF(S)/OWL compliant data. These frameworks are the basic building blocks on top of which it is possible to design and implement complex systems that rely on Semantic Web standard languages and technologies. However, these tools require developers with extensive knowledge of models of Semantic Web technologies and knowledge engineering. This knowledge is mandatory in order to use these frameworks effectively. We propose a software framework aimed at easing

[1]Jena, https://jena.apache.org/
[2]OWL API, http://owlapi.sourceforge.net/
[3]RDF4J, http://rdf4j.org/

the software development of knowledge-aware systems by filling the gap between Semantic Web technologies and Object-Oriented applications. This framework is called *Lizard*. *Lizard* is an Object-RDF mapper providing software components with the access to the knowledge base following the Object-Oriented paradigm. In particular, given an ontology specified in OWL language as input, Lizard is able to generate an application bundle (called ontology bundle) that provides applications with an API for accessing RDF facts stored in a knowledge base This API enable software components to access programmatically to the knowledge base following the object-oriented paradigm and without dealing with Semantic Web models and languages. The API reflect the semantics of the input ontology and allow transparent access to the knowledge base. Differently from existing systems (such as SuRF[4] or ActiveRDF[5] [157]), the Ontology Bundle also provides a RESTful layer that exposes the API following the REST architectural style over HTTP. Lizard also generates a description of the REST APIs (in OpenAPI language[6]). This description can be used to generate the code implementing clients of the RESTful API in more than 40 programming languages. As a result, ontology bundles *(i)* avoid client applications to deal with OWL and RDF; *(ii)* avoid client applications to interact with a knowledge base by means of SPARQL queries; *(iii)* allow client applications to programmatically interact with knowledge bases following the Object-Oriented paradigm; *(iv)* allow client applications to interact with a knowledge base in more than 40 programming languages.

Lizard is an open source project available on online as a Git repository at the following link[7]. The repository contains three sub-projects: *(i) Lizard Main* implements the main process for generating the ontology API; *(ii) Lizard Commons* contains utility classes for the main process and implements the RESTful Web Service that exposes the Java API functionalities (cf. Section 5.3.2); *(iii) Lizard Jetty* runs a Jetty server exposing the RESTful Web Service.

[4]SuRF, https://pythonhosted.org/SuRF/

[5]Active RDF, https://github.com/ActiveRDF/ActiveRDF

[6]OpenApi (or Swagger), https://swagger.io/

[7]Lizard's Repository, https://github.com/anuzzolese/lizard

5.1 Requirements

The requirements that led the design of Lizard have been derived from the case study of this thesis (i.e. the MARIO project). The objective of the project was developing a social robot for assisting people with dementia. A social robot is an example of agent that continuously performs knowledge-intensive tasks and needs an easy access to its knowledge base. The main requirements of Lizard are the following:

R1 **Generation of procedural API for accessing data complying with an OWL ontology**. This requirement guarantees that *(i)* Lizard generates an API that follows the semantics defined in a source ontology; *(ii)* The procedural API guarantees the programmatic access to ontological artifacts of a knowledge base.

R2. **Generation of REST API for accessing data complying with an OWL ontology**. This requirement guarantees that: *(i)* Lizard generates a REST API that follows the semantics defined by a source ontology; *(ii)* Ontology artifacts can be accessed as a service by external components via HTTP requests like GET, POST, PUT and UPDATE; *(iii)* Lizard allows to bind software components to ontological artifacts via REST API.

R3. **Dynamic adaptivity of produced API to the Ontology**. Lizard has to adapt the API it produces and exposes (both Java and REST) according to any change occurring in the Ontology. By change we mean any addition, deletion or update of the Ontology ranging from a single axiom to a whole Ontology. This requirement is crucial in order to keep valid at runtime the binding between the API produced by Lizard and the Ontology.

R4. **Programmatic access to the Knowledge Base via Java**. The Java API produced by Lizard has to grant access to the Knowledge Base to the components of the software architecture (cf. Chapter 7). The Knowledge Base consists of a dataset in RDF format, modelled according to the semantics expressed by the Ontology Network and stored in a triplestore (e.g., Virtuoso[8] or Apache Jena TDB[9]). The access via Java allows software components to perform operations such as: query the Knowledge Base, retrieve knowledge expressed as RDF, update the Knowledge Base, and delete facts from the Knowledge Base.

R5. Programmatic access to the Knowledge Base via REST. The REST APIs generated by Lizard have to guarantee the access to the Knowledge Base to the components of the software architecture (cf. Chapter 7). The access via REST has to reflect all the operations enabled via Java, i.e., querying, updating, deleting facts from, and retrieving facts from the Knowledge Base.

R6. Transparent access (with respect to client application) to the Knowledge Base. It is desirable to enable software developers to access the knowledge base without a deep knowledge of RDF(S)/OWL and SPARQL languages.

R7. Easy generation of language-agnostic clients for the APIs. It is desirable that applications intending to use the APIs are provided with facilities for generating API clients.

R8. Access the Knowledge Base by following the Object-Oriented programming paradigm. Lizard has to allow client applications to access to the facts stored in the Knowledge Base by following the Object-Oriented paradigm. This requirement aims at filling the gap between Semantic Web technologies and Object-Oriented applications.

5.2 Architecture

Intuitively, given an OWL ontology as input, Lizard generates a software artifacts, called Ontology Bundle implementing a set of APIs for creating, retrieving, updating, deleting facts from a Knowledge Base by reflecting the semantics defined in the

[8]Virtuoso, http://virtuoso.openlinksw.com/.

[9]Apache Jena TDB, https://jena.apache.org/documentation/tdb/.

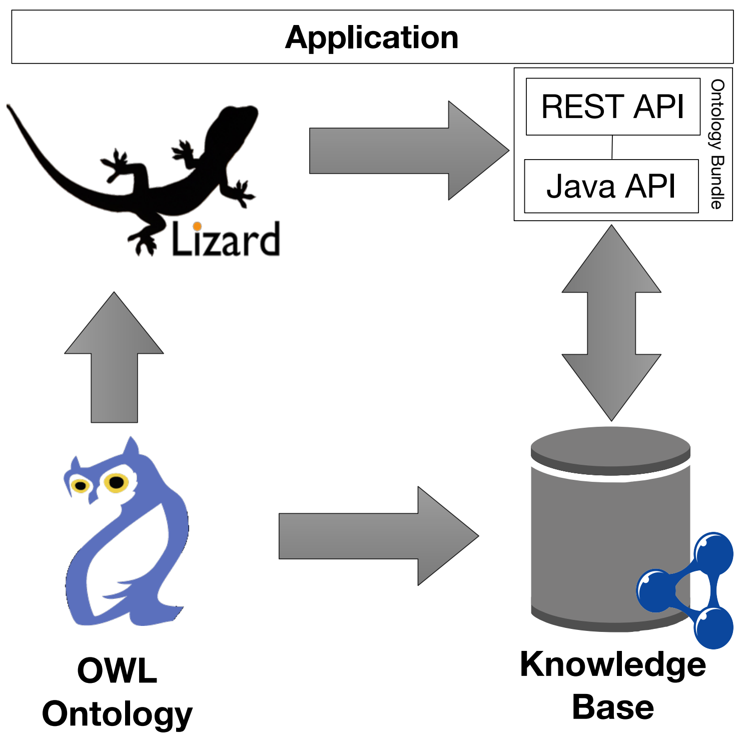


Figure 5.1: A diagram that shows the intuition behind Lizard and its operating scenario.

input Ontology. The Figure 5.1 shows the intuition behind Lizard and its operating scenario. Lizard comes into play when there is an application that need to interact with a Knowledge Base storing a set of RDF triples that comply with an OWL ontology. In this situation the application's developer either could interact with the knowledge base by means of standard semantic web technologies or could generate a Lizard's Ontology Bundle and interact with RDF resources by following the Object-oriented paradigm. In the former case, it is required that the application's developer has a deep understanding of languages, models and protocols involved by Semantic Web technologies. In the latter case, the application can interact with the knowledge base without any prior knowledge thus avoiding application to directly deal with RDF(S), OWL and SPARQL. The Lizard's Ontology Bundle reflects the semantics defined by the input ontology and provide the access to the RDF resources though a set of RESTful APIs which rely on the generated Java APIs.

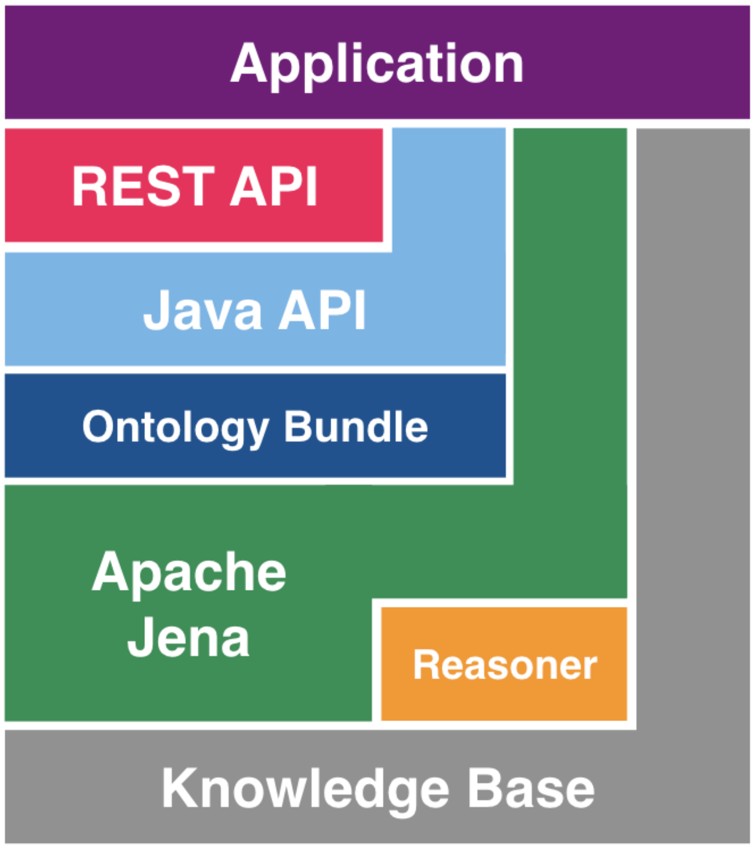


Figure 5.2: The solution stack provided by Lizard that allow applications to interact with a Knowledge Base.

The Figure 5.2 presents the solution stack provided by Lizard. This solution stack provides application with four mechanisms for interacting with a Knowledge Base. These mechanisms range from the direct interaction with the knowledge base to an interaction mediated by a set of RESTful APIs.

Direct interaction with the Knowledge Base. The basic mechanism is the direct interaction with the knowledge base. This is done through the SPARQL and SPARUL protocols and languages. SPARQL is able to retrieve and manipulate data stored in RDF through four different query variations: SELECT (that extracts values and returns the result in a table format), CONSTRUCT (that extracts values and returns the result in RDF format), ASK (that assesses a boolean condition in

```
Query q = QueryFactory.create("SELECT * WHERE {?s ?p ?o}");
QueryExecution qexec =
  QueryExecutionFactory.createServiceRequest(
   "http://dbpedia.org/sparql",q);
ResultSet rs = qexec.execSelect();
while (rs.hasNext()) {
  QuerySolution qs = rs.next();
  Resource s = qs.getResource("s");
  Resource p = qs.getResource("p");
  Resource o = qs.getResource("o");
  ...
}
```

Frame 5.1: The Java code needed for executing a SPARQL query on the DBPedia endpoint by using Apache Jena.

the knowledge base), DESCRIBE (that describes a resource of the knowledge base). SPARUL is a declarative query language that allows to add and remove facts from the knowledge base through INSERT and DELETE methods. The main advantage of SPARQL and SPARUL is their flexibility. In fact, these two languages allow to define complex patterns and conditions to be verified in a single query. The disadvantage is that SPARQL and SPARUL require an extensive knowledge of the RDF model, the SPARQL/SPARUL languages, and SPARQL/SPARUL protocols.

Interacting with the Knowledge Base through RDF API. RDF API frameworks (such as Apache Jena, RDF4J, RDFLib[10] and RDFHDT [74]) define an high-level interface for working with RDF data in a programming environment. These frameworks implement of SPARQL/SPARUL protocols thus providing client applications with facilities for submitting and retrieving queries with few lines of code (cf. Frame 5.1). RDFAPI frameworks also provide client applications with a set of programmatic API for listing all triples having a specific subject/predicate/object or adding (removing) a triple to (from) the knowledge base. These APIs do not

[10]RDFLib, https://github.com/RDFLib

require the knowledge of SPARQL/SPARUL protocols but some familiarity with RDF is still needed.

Interacting with the Knowledge Base through Object-Oriented Java API. The Ontology Bundle generated by Lizard allows Java applications to interact with a Knowledge Base following the Object-Oriented paradigm, hence abstracting from the RDF/OWL model. Client applications that include an Ontology Bundle within their project will be able to interact with the knowledge base by means of a set of Java classes (that correspond to the classes defined in the ontology describing the Knowledge Base) and methods (that correspond to the properties defined in the ontology describing the Knowledge Base). The Java classes are populated with the resources of the knowledge base. Therefore, developers with no knowledge of semantic web technologies are able to interact with the Knowledge Base by means of the Ontology Bundle classes.

Interacting with the Knowledge Base through REST API. The Ontology Bundle exposes the Java classes as REST services, thus allowing applications written in other programming languages to benefit of the functionalities of the Ontology Bundle.

5.3 Ontology Bundle

Given an OWL ontology as input, Lizard generates an Ontology Bundle which allows client applications to interact with a knowledge base following the Object-Oriented approach. The functionalities provided by the ontology bundle comply with the semantics defined in the input ontology. These functionalities are implemented in a Java programming library that can be also accessed as a Web Service via a REST architecture running on top of the Java API.

An Ontology Bundle consists of three main components: Java API which is a Java programming library implementing the functionalities for interacting with the knowledge base, REST API which is a REST service that exposes the Java API as a Web Service, and a description of the functionalities exposed via REST provided in Swagger language. Client applications are able to access the functionalities implemented by the Ontology Bundle in three different ways: *(i)* Client applications written in Java language can include the Ontology Bundle as an external library; *(ii)* Remote applications (i.e. applications running on a machine that does not host the target Knowledge Base) can access the ontology API via HTTP methods (e.g. GET/POST) provided by the REST web service; *(iii)* Remote applications can alternatively generate a client for the REST API by using Swagger's Codegen[11] and include the generated library within their code.

5.3.1 Java API

The Java API library is generated using Sun's JCodeModel framework[12] and it implements all the functionalities for interacting with the Knowledge Base following the Object-Oriented paradigm. This library relies on Apache Jena[13] for manipulating RDF data and accessing data stored in the knowledge base.

Binding Java API with the Knowledge Base. Lizard enables the interaction with three kinds of Knowledge Bases: *(i)* SPARQL endpoint; *(ii)* Apache Jena TDB; *(iii)* RDF files. From the Lizard's configuration file the user of the Java API can choose which knowledge base s/he wants to interact with. For SPARQL endpoints s/he has to provide the URL of the endpoint, whereas for the other types of Knowledge Bases s/he has to indicate the file path within the file system. The latter two types are available only if the Java APIs run on the same machine of the knowledge base.

The Knowledge Bases are abstracted as Jena Model[14] The class *JenaLizard-*

[11]Swagger Codegen, https://swagger.io/tools/swagger-codegen/

[12]JCodeModel, https://javaee.github.io/jaxb-codemodel/

[13]Apache Jena, https://jena.apache.org/

[14]Apache Jena's Model https://jena.apache.org/documentation/javadoc/jena/org/apache/jena/rdf/model/Model.html

Context manages this abstraction. This class instantiates the Model depending on the Lizard's configuration values and then makes available the Model to the Java APIs. Moreover, Lizard can be also configured for applying inference rules on the Model. In this case, *JenaLizardContext* retrieves from the Java APIs the ontologies which the APIs correspond to and returns a Jena's Inference Model.

The Jena Models for SPARQL endpoints and TDBs keep the knowledge base updated (i.e. whenever something is updated on the Jena Model is also updated in the Knowledge Base). This feature is not available for Jena Model for the RDF files, hence the source files have to be explicitly updated. Lizard overcomes this issue by setting an update listener on the Jena Model for the RDF file Whenever something is updated on the Jena Model the listener makes sure that it is also updated on the source file.

5.3.1.1 Mapping OWL on Java

The structure of the Java API aims at reflecting as much as possible the structure and the semantics of the input ontology. The binding between OWL and Java language has been studied in some previous work [108, 2, 68]. The proposed mapping can be summarized as follows. OWL classes are used to "group individuals that have something in common in order to refer to them". Similar role is played by Java classes which "is a blueprint or prototype from which objects are created". Therefore, Java objects belonging to the same class follow the same description. As a consequence, individuals correspond to Java objects.

Properties are binary relations connecting two individuals. Properties define a domain, i.e. the class of entities that can instantiate the property, and a range, i.e. the class of entities that describes the allowed property values. In the object-oriented model, properties match with associations between two classes. Different design patterns can be applied to realize associations in Java. The choice of a design pattern depends on the responsibility and the cardinality of the association [118]. Following [108, 2], we chose to assign the responsibility of a property to the classes in the domain (that are the classes of individuals that intuitevely "holds" the property).

Therefore, a class c in the domain of a property `p` will define a field for storing the property values that objects belonging to c have for `p`. The property values for `p` can be accessed through get/set methods. The default cardinality for properties is 0:n on the domain and 0:n on the range, meaning that entities belonging to the domain of a property can be connected to minimum 0 and maximum infinite entities in the range (and vice-versa). This implies that the field corresponding to `p` has to allow to store a collection of property values. OWL allows to restrict the cardinality of properties but we left a proper management of these kind of restrictions to a future work.

OWL distinguishes between two main categories of properties *Object properties* (that link individuals to individuals) and *Datatype properties* (that links individual to data values). The type of the range of a property `p` affects the type of the field storing `p`. As discussed above, individuals are mapped to objects, therefore for object properties, the type of the field for `p` corresponds to the class in the range of `p`. OWL makes use of RDF datatype scheme (which is based on XML Schema). Primitive datatypes (like string, int, double, boolean etc.) are directly mapped to Java primitive types. The Jena library provide a type mapper for converting values from the XML schema to Java primitive type and vice-versa. The custom datatypes that could be defined in an ontology are treated as an `rdfs:Literal`.

The correspondences between ontology and Java entities will be furtherly and detailed in the following paragraphs. Here we provide the reader with an overview of the mapping:

1. Ontology classes correspond to Java Classes and Interfaces;
2. RDF individuals are mapped to Java objects;
3. Properties match with Java methods that manipulate property values stored in the fields of objects;
4. RDF Literals are turned into Java primitive data types (where it is possible, in strings otherwise);

5. All the classes defined in the input ontology are enclosed in a Java package.

5.3.1.2 Naming conventions

In OWL, ontologies, classes, properties, individuals and datatypes are denoted by IRIs (i.e. Internationalized Resource Identifier). Java naming rules do not match with the grammar for building IRIs. We devised some simple rules for deriving Java-consistent names from IRIs.

Package names are derived from the ontology IRIs as follows: *(i)* We discard the scheme of the IRI (e.g. "http://") and the prefix "www" from the host part (if it is present). *(ii)* We place suffix of the host part (e.g. "com", "org" etc.) at the beginning of the package name, followed by a dot and the rest of the host path. *(iii)* Then, we append the path of the IRI by replacing slashes with dots. *(iv)* Finally, if the IRI terminates with ".owl", this sub-string is replaced with "_owl". For example, the name package that corresponds to the Action ontology module `http://ontologydesignpatterns.org/ont/mario/action.owl` is `org.ontology designpatterns.ont.mario.action_owl`.

The name of the classes is derived by concatenating the right most part of the IRI identifying the class with the namespace prefix of the ontology. For example, the namespace prefix for the Action ontology is "action" and the name of the Java class the corresponds to the class `action:Action` is *Action_Action*[15]. A similar rule is used to generate the names of the methods from the IRIs of the properties. In this case, the prefix of the ontology is preceded by the "get"/"set" depending on the aim of method. For example, the method for getting the values for the property `action:byAgent` is named *getAction_byAgent*.

5.3.1.3 Preliminary Tasks

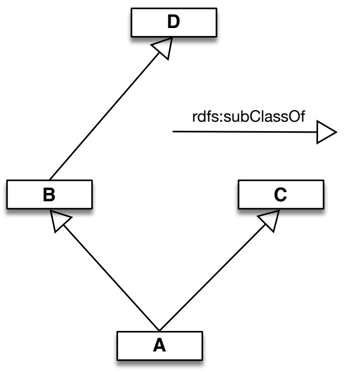
The first task performed by Lizard is to download all the ontologies imported by the input ontology. Each imported ontology will constitute a different Java pacakge. All the ontologies (i.e. the input ontology and the imported ontologies) are included in

[15]For the Java naming conventions, a class name should start with an uppercase letter.

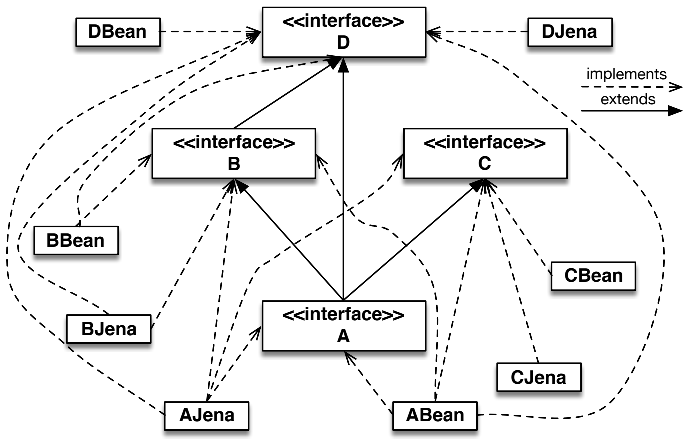
a single Jena model (i.e. called *input model*). Lizard then uses the Jena reasoning subsystem in order to derive the facts inferred from the assertions stated in the input model. The reasoning profile used by Lizard is the Jena's OWL micro profile. The Jena's OWLMicro reasoner supports RDFS rules plus the various property axioms like: `owl:disjointWith`, `owl:intersectionOf`, `owl:unionOf` and `owl:hasValue`. It omits the cardinality restrictions and equality axioms. The reasoner enables to construct the hierarchy of the classes and properties, to infer domain and ranges of properties and so on. The reasoner also checks the validity of the input model, thus avoiding that the knowledge generated by the Java API arises inconsistencies.

5.3.1.4 Hierarchy of the Java Classes

Lizard generates a Java interface and two Java classes (called Jena class and Bean class) for ontology class defined in the input model. The interface defines the signature of the methods that have to be implemented by the concrete classes that are intended to implement the semantics of the ontology class. The generated interfaces also reflect the hierarchy of the ontology classes, since only interfaces (unlike classes) are allowed to inherit from multiple parents as for ontology classes. Each interface is implemented by two classes: the Jena class and the Bean class. The Jena class aimed to implement the methods that manipulate data stored in the knowledge base. The Bean class is created to encompass all the properties of an RDF resource in a single object. The Figure 5.3 shows how Lizard turns a hierarchy of classes defined by an ontology into a hierarchy of Java classes and interfaces. The ontology (cf. 5.3a) defines four classes. A inherits from B and C, and B inherits from D. The inheritance is stated using the property `rdfs:subClassOf`. For each class in the input ontology Lizard generates an interface (i.e. A, B, C, and D) and two concrete classes (i.e. AJena, ABean, BJena, BBean, CJena, CBean, DJena, DBean). The hierarchy of the interfaces reflects the hierarchy defined in the ontology (cf. 5.3b): the interface A inherits from B, C, and D (Lizard makes explicit the fact that A inherits from D); the interface B inherits from D. A,B,C and D also inherit from *LizardInterface* which aims at generalizing all the interfaces. The concrete classes



(a) An ontology defining a hierarchy involving four classes.



(b) The Java classes and interfaces generated by Lizard that reflect the hierarchy defined in the ontology showed in Figure 5.3a.

Figure 5.3: An example showing how the hierarchy of classes defined in the input ontology (Figure 5.3a) is reflected in the Java classes generated by Lizard (Figure 5.3b).

(e.g. AJena and ABean) implement the interface corresponding to the ontology class the concrete classes are generated from (i.e. A) and its super-interfaces (i.e. B, C and D). Interfaces, Jena classes, and Bean classes are grouped in three different packages. Interfaces are grouped in the main package of the Java API for the input ontology, whereas Jena classes and Bean classes in its two different sub-packages. More details of the classes are provided in the following paragraphs.

5.3.1.5 Assigning Methods to Classes

A method is a collection of statements that perform a specific task and return result to the caller. Roughly speaking, within an Ontology Bundle, a method corresponds to an ontology property and aims at realizing the CRUD operations related to that property. The concrete methods manipulate the field of the class that correspond to the property. Methods are assigned to the classes that explicitly or implicitly fall into the domain or the range of the property. Therefore only objects belonging to a class

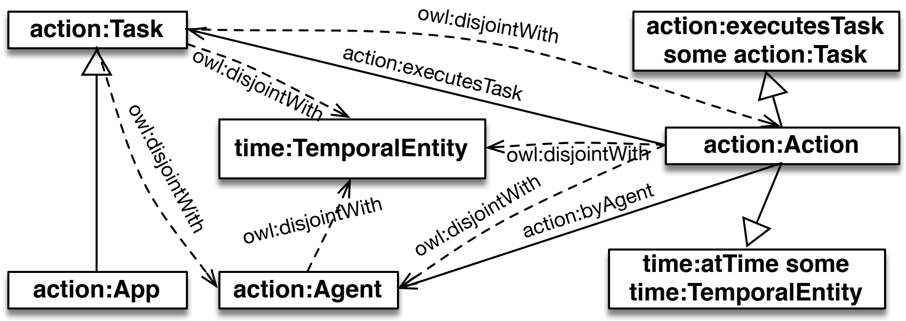


Figure 5.4: The Action ontology module of the Mario Ontology Network.

that explicitly or implicitly are in the domain or the range of a property are able to instantiate the property. For example, consider the *TaskExecution* Ontology Design Pattern[16] as implemented by the Action module of the Mario Ontology Network[17], depicted in Figure 5.4. The object property `action:byAgent` has `action:Action` as domain and `time:TemporalEntity` as range. According to the semantics of the ontology, having the property `action:byAgent` implies being an `action:Action`. It is worth noticing that the class `action:Action` is disjoint from all other classes in the ontology (i.e. `action:Task`, `action:Agent` and `time:TemporalEntity`), and, therefore entities belonging to the other classes cannot instantiate the property `action:byAgent` (otherwise the model would violate the ontology). Therefore, only objects belonging to the class *Action_Action* can have the field *action_byAgent*.

As a result, the interface *Action_Action* defines the following methods for manipulating the field corresponding to *action_byAgent*. For the sake of clarity of presentation, for each meethod's signature we present a SPARQL/SPARUL query implementing the method's semantics. It is worth noticing that the internal implementation of the API does not execute SPARQL/SPARUL queries but rely on Jena API instead.

- **public Set<Agent> getAction_byAgent()** returns all the agents that performed *this*[18] action. This method corresponds to the query shown in

[16]Task Execution ODP, http://ontologydesignpatterns.org/wiki/Submissions:TaskExecution
[17]MON's Action module, http://ontologydesignpatterns.org/ont/mario/action.owl
[18] *"This"* means "the object on which the method is invoked".

```
SELECT DISTINCT ?agent WHERE {
  ?this action:byAgent ?agent .
}
```

Frame 5.2: SPARQL query that corresponds to method *getAction_byAgent()*. The variable *?this* is bound to the IRI of the *this* object.

```
DELETE {?action action:byAgent ?agent .} WHERE {
  ?this action:byAgent ?agent .
}

INSERT DATA {
  ?this action:byAgent ?agent1 .
    ...
  ?this action:byAgent ?agentN.
}
```

Frame 5.3: SPARUL queries that correspond to method *setAction_byAgent(Set<Agent> agents)*. The variable *?this* is bound to the IRI of the *this* object, whereas the variables ?agent1,.., ?agentN are bound to the IRIs of the Agents passed as argument of the method

Frame 5.2.

- **public void setAction_byAgent(Set<Agent> agents)** sets the agents that perform *this* action as the set passed as parameter. This method corresponds to the queries shown in Frame 5.3.

- **public void addAllAction_byAgent(Set<Agent> agents)** adds the set passed as parameter to the agents of *this* action; This method corresponds to the queries shown in Frame 5.4.

- **public void removeAllAction_byAgent(Set<Agent> agents)** ensures

```
INSERT DATA {
  ?this action:byAgent ?agent1 .
  ...
  ?this action:byAgent ?agentN.
}
```

Frame 5.4: SPARUL query that corresponds to method *addAllAction_byAgent(Set<Agent> agents)*. The variable *?this* is bound to the IRI of the *this* object, whereas the variables ?agent1,.., ?agentN are bound to the IRIs of the Agents passed as argument of the method

```
DELETE DATA{
    ?this action:byAgent ?agent1 .
    ...
    ?this action:byAgent ?agentN .
}
```

Frame 5.5: SPARUL query that corresponds to method *setAction_byAgent(Set<Agent> agents)*. The variable *?this* is bound to the IRI of the *this* object, whereas the variables ?agent1,.., ?agentN are bound to the IRIs of the Agents passed as argument of the method

that the agents passed as parameters are no longer agents of *this* action. This method corresponds to the queries shown in Frame 5.5.

Moreover, the interface for the class `action:Action` implements two static methods to retrieve the actions having a certain property value for `action:byAgent`:

- **public static Set<Action> getByAction_byAgent()** which returns all the actions having the property `action:byAgent` instantiated. The body of the method generated by Lizard is provided in the Frame A.2 of the Appendix A.1. This method retrieves from the knowledge base all the triples having the property `action:byAgent` as predicate and the value passed as

```
SELECT DISTINCT ?action WHERE {
  ?action action:byAgent ?agent .
}
```

Frame 5.6: SPARQL query that corresponds to method *getByAction_byAgent()*.

```
SELECT DISTINCT ?agent WHERE {
  ?action action:byAgent ?agent .
}
```

Frame 5.7: SPARQL query that corresponds to method *getByAction_byAgent(LizardInterface value)*. The variable *?action* is bound to the IRI of the *value* object passed as parameter of the method.

parameter as object. Then, the method instantiates all the subjects of these triples as objects of the class *Action* and collect them into a set which is returned as result. This method corresponds to the query shown in Frame 5.6.

- **public static Set<Action> getByAction_byAgent(LizardInterface value)** which takes as input a *value* and returns a set of all the actions having that as property-value for `action:byAgent`. The body of the method generated by Lizard is provided in the Frame A.3 of the Appendix A.1. This method retrieves from the knowledge base all the triples having the property `action:byAgent` as predicate and the value passed as parameter as object. Then, the method instantiates all the subjects of these triples as objects of the class *Action* and collect them into a set which is returned as result. This method corresponds to the query shown in Frame 5.7.

For each property that a class can instantiate, the interface defines the signature of the above four methods and implements the two static methods.

It is worth noticing that the OWL language allows to define property restrictions on classes. A property restriction is an anonymous class that defines a condition to be

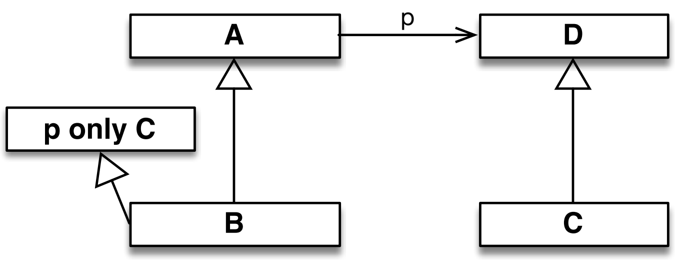


Figure 5.5: A simple ontology arising a name clash in the method signatures.

satisfied by all the individuals of the class. OWL distinguishes two kinds of property restrictions: value constraints and cardinality constraints. A *value constraint* puts constraints on the values of the property. For example, the range of the property `action:executesTask` for the individuals belonging to `action:Action` is restricted to `action:Task`. A *cardinality constraint* puts constraints on the number of values a property can take. For example, each individual of the class `action:Action` should have at least one value for the property `action:executesTask`. Lizard takes into account value restriction to infer the range of a property for a specific class. Suppose that the Action ontology module (cf. Figure 5.4) states that the range of the property `action:executesTask` is `owl:Thing` (instead of `action:Task`). The property restriction `action:executesTask some action:Task` restricts the range of this property when applied on `action:Action`. As a consequence, when Lizard generates the code corresponding to the class `action:Action`, it will consider the property `action:executesTask` having `action:Action` as domain and `action:-Task` as range. As the cardinality constraint is concerned, only existential constraint are taken into account by Lizard. Constraints on the maximum cardinality of a property are currently ignored. A proper management of this kind of property restrictions is left to future work.

However, property value restrictions may arise a name clash in the methods signatures. Consider the simple ontology depicted in Figure 5.5. This ontology defines four classes `A` which inherits from `B` and `C` which is a subclass of `D`. The ontology also defines an object property `p` with domain `A` and range `D`. For the individuals of `B`, the range of the property `p` is restricted to `C`. Therefore, for the property `p`, Lizard would generate for the interfaces A and B the following methods:

```
public interface A {
   ...
   public Set<D> getP();
   public void setP(Set<D> d);
   public void addAllP(Set<D> d);
   public void removeAllP(Set<D> d);
   ...
}
```

```
public interface B  extends A {
   ...
   public Set<C> getP();
   public void setP(Set<C> c);
   public void addAllP(Set<C> c);
   public void removeAllP(Set<C> c);
   ...
}
```

Since the Java compiler ignores the argument of the generic set (i.e. it ignores D in the interface A and C in the interface B), the signatures of the methods defined in A and B are equivalent for the compiler and the methods defined in B cannot override the methods in A. This situation causes a name clash. When this kind of errors occurs, Lizard addresses the problem by assigning the the methods arising the name clash (e.g. *setP*) only to the most generic interface (in this example A).

Other Static Methods Provided by an Interface. Each Java interface implements two static methods, namely *getId(String entityUri)* and *getAll()*. The body of the *getId* method for the interface *Action* is provided in the Frame A.1 of the Appendix A.1. This method takes as input the URI of an individual and returns as output an object of the class *Action*. The method checks in the knowledge base if the URI passed as parameter identifies an RDF individual that belongs to the class `action:Action`. If this is the case, it instantiates and returns an object of the class *Action*. The method *getAll()* retrieves from the knowledge base all the individuals belonging to the class `action:Action`. Then, for each individual the method instantiates an object of the *Action* and it returns all the instantiated objects in a collection.

5.3.1.6 Jena and Bean Classes

The aim of the Jena classes is to implement the binding between the Java objects and the data stored in the knowledge base. Jena classes have no fields and the methods directly manipulates values of the knowledge base. Jena classes extend the class *InMemoryLizardClass*, which in turn extends LizardClass. This class defines the field *individual* to keep a reference to the IRI which the object refers to and implements some general-purpose methods (e.g. for casting objects).

Bean classes aim at wrapping all the property values for an RDF individual in a single object. Objects of bean classes are mainly used by the REST API for building responses to REST calls. For each property assignable to the individuals of the corresponding Ontology class, Bean classes define a field, and all the interface methods related to the property (i.e. "get", "set", "removeAll", and "addAll"). A Bean class is shown in Frame A.12 of the Appendix A.3. Bean objects are instantiated by using the methods *asBean()* and *asMicroBean()* of the Jena classes. These two methods will be described later in this section.

Constructor. The constructor of the Jena class takes as input an IRI (or an object of the *RDFIndividual* which represents all the RDF individual stored in a knowledge base) and creates a new entity of the corresponding ontology class. This is done through the RDF API provided by Jena. The constructor of the class *Action_ActionJena* is shown in the Frame A.4 of the Appendix A.2. This constructor takes as input an IRI, for example `http://example.org/push`, stores the IRI in the variable individual of the super class (*LizardClass*),and asserts the following triple in the knowledge base:

```
<http://example.org/push> rdf:type action:Action .
```

Interface Methods. These methods (whose signature is defined by an interface implemented by the Jena class) aim at manipulating values corresponding to a property that objects can instantiate. These methods retrieve, remove, add, update RDF facts stored in the knowledge base by using the RDF API of Apache Jena. In the following the details of the methods implemented by the class *Action_ActionJena* for the property `action:byAgent`

- **public Set<Agent> getAction_byAgent()** (the code generated by Lizard is shown in the Frame A.5 of the Appendix A.2): this method retrieves from the knowledge base all the triples having the individual corresponding to *this* object as subject and `action:byAgent` as predicate. For each object retrieved from these triples, the method instantiates a new object of the class *Action_AgentJena*, and returns these objects as a collection of Agents.

- **public void setAction_byAgent(Set<Agent> agents)** (the code generated by Lizard is shown in the Frame A.6 of the Appendix A.2): this method removes from the knowledge base all the triples having the individual corresponding to *this* object as subject and `action:byAgent` as predicate. Then, for each *Agent a* passed as parameter, the method add to the knowledge a triple that having *(i)* the individual corresponding to *this* object as subject; *(ii)* `action:byAgent` as predicate; *(iii)* the individual corresponding to *a* as object.

- **public void addAllAction_byAgent(Set<Agent> agents)** (the code generated by Lizard is shown in the Frame A.7 of the Appendix A.2): for each *Agent a* passed as parameter, the method add to the knowledge a triple having *(i)* the individual corresponding to *this* object as subject; *(ii)* `action:byAgent` as predicate; *(iii)* the individual corresponding to *a* as object.

- **public void removeAllAction_byAgent(Set<Agent> agents)** (the code generated by Lizard is shown in the Frame A.8 of the Appendix A.2): this method removes from the knowledge base all the triples having the individual corresponding to *this* object as subject and `action:byAgent` as predicate.

asBean() and asMicroBean() Methods. Objects of the Bean classes aim at enclosing all the property values retrieved from the for the knowledge base for the individual the Jena object refers to. The method *asBean()* instantiates an object of the Bean class implementing the same interface of the Jena class. The method executes a SPARQL Describe query on the knowledge base to retrieve the data about the individual *this* object refers to. The result of the query is then parsed by some private service methods of the Jena class. The Jena class provide one private method each property of the object. These methods implement the same logic of the *get* methods (i.e. they return the values that the individual has for the property). An example of asBean() method is shown in Frame A.9 of the Appendix A.2, whereas an example of private service method for the property `action:byAgent` is shown in Frame A.11. By using the private service methods (instead of the Jena methods), we reduce the number of queries executed on the knowledge base for creating a bean object. If the asBean() method had used the jena *get* methods, then it would have been executing one query for each property of the object (instead of one describe query). Once retrieved the values from the service methods, asBean() fills the fields of the bean object. In case of object properties, the fields are filled with Jena objects. This allows the caller of asBean() to retrieve all the data associated to object property values.

As we will see in the Section 5.3.2, the bean objects are serialized as JSON objects and returned as results of the REST API calls. The serialization in JSON object retrieves all the property values from the bean object. If a property refers to another object, then the serialization process will also serialize the referred object. It is easy to see that this process could be infinite (it stops if the object to serialize has only datatypes as property values). To overcome this issue, we devised the asMicroBean() method. This method makes sure that the caller of asMicroBean() is able to retrieve only the values related to *this* object. This is done by filling the fields corresponding to object properties with bean objects (instead of jena objects). Frame A.10 of the Appendix A.2 provides an example of asMicroBean() method. By checking the field *isCompleted* of the Bean objects, the caller will be able to distinguish bean objects for which all the values are already retrieved from the knowledge base the objects that are incomplete.

5.3.2 Rest API

Lizard provides a Web Service that exposes Java API by following the RESTful architectural style over HTTP. The Web Service is realized by the Lizard Common's class *RestImpl* which implements a Web resource that produces and consumes data in JSON format. All the functionalities provided by Java API can be invoked through this Web Service. In particular, client applications can invoke the methods of the Java API through different paths. The path and the parameters of request indicates which ontology, class and individual the client wants to operate on and the operation (e.g. "set" a property) that has to be invoked. The *RestImpl* class takes advantage of the Java Reflection mechanism to invoke the right methods for each possible path. In this section we overview the implementation of the Web Service.

Mapping OWL on REST Architecture. A REST architecture is composed of four archetypes. *(i)* A *document* resource is the *base archetype* of REST architecture. It is an entity that includes fields with values and links to other documents. It is akin to RDF named individuals and Java Objects. *(ii)* A *collection* is a *server--managed directory* of resources. Clients may purpose to add a new item to the collection, but it is up the collection to decide to create the new resource or not and the URI of the new resource. A *store* is a *client-managed directory* of resources. Stores let the clients decide when create, updated or delete resources and do not generate the URI of the resources. Both collections and stores are suitable for classes and properties. However, not allowing clients to decide the URI would avoid the possibility of storing facts on existing RDF named individuals. Hence we use stores for classes and resources. *(iii)* A *controller* is procedure that takes as input values and returns values. We use controllers for mapping static methods of interfaces like *getByProperty*.

URI Design. URI is composed by static (i.e. fixed names chosen by the designer) and variable segments (i.e. filled with some identifier). In the following we enclose variable segments of URIs within braces. The best practices for designing RESTful architecture[139] suggest (among others) *(i)* to use singular nouns for naming documents and plural nouns for naming collections and stores; *(ii)* to hierarchically organize path for resources; *(iii)* to not use function names; *(iv)* to use verbs for naming controllers; *(v)* to use variable segments for identity based values. These rules result paths like:

```
/leagues/{leagueId}/teams/{teamId}/players/{playerId}
```

However, these standards rules cannot be fully applied to design the URIs for the REST API due to the following reasons: *(i)* The names of the stores (i.e. classes and properties) are available only at running time (i.e. when the Ontology Bundle is generated). *(ii)* Varying the name from singular to plural could confuse the clients that know how the concepts are named in the ontology. *(iii)* Semantic Web technologies are based on the assumption that things (i.e. individuals, classes and properties) are denoted by IRIs. It would be desirable naming segments with IRIs but it is not allowed to use all the IRI characters (e.g. “:”) in the path. Therefore, we identify classes and properties by the abbreviated notation (i.e. using the prefix of the namespace) where the colon is replaced with an underscore. Since named individuals may have any prefix, the abbreviated notation cannot be used. We identify individuals with query parameters (that can be passed as payload of HTTP requests).

Ontology and Description of the API. The first part of the path indicates the ontology the client wants to operate on. All the functionalities realized by the Java API for the ontology are grouped under this path. There are no operations provided for this path only. A client application can retrieve the description (provided in OpenAPI language) of the operations available for the ontology by issuing an HTTP GET at the path:

```
/{ontology}/swagger.json
```

Instantiating New Individuals. A client can instantiate new individuals of a class by issuing a POST request at the following path:

`/{ontology}/{class}`

The POST method requires a parameter, called "iri", that indicates the IRI of the RDF individual the client wants to instantiate. Once received the request, the *RestImpl* class identifies the name of the Jena class in charge of instantiating the individual and uses the Java reflection API to invoke the constructor of the Jena class. Suppose that a client issues the following POST request:

```
POST /org_ontologydesignpatterns_ont_mario_person_owl/person_person
Content-Type: application/json
{ ``iri'':``http://example.org/luigi'' }
```

The class *RestImpl* converts the path of the POST request `/org_ontologydesignpatterns_ont_mario_person_owl/person_person/` in the name of the corresponding Jena class *org.ontologydesignpatterns.ont.mario.person_owl.jena.PersonJena*. The Java reflection API allows to invoke the constructor of the Jena class.

Retrieving Individuals of a Class. A client can retrieve the individuals belonging to a class by issuing a GET request at the following path

`/{ontology}/{class}`

This request (without parameters) invokes the *getAll* method defined in the corresponding Java interfaced generated by Lizard (cf. Section 5.3.1.5). This method returns a collection of Jena objects which are transformed in Bean objects (by invoking the method *asMicroBean()*) and returned to the caller. A client can retrieve an individual of a class by providing its IRI as parameter of the request:

```
GET /org_ontologydesignpatterns_ont_mario_person_owl/person_person
Content-Type: application/json
{ ``iri'':``http://example.org/luigi'' }
```

This request returns the JSON serialization of the Bean object corresponding to IRI passed as parameter. The Bean object is retrieved by invoking the *get* method defined in the interface that corresponds to the ontology class. If the *get* method returns an object (which is a Jena object), then it is transformed in a Bean object by invoking *asMicroBean()*. It is worth noticing that client applications can retrieve all the individuals stored in the knowledge by activating the inference engine and issuing a GET request.

Retrieving Individuals of a Class Having a Property. RDF named individuals stored in the knowledge base which have a given property can be retrieved by issuing a GET request at the following path:

```
/{ontology}/{class}/having/{property}
```

The request returns all the individuals having a certain property which belong to a certain class, but if the client activates the inference engine and chooses the class `owl:Thing` it can retrieve all the individuals having the property. The result set is built by transforming in Bean objects the collection of Jena objects returned by the *getBy* method of the Interface that corresponds to the ontology class. Clients can furtherly restrict the result of this operation by requesting the individuals that have a specific value for a property. This is done by passing as parameter the value of the property which can be either an IRI or a literal. The following request returns all the persons having "Luigi" as first name.

```
GET /org_ontologydesignpatterns_ont_mario_person_owl/person_person/
having/Person_firstName
Content-Type: application/json
{ ‘‘value’’:‘‘Luigi’’ }
```

Managing Properties. Client applications are able to manage property values assigned to RDF individuals by issuing GET, POST, PUT and DELETE requests at the path

/{`ontology`}/{`class`}/{property}

The individual the client wants to operate on and, possibly, the property values it wants to add/remove/update are provided as parameters of the request. In particular:

GET. The values that an IRI has for a property `p` can be retrieved by providing the IRI as parameter of a GET request. To accomplish this request, the *RestImpl* class retrieves the object (that corresponds to the IRI) from the knowledge base, it invokes its *getP()* methods, and, finally returns the result. If the `p` is an object property the result objects are transformed in Bean objects. For example, the following GET request retrieves the first name of `http://example.org/luigi`

```
GET /org_ontologydesignpatterns_ont_mario_person_owl/person_person/
having/Person_firstName
Content-Type: application/json
{ ‘‘iri’’:‘‘http://example.org/luigi’’ }
```

PUT. The PUT method enables client applications to set the values that an individual has for a property `p`. This method requires as input an IRI and the values that client wants to set. This request is fulfilled by retrieving from the knowledge base the object (that corresponds to the IRI) and by invoking the method *setP*. The following request set the first name of `http://example.org/luigi` as the literal “Luigi”

```
PUT /org_ontologydesignpatterns_ont_mario_person_owl/person_person/
having/Person_firstName
Content-Type: application/json
{ ‘‘iri’’:‘‘http://example.org/luigi’’,
  ‘‘value’’: ‘‘Luigi’’}
```

POST. The POST method adds the values passed as parameter to set of values that an individual has for a property `p`. This method requires as input an IRI and the values that client wants to add. The *RestImpl* class retrieves from the knowledge base the object (that corresponds to the IRI) and invokes the method *addAllP*. The following request add an additional first name (i.e. “Maria”) to the individual `http://example.org/luigi`

```
POST /org_ontologydesignpatterns_ont_mario_person_owl/person_person/
having/Person_firstName
Content-Type: application/json
{ ‘‘iri’’:‘‘http://example.org/luigi’’,
‘‘value’’: ‘‘Maria’’}
```

DELETE. The DELETE method removes the values passed as parameter from the set of values that an individual has for a property `p`. This method requires as input an IRI and the values that client wants to remove. The *RestImpl* class retrieves from the knowledge base the object (that corresponds to the IRI) and invokes the method *removeAllP*. The following request removes “Maria” from the first names of the individual `http://example.org/luigi`

```
DELETE /org_ontologydesignpatterns_ont_mario_person_owl/person_person/
having/Person_firstName
Content-Type: application/json
{ ‘‘iri’’:‘‘http://example.org/luigi’’,
‘‘value’’: ‘‘Maria’’}
```

5.3.2.1 Description of the REST API

Lizard generates a description (in OpenAPI language) of all operations made available by the REST API. Intuitively, this description aims at specifying all possible instantiations of the REST API allowed by *RestImpl* class that are compliant with the ontology. For example, according to the Action ontology module of the Mario Ontology Network, `action:executesTask` and `action:byAgent` are assignable to individuals that belong to `action:Action`. The Java Interface for `action:Action` defines the corresponding methods and the Jena and Bean classes provides the implementation. The REST API description for the Action ontology will specify for `action:Action` class and for these two properties the following paths:

1. `/org_ontologydesignpatterns_ont_mario_action_owl/action_action` that accepts GET and POST HTTP methods;

2. `/org_ontologydesignpatterns_ont_mario_action_owl/action_action/action_byAgent` that accepts GET, PUT, POST and DELETE HTTP methods;

3. `/org_ontologydesignpatterns_ont_mario_action_owl/action_action/having/action_byAgent` that accepts GET method;

4. `/org_ontologydesignpatterns_ont_mario_action_owl/action_action/action_executesTask` that accepts GET, PUT, POST and DELETE HTTP methods;

5. `/org_ontologydesignpatterns_ont_mario_action_owl/action_action/having/action_executesTask` that accepts GET method.

An excerpt of the REST API description generated by Lizard for the Action ontology is provided in Frame A.13 of the Appendix A.4. The description also specifies the content and the status code (e.g. 200, 404 etc.) of the responses provided by the REST API. The content specifies the structure of the objects returned from a REST call and it complies with the Bean objects of the Java API.

Applications that want to interact with the Knowledge Base can use Swagger codegen[19] to generate a client for the REST API. Swagger codegen takes as input the description (in OpenAPI language) of the REST API and provides as output a library implementing the client for the REST API that can be included into client applications that want to interact with the Knowledge Base. Swagger codegen is able to generate REST API clients for forty programming languages.

[19]Swagger codegen, https://swagger.io/tools/swagger-codegen/

5.4 Discussion

This Chapter presented Lizard, a framework which is aimed at easing the software development of knowledge-aware systems by filling the gap between Semantic Web technologies and object-oriented applications. Lizard allows application running on robotic frameworks to access RDF facts stored in the knowledge base in a programmatic way (cf. RQ3 Section 1.1). Given an OWL ontology as input, Lizard is able to generate an API for accessing RDF triples stored in a triple store without dealing with Semantic Web models and languages. Section 7.4.1 will present two knowledge intensive applications that rely on Lizard for interacting with the robot's knowledge base. These applications demonstrate Lizard's benefits, feasibility and limitations when developed, deployed and tested in a real socially assistive scenario.

Besides the lack of integration with the W3C's Linked Data Platform specification[20] which we plan to address in the next release, a current limitation of Lizard is the poor interaction paradigm offered to applications. Lizard, like other tools for programmatically accessing knowledge bases, enables a triple-based interaction with triple stores, i.e. generated Java methods deal with a single triple at time. This is strongly limiting compared to querying a knowledge base with the possibility of joining several triple patterns in a single SPARQL query. A valuable direction for improving usability of the API and interaction with the knowledge base could be enabling a pattern-based interaction. For example, static methods could be generated by Lizard to instantiate an ontology pattern by invoking a single method. In this direction, ontology modularization research field [54, 55, 189] could provide Lizard with techniques to identify patterns in input ontologies.

[20]https://www.w3.org/TR/ldp/

Chapter 6

A Frame-based Approach for Integrating Ontologies

Social robots have to deal with a wide range of heterogeneous data coming from several sources such as data extracted from users' speech, sensors, web etc. Supposing that the syntax of this data is homogeneous (which is not always the case), then the data could be expressed with a semantics either known by the robot or not. In the former case, robots are able to process, ingest and react to input data, whereas in the other they cannot unless semantic heterogeneity of data is addressed. This Chapter proposes an approach for addressing semantic heterogeneity of data processed by robots (cf. Section 1.1 RQ4) formalized as an ontology matching task. Ontologies are artifacts encoding a description of a domain of interest for some purpose. Ontologies can be defined by different people and can vary in quality, expressiveness, richness, and coverage, hence increasing semantic heterogeneity of the resources made available through the Linked Open Data. Among the various semantic technology proposed to handle heterogeneity, Ontology Matching [191] has proved to be an effective solution to automate integration of distributed information sources. Ontology Matching (OM) finds correspondences between semantically related entities of ontologies. However, most of the current ontology matching solutions present two main limits: *(i)* they only partially exploit the natural language descriptions of ontology entities and lexical resources as background knowledge; *(ii)* they are mostly unable to find correspondences between entities specified through different logical types (e.g. mapping properties to classes). We argue that using lexical resources, such as linguistic frames, as background knowledge for matching ontology entities may lead to a step ahead in the state of the art of ontology matching.

Frame Semantics [75] is a formal theory of meaning based on the idea that human can better understand the meaning of a single word by knowing the relational knowledge associated to that word. For example, the meaning of the verb *buy* can be clarified by knowing that it is used in a situation of a commercial transfer which involves individuals playing specific roles, e.g. a buyer, a seller, goods, money and so on. In other words, the verb buy *evokes* a scene where there are some individuals are playing specific roles. Our hypothesis is that the frames evoked by words associated with an ontological entity can be used to derive the intended meaning of that entity thus facilitating the ontology matching task.

In this Chapter we introduce a novel approach aimed at finding correspondences between ontology entities according to the intensional meaning of their models, hence abstracting from their logical types. This strategy allows us to match ontological entities with respect to their intensional meaning (that we suppose is evoked by the textual annotations associated with them) instead of their axiomatization, hence to abstract from their logical type. In fact, the axiomatization could have been forced by the choice of certain language for specifying the ontology, by the personal modeling style of the designer, or, other requirements (e.g. the compatibility with an existing ontology) unrelated to the modeled domain. This method is aimed at providing robots with the means for automatically integrating knowledge coming from heterogeneous sources.

The proposed approach is not intended to replace existing OM solutions relying on the logical specification of ontology entities (e.g. [107]), on the contrary it can be used in combination with other logic-based techniques. For instance, the strategy proposed in [126] can be used for combining our method (which focuses on the lexicon related to entities) with other logic-based techniques (e.g. [107]). We argue that this approach may lead to a step ahead in the state of the art of ontology matching, and positively affect related applications such as question answering and knowledge reconciliation, ontology population and language generation.

6.1 Types of Semantic Heterogeneity

Klein [113] provides a classification of the types of heterogeneity between ontologies. A first distinction is between mismatches at language level and at ontology level. The languages of two ontologies can differ in their syntax, or, in the primitives that are used to specify an ontology. OWL is the standard language for encoding ontologies in the Semantic Web context. However, OWL constructs forces ontology designer to convey to some logical patterns. For instance, OWL does not provide a construct for defining n-ary relations. When there is the need of defining n-ary relations, a common pattern is to represent it by means of OWL classes. The second level of mismatches is the ontology-level. A very useful distinction of ontology-level mismatches is made by [164] who distinguished the conceptualization of ontologies from their explication. A *conceptualization* mismatch is a difference in the interpretation of a certain domain. This mismatch leads to define different ontological concepts or different relations among concepts (i.e. the two ontologies present different coverage, granularity or scope). An *explication* mismatch is a difference in the specification of a certain conceptualization (e.g. using a o relation instead of a class). A mismatch of this type can be caused by differences in: *(i)* the choices of the modeler about the style of modeling (e.g. using a datatype property instead of an object property); *(ii)* the adopted terminology for naming concepts (e.g. using synonyms for representing the same concept); *(iii)* local requirements of the ontology project, for instance the request of using a certain language for specifying the ontology could lead the modeler to some choices (e.g. representing n-ary relations in OWL as classes) instead of others (e.g. PURO metamodel [202] does not have the arity of relationships limited to two).

6.2 Proposed Approach

Following [89], we devised an approach for ontology matching that considers frames as "*unit of meaning*" for ontologies and exploits them as a mean for representing the intensional meaning of the entities. Our strategy consists of two steps, summarized as follows. First, we create a mapping between input ontologies and frames (see section 6.2.1)[1]. In the second step we use the mapping ontologies-frames to find correspondences among entities defined in the input ontologies (see section 6.2.2).

6.2.1 Mapping Ontology Entities on Frames

The first step of our strategy associates each ontology entity with one or more frames representing its intensional meaning. Besides representing the intension of an entity, frames also provide contextual information relevant for the described concept that can be exploited for the comparison with other entities (see 6.2.2).

Selecting frames evoked by annotations. In order to associate ontological entities with frames we analyze the textual annotation associated with them. In fact, annotations provide humans with insights of the intensional meaning of a certain entity. The main idea of this approach is that words used in the annotations *evoke* frames that are representative of the intensional meaning of the entity.

An ontological entity can be associated with three textual annotation: *(i)* an *identifier* (e.g. `rdf:ID`) which is not a proper annotation since it has been originally thought for machines, but, in order to improve the readability of RDF data is often meaningful; *(ii)* a *label* (e.g. `rdfs:label`), a short text content used for naming the entity; *(iii)* a *comment* (e.g. `rdfs:comment`) which is a description of the resource in natural language, often providing examples of the concept being defined. However, comments often contain words that are not directly connected with the intension of the entity (e.g., comments often use the verb "*represent*" which is not always

[1]This step can be seen as ontology matching as well. In fact, it aims at mapping a frame (which is an ontology since it provides a conceptualization of a certain situation) and an ontology.

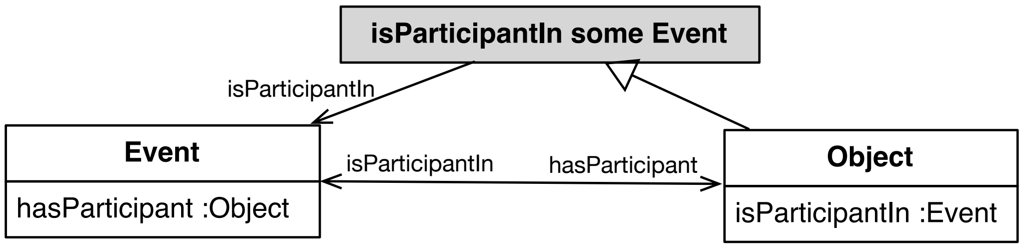


Figure 6.1: The UML class diagram of the Ontology Design Pattern *Participation*.

the most appropriate term to characterize an entity). Therefore we only consider *identifier* and *label* as characterizing annotations of an entity. Our hypothesis is that frames evoked by words contained in these annotations provide a model for the intensional meaning of the entity.

In associating entity with frames, the ambiguity of words has to be taken into account. For instance, depending on the word *"bind"* may evoke either the frame *Imposing obligation*[2] (when it is intended as "*bind by an obligation*") or the frame *Becoming attached*[3] (when it is intended as "*wrap around with something so as to cover or enclose*"). Therefore, to associate entities with the most appropriate frames, we have *(i)* to disambiguate the sense of the word in the text characterizing entities; *(ii)* and then, to select frames evoked by the sense of the words (e.g. by exploiting Framester's WordNet-FrameNet mapping [81]).

Figure 6.1 shows the UML class diagram of the Ontology Design Pattern *Participation*[4]. The ontology defines four entities, two classes and two object properties connecting them. Table 6.1 shows an example of association of the entities defined by the ODP Participation with the FrameNet's frames. We used UKB [1] for WSD and the mapping WordNet to FrameNet provided by Framester [81]. Due to the vagueness of the terms "Object" and "Event", the WSD confidence is quite low and the association of `Object` and `Event` with frames (*Popularity*, *Communication* etc.) is weak as well. Conversely, the WSD confidence associating `isParticipantIn` (or

[2]https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Imposing_obligation.xml
[3]https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Becoming_attached.xml
[4]http://ontologydesignpatterns.org/wiki/Submissions:Participation

Ontology Entity	**Associated Sense**	**WSD Confidence**	**Evoked Frame**
`isParticipantIn`	wn31:110459618-n	0.5226	Competition
`isParticipantIn`	wn31:110459618-n	0.5226	Participation
`isParticipantIn`	wn31:110459618-n	0.5226	People
`isParticipantIn`	wn31:302340196-a	0.150614	Popularity
`hasParticipant`	wn31:110459618-n	0.522534	Competition
`hasParticipant`	wn31:110459618-n	0.522534	Participation
`hasParticipant`	wn31:110459618-n	0.522534	Popularity
`Object`	wn31:106321227-n	0.193556	Communication
`Object`	wn31:200809123-v	0.151107	Assessing
`Object`	wn31:200809123-v	0.151107	Cogitation
`Object`	wn31:106142175-n	0.0966854	Artifact
`Object`	wn31:106142175-n	0.0966854	Fields
`Event`	wn31:111430739-n	0.305213	Being_in_effect
`Event`	wn31:111430739-n	0.305213	Causation
`Event`	wn31:111430739-n	0.305213	Objective_influence
`Event`	wn31:111430739-n	0.305213	Subjective_influence
`Event`	wn31:113966452-n	0.242894	State_of_entity

Table 6.1: An example of association ontology entity-frames.

`hasParticipant`) is rather high. It is easy to see that the frames *Competition*[5] and *Participation*[6] are somehow connected to the meaning of `isParticipantIn` (or `hasParticipant`). The frames *People* and *Popularity* are also related[7] to the sense of the object properties, but they can not be mapped to the object properties (as we can see in the next steps, the confidence of the mapping is too low).

Mapping Ontology Entities on Frames. At this point ontology entities are associated with frames that are somehow related (i.e., evoked) to their intensional

[5]https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Competition.xml

[6]https://framenet2.icsi.berkeley.edu/fnReports/data/frame/Participation.xml

[7]The sense assigned to "*participant*" is "*a person who participates in or is skilled at some game*" which is a particular case of the meaning represented by the object properties `isParticipantIn`.

meaning, now an effective mapping between them has to be created. An example of mapping is provided by FrameBase's integration rules [184]. However, they focused on the transformation of object properties (called, *binary predicates*) in binary projection of frames, and classes in their valences. These assumptions are too restrictive. For instance, Nuzzolese et al. [153] used object properties for representing *frame elements*. The choice of certain ontological type for representing a concept depends on requirements that are external from the domain that is being represented. Therefore, we claim that the mapping ontologies-frames has to be done without assuming any fixed correspondence between the ontological types of the two models (e.g. without assuming that object properties always correspond to binary projections of frames).

In order to identify the effective mapping between ontologies and frames, we go through ontology entities and for each entity we compute all possible mappings between entities and frames selected in the previous step (i.e. those evoked by its annotations). In frame semantics, a frame is characterized by its roles (also called frame elements) and each element possibly define the semantic type of the individual that can play that role in the frame. Frames, frame elements and semantic types have a name and a description. For each ontology entity, we compute the similarity of the entity with the evoked frames, its elements, and its semantic types. Therefore an ontology entity may correspond to one of these components defined in the evoked frames. The similarity between a frame component and an ontology can be estimated by computing the semantic text similarity of the descriptions of the two elements. The semantic text similarity can be computed by using NLP tools such as ADW [167].

For example, Consider the object property `isParticipantIn` having `Object` as domain and `Event` as range. Its associated text evokes four frames: *Competition*, *Participation*, *People* and *Popularity*. We start considering *Competition* and we compute any possible mapping between the object property and the frame. The core elements of *Competition* are *Competition* and *Participant*. All possible correspondences between elements of the two models are shown in Figure 6.2. A dashed line represents a possible correspondence between elements of the two models and its label is a confidence measure of the Semantic Text Similarity (STS) between the comments of the two elements. It is easy to see that the top-scoring alignment is that which maps the object property `isParticipantIn` to the frame element *Participant* and the class `Event` to the frame `Competition`. The correspondences involving the class `Object` have not totalised an high score, therefore `Object` is not mapped by means of STS. With respect to the frame *Competition*, `Object` would represent the semantic type of the frame element *Participant*. However, FrameNet does not define any semantic type for *Participant*. Similarly, the frame element *Competition* is used "*for the name of the competition*", therefore is not mapped to any element of the ODP since it does not define any similar element.

Actually, some clues emerged that could be used for mapping `Object`. The confidence of Semantic Text Similarity is high enough to consider `Event` and `isParticipantIn` mapped to respectively the frame *Competition* and its frame element *Participant*. Furthermore, we can notice that `isParticipantIn` is defined in the in the ODP *Participation* as an object property connecting `Object` and `Event`. The object property `isParticipantIn` (which corresponds to the frame element *Participant*) connects a class (i.e. `Event`) that is aligned to the frame *Competition* to another class (i.e. `Object`). Therefore, `Object` can be reasonably treated as semantic type of *Participant*. It is worth noting that only at this point we use the ontological type of the entities.

In conclusion, the mapping holding between the ODP *Participation* and the frame *Competition* is the following:

1. The class `Event` is aligned to the frame *Competition*;

2. The class `Object` corresponds to the semantic type of the frame element *Participant*;

3. The object property `isParticipantIn` matches to the frame element *Participant*.

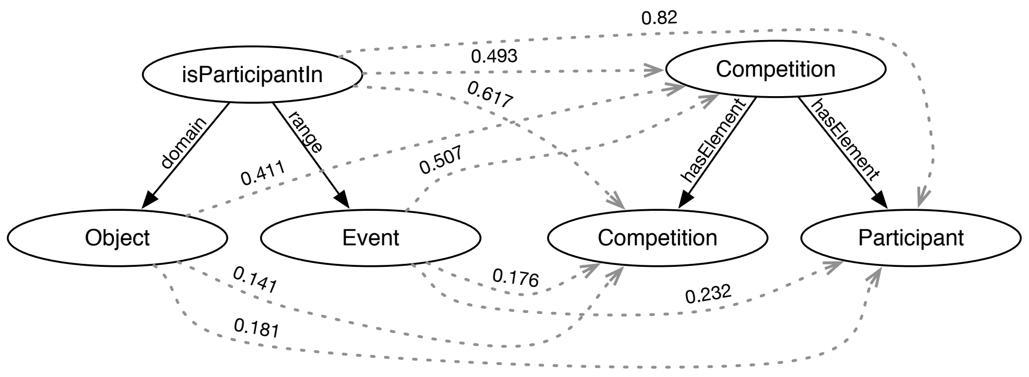


Figure 6.2: An example of alignment between the object property `isParticipantIn` and the frame *Competition*. A dashed line represents a possible correspondence between elements of the two models. These edges are labeled with a confidence measure based on the semantic text similarity of the two elements.

```
:footballTournament a participation:Event;
  rdfs:label "footbal tournament''.

:Jo a participation:Object;
  rdfs:label "Jo'';
  participant:isParticipantIn :footballTournament .
```

Frame 6.1: An example of usage of the mapping of the frame *Competition* on the ODP *Participation*

For instance, the frame occurrence `Competition(football tournament, Jo)`[8] can be stored by means of the ODP Participation as shown in the Frame 6.1.

The described procedure for mapping the isParticipantIn object property to a Frame (or its entities) has to be applied to every element of the source ontology and the frames it evokes. The result will be a series of mapping between the ontology and evoked frames. Each mapping will have a certain confidence computed by summing the score of the word sense disambiguation and the score of semantic text similarity.

[8]The frame occurrence can be extracted from the sentence "Jo played in the football tournament.".

A threshold can be set to discard the mapping with low confidence.

6.2.2 Frame-based Ontology Matching

Once input ontologies and frames are aligned, each ontology entity is associated with a formal specification of its intensional meaning (that we call *frame-based specification*). As pointed out in [180] the properties *subclass of* and *sub-property of* are not enough to explicit complex relation between entities. In light of this consideration we express the relation between frames and ontology entities by interpreting both as *predicates*. A formalization of frames as *multigrade predicates* is provided by [81]. A straightforward interpretation of ontology entities as predicates represents classes as n-ary predicates (the arguments of the n-ary predicate are the entities in its neighborhood) and properties as binary predicates. For instance, the class `TimeIndexedPartipation`[9] can be represented as a ternary predicate with arguments provided by `Event`, `TemporalEntity` and `Object`. Interpreting frames and ontology entities in predicates allows us to express complex relationship which cannot be formalized by only using OWL/RDFs vocabularies. Framester ontology [81] defines a set relationship holding between predicates. Using the Framester vocabulary the class `TimeIndexedPartipation` can be specified as `projectionOf` the frame *Participation*, with members `involveEvent`, `atTime` and `includesObject` (which can be interpreted as subroles of *Event*, *Time* and *Participant*). Also the property `isParticipantIn` of the ODP *Participation* can be specified as `projectionOf` the frame *Participation*, with members `Object` and `Event`. Therefore, the class `TimeIndexedParticipation` and the object property `isParticipantIn` are "aligned" to the same frame and a complex correspondence between `TimeIndexedParticipation` and `isParticipantIn` can be derived. In this case `isParticipantIn` is a `subframeOf TimeIndexedParticipation`. The subframe relation might be used for creating a `CONSTRUCT` SPARQL query or an inference rule[10] transforming instances of the class in instances of the property. Fig-

[9]Time Indexed Participation ODP https://goo.gl/qX3DDr

[10]Refer to [184] for examples of these kinds of rules.

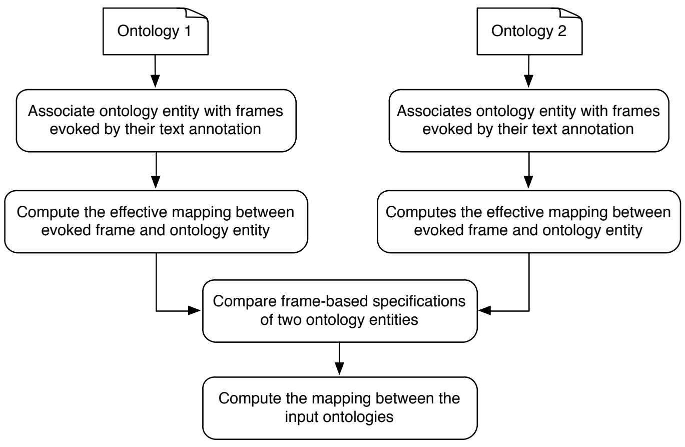

Ontology 1
Ontology 2
Associate ontology entity with frames evoked by their text annotation
Associates ontology entity with frames evoked by their text annotation
Compute the effective mapping between evoked frame and ontology entity
Computes the effective mapping between evoked frame and ontology entity
Compare frame-based specifications of two ontology entities
Compute the mapping between the input ontologies

Figure 6.3: The workflow summarizing the macro steps of the proposed approach for matching two ontologies.

ure 6.3 summarizes the proposed approach for matching two ontologies. Firstly, each entity of the input ontologies is associated with frames evoked by its text annotation, and, then with a frame-based specification of its meaning (steps 1 and 2). Subsequently, frame-based specification of ontology entities of the two input ontologies are compared (step 3). Finally, step 4 computes the mapping between the input ontologies.

6.3 Discussion

In this Chapter we introduced a novel approach for ontology matching. This method exploits the frame semantics as cognitive model for representing the intensional meaning of ontology entities. The frame-based representation enabled at finding complex correspondences between ontology entities abstracting from their logical type thus leading a step ahead the state of the art of ontology matching.

Aligning ontologies with frames, in particular linguistic frames, has a second non-negligible benefit related to both understanding and generation of natural language. An increasing number of NLP frameworks is adopting linguistic frames as “preferred vocabulary” for extracting structured knowledge from text. Examples of such tools are FRED [90], KNEWS [22], Google’s SLING [179] and Open-SESAME [203].

These tools, combined with an alignment mapping linguistic frames on robot's knowledge base schema, enables robot to directly ingest and integrate structured knowledge extracted from text. As far as natural language generation is concerned, it is worth noticing that a linguistic frame (such those defined in FrameNet) often comes with annotated sentences verbalizing some of its instances. For example the Frame "Buy" is accompanied by the sentence "Luigi bought from Mary a car for 4000 euro" which corresponds to the instance BUY(Buyer: Luigi, Seller: Mary, Money: 4000 euro, Goods: a car). This feature of linguistic frame repositories offers the unique opportunity of having instances of structured knowledge (i.e. a frame schema and its instances) with a set of corresponding verbalizations. These verablizations could be used to extract generalized templates for verbalizing robot's structured knowledge (e.g. `Buyer` bought from `Seller` `Goods` for `Money`). In conclusion, we claim that aligning linguistic frames with ontologies could be the key for filling the gap between structured and unstructured knowledge.

The proposed approach is being implemented and evaluated, therefore we cannot show results that directly demonstrate the feasibility of the proposed approach. But, good results on classification linked data entities with respect to foundational distinctions by using labels and descriptions of entities demonstrate that textual annotations suggest the semantics of entities (cf. Section 4.2). Therefore, we hypothesize that textual annotations in ontology found on the Web are rich enough to be used for the frame ontology alignment task.

We plan to evaluate the frame-ontology alignments in a both *direct* and *indirect* way. The benchmarks used for assessing ontology matching systems are mostly unable to evaluate the capability of finding correspondences among ontology entities with different logical types. In order to accomplish this purpose we are extending the existing benchmarks for ontology matching. An example in this direction is [214]. On the other hand, we are using the proposed approach in a question answering system for selecting relevant resources answering a given question. The frame occurrences in a question together with the frame-ontology alignment help in formulate the query over the linked data, hence identifying resources that answer the given question.

Chapter 7

A Knowledge Base Centered Software Architecture for Social Robots

This chapter presents a component-based architecture relying on semantic web technologies for supporting knowledge-intensive tasks performed by social robots. The design of the architecture has been guided by requirements coming from a real socially assistive robotic application (presented in Section 1.3). The aim of the architecture is orchestrating semantic technologies (cf. Section 1.1 RQ5), leveraging on contributions of this thesis and state-of-the-art tools, with the ultimate goal of creating a robotic platform for *(i)* easing *customizability* and *extensibility* of robot's behavior and its social skills; *(ii)* improving both inner (among architectural components) and outer (with external entities) *interoperability*; *(iii)* enabling a *rapid prototyping* of robotic applications; *(iv)* enhancing *reusability* of architectural components. The contributions presented in the previous chapters are included in this architecture in order to provide developers with functionalities for structuring and accessing robot's knowledge, exploiting Linked Open Data, and integrating input data with the existing knowledge. The core of the architecture is the knowledge base. Robot behaviors can be developed as pluggable applications operating on top of the software architecture. The robot behaviors, as well as other components, contribute to and benefit from the knowledge base. For example, on the one hand, the robot behaviors acquire and store knowledge such as user's personal data, events, environmental data etc. On the other hand, the robot behaviors use the acquired knowledge to perform the tasks and the execution of the behaviors is influenced by information stored in the knowledge base.

A prototype of this architecture has been developed within the context of the MARIO project. This prototype follows principles and design defined in this chapter and gives the possibility of evaluating feasibility and benefits of the software architecture in a real social assistive context.

This chapter is structured as follows. Section 7.1 defines the functional and non-functional requirements that lead the development of this architecture. Section 7.2 provides an overview of the architecture. The components constituting the software architecture are described in Section 7.3. Finally, architecture prototype and its evaluation are presented in Section 7.4.

7.1 Requirements of Software Architectures for Social Robots

In order to introduce the software architecture we need to introduce the requirements of a social robots. In this section we emphasize on requirements that aim at increasing the acceptability of the robot and facilitating the development of robotic applications that manage the interaction with humans. These requirements have been generalized from the use cases identified in the context of the MARIO project [29]. An overview of the functional requirements is presented in Section 7.1.1, whereas Non Functional Requirements are discussed in Section 7.1.2.

7.1.1 Functional Requirements

Functional requirements are defined as capabilities that must be met by a system to satisfy a form of request. Therefore, functional requirements vary a lot depending on the objectives of the specific system. In this section we discuss the general functional requirements that a social robots should meet.

Perceiving/Interacting/Motioning within the Environment. A Social Robot should be able to perceive the structural features of its operating environment, to move itself within and physically interact with its operating environment. These requirements *must* be met by all embodied agents that need to interact with their operating environment through their physical body, such as mobile robots or service robots. However, a Social Robot could overlook these requirements if the interaction with humans is limited to non-physical languages (e.g. spoken language) and if it does not need to pereceive the external environment. Examples of this kind of robots are typically employed as personal assistants (e.g. Amazon Echo, Google Home etc.).

Interacting with Humans. Interaction between robots and humans can take several forms depending on human-robot proximity (cf. [103]). For a social interaction it is important that humans and the robots are co-located in the same environment. Within the same environment the interaction may require mobility, physical manipulation, cognitive (e.g. natural language understanding) or emotional (e.g. emotion recognition) abilities. Here, we highlight the most important abilities that enable interaction with humans and increase the social acceptability of social robots.

Dialoguing is a form of interaction where two or more parties communicate. There are two main forms of human-robot dialogue *verbal* and *non-verbal.* Social Robots should be able to interact with humans using natural language (i.e. verbal communication). Natural language dialoguing involve capabilities related to speech and natural language processing such as: *(i) Speech recognition*, i.e. the ability of recognizing and translating spoken language into digital-encoded text; *(ii) Natural language understanding* (also called machine reading), i.e. the ability of understanding the meaning of the text and transforming the meaning to a formal structured representation that can be interpreted by machines; *(iii) Dialogue managing*, i.e. the ability of keeping the history and state of a dialog, managing the general flow of the conversation and formulating the semantic representation of the robot's utterances;

(iv) Natural language generation, i.e. the ability of generating natural language text from a semantic representation of the utterance; *(v) Speech Synthesis*, i.e. the ability of converting the natural language text into speech.

Non-verbal interaction include the use natural cues (e.g. gaze, gestures, body positioning etc.). The use of basic cues can bootstrap a person's ability of developing a social relationship with a robot [62]. For example, facial gestures [38] and motion behaviors [63] may facilitate to develop a social relationship with a robot.

Emotions play a significant role in facilitating human-robot interaction (e.g. [155, 43]). Therefore, it is important that a Social Robot is able to recognize and identify emotions in humans, and to express emotions.

Learning and Memorising Knowledge. In order to increase its social acceptability and to evolve its social skills, a Social Robot must be able to learn new knowledge (e.g. facts, rules, norms etc.) and store and integrate the new knowledge with the already acquired knowledge. Continuously evolving the robot's knowledge is useful for adapting the robot's behavior in order to accommodate humans' requests in a way they expect.

7.1.2 Non Functional Requirements

One of the major challenges in robotics concerns the design of software architectures to support the development robot behaviors as plug and play applications [49]. The robot software architecture should offer extensibility mechanisms to support the composition of new robot behaviors by combinating and reusing of the existing services as building blocks. The requirements of flexibility, modifiability and extensibility of the software architecture is even stronger for social robotics applications. In fact, social robotics applications might involve a wide variety of components ranging from the component that controls the wheel engines (i.e. components that directly access to the robot's hardware) to the component aimed at understanding what the user says (i.e. components that perform high-level tasks). Managing all these aspects in every behavior is expensive since the basic components are re-implemented instead of reused.

In order to facilitate the development of social robotic applications, an architecture is required *(i)* to abstract as much as possible basic robot capabilities (e.g. speaking, moving etc.), thus let developers focus only on high level behavior; *(ii)* to allow behaviors to share and reuse common functions and information, thus enabling behaviors to delegate work to other components of the architecture; *(iii)* to enable incremental development of robot behaviors, thus allowing the development of more convoluted behaviors from basic ones.

7.2 Robot Software Architecture Overview

The objective of the robot software architecture presented in this chapter is to provide a flexible and extensible platform for development, deployment and management of social-robot behaviors. A robot behavior is a sequence of actions performed by the robot in order to achieve a goal. An example of behavior is “entertaining the user”. This behavior might involve a series of actions like: “approaching” and “dialoguing” to the user, “showing” videos, “reproducing” music and so on. Actions requires some robot capabilities like: “moving”, “speaking”, “listening”, “understanding”, “showing images” and so on.

Robot capabilities can be classified into basic and convoluted capabilities. A similar classification has been proposed by Duffy in his PhD thesis [61]. Basic capabilities include both the robot primitive functionalities (e.g. reproducing or recording sounds) and basic platform services strongly related to the robot primitive functionalities (e.g. speech recognition). Convoluted capabilities are higher level services/functionalities (e.g. making phone calls) built on top of the basic ones. This solution defines a layered architecture in which capabilities of higher level invoke or activate capabilities on lower levels (cf. [5, 53, 134])

From a developer point of view, both classes of capabilities correspond to functionalities provided by the robot platform. The main difference between these two

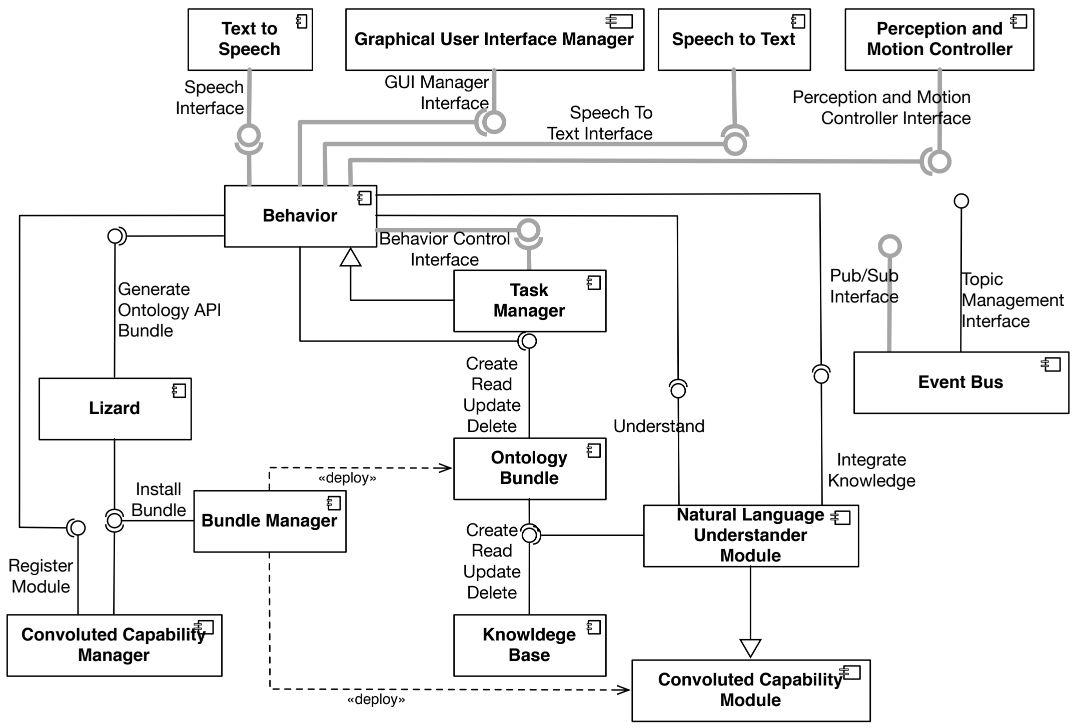

Text to Speech
Graphical User Interface Manager
Speech to Text
Perception and Motion Controller
Speech Interface
GUI Manager Interface
Speech To Text Interface
Perception and Motion Controller Interface
Behavior
Behavior Control Interface
Task Manager
Generate Ontology API Bundle
Pub/Sub Interface
Topic Management Interface
Event Bus
Lizard
Create Read Update Delete
Understand
Integrate Knowledge
Ontology Bundle
«deploy»
Install Bundle
Bundle Manager
Natural Language Understander Module
Register Module
Knowlwdge Base
Convoluted Capability Manager
Convoluted Capability Module

Figure 7.1: The software architecture of the social robot.

classes of capabilities is that convoluted capabilities are services that can be included in the robotic platform after the first deployment of the architecture. Another difference is that basic capabilities are inherently shared by most of the robot behaviors (e.g. moving or speaking), convoluted capabilities might be behavior dependent or not. For example, the behavior enabling the user to make phone call requires the ability of the robot of understanding what the user says (which could be a behavior-independent ability of the robot) and the ability of making phone calls (which is an ability used only by one behavior).

The architecture shown in Figure 7.1 aims at fulfilling the requirements described in Sections 7.1.1 and 7.1.2. In particular, this architecture gives emphasis to flexibility, modifiability and extensibility requirements. The requirement of extensibility is also in line with the principles of the behavior-based robotics [5]. Behavior-based robots are initially provided with basic behaviors (such as that for charging the battery or for avoiding obstacles) and more complex behaviors are added in a second stage.

This architecture also enables (at running-time) to deploy new robot-capabilities (that do not require new hardware components) and to extend the structure of the knowledge base. This feature allows to incrementally develop the robot's architecture. A concrete example of this kind of architecture has been developed in the context of the MARIO project which is the case study for this thesis. The next section provides an overview of the components of the architecture.

7.3 Components

This Section describes the main components of the architecture depicted in Figure 7.1.

7.3.1 Behaviors and Task Manager

This Section overviews the behavior-based mechanism supported by this architecture. Behaviors are software artifacts implementing the behavioral capabilities of the robot. Behaviors are orchestrated by the Task Manager which is a special behavior that actively coordinates other behaviors and manages the functional capabilities of the robot.

7.3.1.1 Behavior

A Behavior is a software component that aims at realizing specific goals. Behavior examples include "play music", "locate a user", "take user to a place" etc. A behavior relies on perceptual capabilities of the robot that provide sensor data (as made available by the robotic platform), performs potentially complex processing (e.g. involving retrieving knowledge from and adding knowledge to the knowledge base), and controls robot's actuators and devices to operate on the environment and interact with the user. Each behavior maintains and updates an internal state, and decides the actions to perform based on sensor data, its state, the general state of the robot and its internal behavior-specific logic.

Each behavior has to expose an interface to allow the task manager to control the behavior (i.e. the Behavior Control Interface). This interface allows the Task Manager to start and stop the behavior. Moreover, in order to implement an affordance-based behavior arbitration[1], the Behavior Control interface allows the Task Manager to retrieve the situations that can be managed by the behavior. Using the Behavior Control Interface, the Task Manager can also grants the access to robot's capabilities to the behaviors. Once granted to use the robot's capabilities, *(i)* the behavior can use the interface provided by the Text To Speech component to make the robot speak; *(ii)* the behavior can show to users pictures and videos using Gui Manager Interface provided by the Graphical User Interface Manager; *(iii)* the behavior can subscribe to the Speech to Text topic to retrieve what the user says; *(iv)* the behavior can use the Perception and Motion Controller interface to retrieve sensor data, and make the robot move in its operating environment.

Behaviors are able to store/retrieve knowledge from the Knowledge Base through the Ontology Bundles. An Ontology Bundles implements RESTful service delivering CRUD (Create, Read, Update, Delete) operations for a component of the Ontology Network. A behavior can extend (at both intensional and extensional level) the knowledge base with the specific knowledge that it need by deploying new Ontology Bundles. This is done by using the Lizard interface that allows to generate an Ontology API bundle for the Ontology Module needed by the behavior (in case that the knowledge needed is not already covered by the Ontology Network).

A behavior can use robot's convoluted capabilities. The convoluted capabilities used by the behaviors in Figure 7.1 are the capabilities for natural language understanding[2]. Using Natural Language Understander modules a behavior is able to *(i) Understand* natural language texts, i.e. associating a natural language text with a formal representation of its meaning; *(ii) Integrate Knowledge* extracted from the natural language texts into the knowledge base. A behavior can also extend robot's

[1]More details of this mechanism are provided in the next section.

[2]This is only an example of convoluted capabilities that a behavior can use. The architectural pattern involving the Behavior, the Natural Language Understander Module and the Convoluted Capability Manager can be replicated for other kinds of robot's capabilities.

convoluted capabilities by registering new components using the Register Module Interface of the Convoluted Capability Manager. The new Convoluted Capability Manager is then deployed by the Bundle Manager and becomes available to other behaviors.

7.3.1.2 Task Manager

The *Task Manager* is a special behavior that actively coordinates other behaviors and manages the functional capabilities of the robot. It acts as a high-level controller and supervisor, allowing the robot to execute behaviors. Once started by the task manager, behaviors use robot's computational and sensor/actuator resources to achieve a goal. Specifically, the Task Manager is responsible for:

1. Processing incoming data/events (as provided by other software components) and reasoning over the actual state and available knowledge, in order to detect situations that require behavior activations;

2. Coordinating, scheduling and prioritizing task executions;

3. Activating, suspending, resuming and terminating tasks, as a result of a continuous decision making process;

4. Monitoring task executions, to detect successful task completions as well as abnormal terminations, failures and exceptions that the tasks are unable to directly handle.

The Task Manager implements an hybrid strategy for arbitrating the behaviors (i.e. deciding which behavior to execute at each time). It implements a *purely reactive* strategy through a collection of pre-programmed event-condition-action rules. This strategy targets the most simple requests which do not need to build and reason on a complex, abstract world models. For example, let the user make a phone call or remembering the user to take his pills does not require a complex control strategy. The purely reactive strategy has proven to be effective for a variety of problems that can be completely specified at design-time with simple rules [140]. However, it is inflexible at runtime due to its inability to store new information in order to adapt the robot's behavior on the basis of its experience. Moreover, the burden of predicting all possible input states and choosing the corresponding output actions is completely left to the designer.

An extension of this the purely reactive strategy is a behavior-based approach relying on the notion of *affordance.* The notion of affordance has been introduced by Gibson [91] who devised a theory of how animals perceive opportunities for action. Gibson called these opportunity affordance. He suggested that the environment offers the agents (people or animals) opportunities for actions. For instance, a door can have the affordance of "openability". The strategy for controlling behavior implemented by the task manager exploits and goes beyond of the notion of affordance introduced by Gibson. The behavior-control mechanism is based on the assumption that not only phyiscal objects, but also complex situations (e.g. the user wants to listen to some music and the robot battery need to be charged) afford actions. A complex situation can be seen as the fullfilment of certain conditions at a certain time. These conditions may involve temporal aspects (e.g. lunchtime may afford the task remember the user to take the pills), the perception of certain physical objects, the reception of a command (e.g. I want to listen to some music), or, even the existence of certain state-of-affairs (e.g. the situation the user is sitting on a chair for a long while may afford the task entertain the user). All the conditions that cause the start or the stop of behaviors can be stored in the knowledge base my means of the Affordance module of the ontology network (cf. Section 3.3.1). The task manager continuously check these conditions, and, whenever a condition is satisfied, it retrieves the actions afforded by the fulfilled situation.

7.3.2 Event Bus

The *Event Bus* aims at providing the architecture's components with message-based communication mechanism. This component enables the communication among components in a publish-subscribe form. The Event Bus exposes two interfaces, namely: *(i)* the *Topic Management Interface* which allows software components to create new topics; *(ii)* the *Publish/Subscribe Interface* which allows software components to publish messages on/subscribing to a topic. This component provides an asynchronous communication mechanism that decouples software components. This mechanism is used for realizing the communication between behaviors and basic robot's capabilities (these "communication channels" are highlighed in gray in Figure 7.1).

7.3.3 Bundle Manager

The *Bundle Manager* allows to extend the architecture by dynamically deploying new software components. The Bundle Manager aims at providing an OSGi-compliant[3] platform for the robot's software architecture enabling a dynamic component model. Applications or components, coming in the form of bundles[4] for deployment, can be installed, started, stopped, updated, and uninstalled without requiring a reboot. These features ensure the flexibility and the extensibility of the software architecture. For the sake of simplicity, the Bundle Manager realizes only one interface which allows Lizard and Convoluted Capability Manager to dynamically install Ontology Bundles and Capability Modules respectively.

7.3.4 Knowledge Management Framework

This Section presents the architectural components involved in the management of the robot's knowledge. The Knowledge Management Framework consists two main components, namely the Knowledge Base and Lizard.

7.3.4.1 Knowledge Base

The *Knowledge Base* is the component intended to store the robot's knowledge in a structured format. The Knowledge Base provides facilities to create, read, updated,

[3]OSGi, https://www.osgi.org/

[4]Bundle is a term borrowed from Java-based platforms. A bundle is defined as a group of Java classes and additional resources equipped with a detailed manifest file.

and delete (i.e. CRUD) facts. The reference data model for the knowledge base is the RDF framework[5]. The facts stored into the knowledge base are compliant to Ontology Network (c.f. Section 3). The Ontology Network is defined in OWL language[6] The knowledge base component also includes a reasoning engine that is able to infer logical consequences (i.e. entailed facts).

7.3.4.2 Lizard

Lizard is an Object-RDF mapper providing software components with the access to the knowledge base. Given an ontology as input, Lizard generates an application bundle (i.e. an Ontology Bundle) that provides applications with APIs for accessing RDF facts stored in the Knowledge Base following the Object-Oriented paradigm. The APIs reflect the semantics of OWL ontology and allow transparent access to the knowledge base. The Ontology Bundle also provides a RESTful layer that exposes Object Oriented paradigm by using the REST architectural style over HTTP. The Ontology Bundle avoids client applications to deal with OWL and RDF or to interact with a knowledge base by means of SPARQL queries.

7.3.5 Basic Capabilities

Basic capabilities include both the robot primitive functionalities (e.g. reproducing or recording sounds) and basic platform services strictly related to the robot primitive functionalities (e.g. speech recognition). These capabilities are used by most of the robot behaviors. This Section briefly describes the components providing the basic robot capabilities. The capabilities delivered by these components are the most significant with respect to the requirements for a social robot (cf. Section 7.1.1 and 7.1.2). The Section omits components providing general purpose services (e.g. network connectivity).

[5]RDF 1.1 Concepts and Abstract Syntax, https://www.w3.org/TR/rdf11-concepts/

[6]OWL 2 Web Ontology Language Document Overview (Second Edition), https://www.w3.org/TR/owl2-overview/

7.3.5.1 Text to Speech and Speech to Text

The *Text to Speech* component aims at converting natural language text into speech. The Text to Speech implements an interface that allows behaviors to synthesize and to reproduce synthesized speech. The *Speech to Text* component converts spoken language into digital-encoded text. The Speech to Text component creates a topic for publishing the converted text. The behaviors that need to recognize what users say will subscribe to this topic and they will receive a message whenever the text is converted.

7.3.5.2 Graphical User Interface Manager

Most of the social robots are equipped with one or more (touch-)screens in order to complete the message conveyed by verbal communication. The joint use of verbal and visual language for human computer interaction falls into the broader category of multi-modal human-computer interaction. This architecture supports a bi-modal interaction involving a both verbal and visual language. The verbal communication relies on Text to Speech and Speech To Text components, whereas visual communication is ensured by the Graphical User Interface Manager. The *Graphical User Interface Manager* component aims at providing behaviors with facilities for managing the robot's GUI. The component realizes an interface (i.e. the *GUI Manager Interface*) that allows behaviors to show widgets on the screen. Using this interface the behaviors can also receive a feedback whenever the user interact with such widgets.

7.3.5.3 Perception and Motion Controller

The *Perception and Motion Controller* provides functional capabilities for supporting human-robot interactions. It includes a set of software routines that enable the robot to perform a series of motion behaviors (e.g., approaching the user, following the user, recharging, driving the user to a destination, etc.). The robot is able to estimate current situation using different level of information, coming from several sensors like: *(i) RFID* in order to detect a list of tagged objects; *(ii) Camera* to detect user using face and posture recognition and extract his relative position and distance; *(iii) Laser* to perceive and identify dynamic objects/persons that were not included in the static map (SLAM system). This information can be used to avoid obstacles and approach/follow patient. The *Motion Controller Interface* provide behaviors with high-level functionalities such as: go to X (where X is a point within the robot's operating area), give me user's position, give me the tagged objects that are currently near the robot etc.

7.3.6 Convoluted Capability Subsystem

The *Convoluted Capability Subsystem* manages the convoluted capabilities of the robot. Convoluted capabilities are services that can be dynamically included in the robotic platform after the deployment of the architecture. The new capabilities can be installed by robot's behaviors that intend to make available new functionalities for other behaviors. The *Convoluted Capability Subsystem* consists of the *Convoluted Capability Manager* and the *Convoluted Capability Module.* The Convoluted Capability Manager is responsible for the dynamic deployment of new capability components. It realizes an interface (i.e. "Register Module") which accept as input an application bundle realizing the new capability to deploy. Once received an application bundle, the Convoluted Capability Manager uses the Bundle Manager interface to install the bundle.

7.4 Architecture Prototype for a Real Social Assistive Scenario

A prototype of the software architecture proposed in this chapter has been developed within the context of the MARIO project (cf. Section 1.3). This prototype is aimed at demonstrating feasibility and benefits of the contributions described in this thesis. In particular, Section 7.4.1 two knowledge-intensive robotic running on top of the software architecture. The prototype has been deployed on Kompaï-2 robots, evaluated during supervised trials in different dementia care environments, including a nursing home (Galway, Ireland), community groups and residential settings (Stockport, UK), and a geriatric unit in hospital settings (San Giovanni Rotondo, Italy). Results of the evaluation carried out within the context of the MARIO project are reported in Section 7.4.2.

7.4.1 Delivering Knowledge-intensive Applications for Social Robots

Within the context of the MARIO project we developed two knowledge-intensive applications enabling a social robot at *(i)* Assessing the medical, psycho-social and functional status of a person by undertaking a dialogue-based interaction which is part of the Comprehensive Geriatric Assessment (CGA) diagnostic process; *(ii)* Delivering a reminiscence aimed at stimulating long-term autobiographical memory with verbal interactions that focus on recalling positive memories about people, past activities, experiences and personal events, often with the support of materials such as photos that act as memory triggers.

7.4.1.1 Comprehensive Geriatric Assessment

The Comprehensive Geriatric Assessment (CGA) is a diagnostic process that aims at collecting and analyzing data in order to determine the medical, psychosocial, functional and environmental status of elderly patients, with the goal of improving the diagnostic plan and supporting physicians in the definition of personalized plans for treatment and long-term care.

A multidimensional assessment phase is at the heart of the CGA process and represents a critical, time consuming activity for caregivers. To gather information about the patient, physicians rely on a set of widely accepted, internationally validated formal assessment tools and standardized rating scales designed to evaluate patient's functional abilities, physical and mental health, and cognitive status.

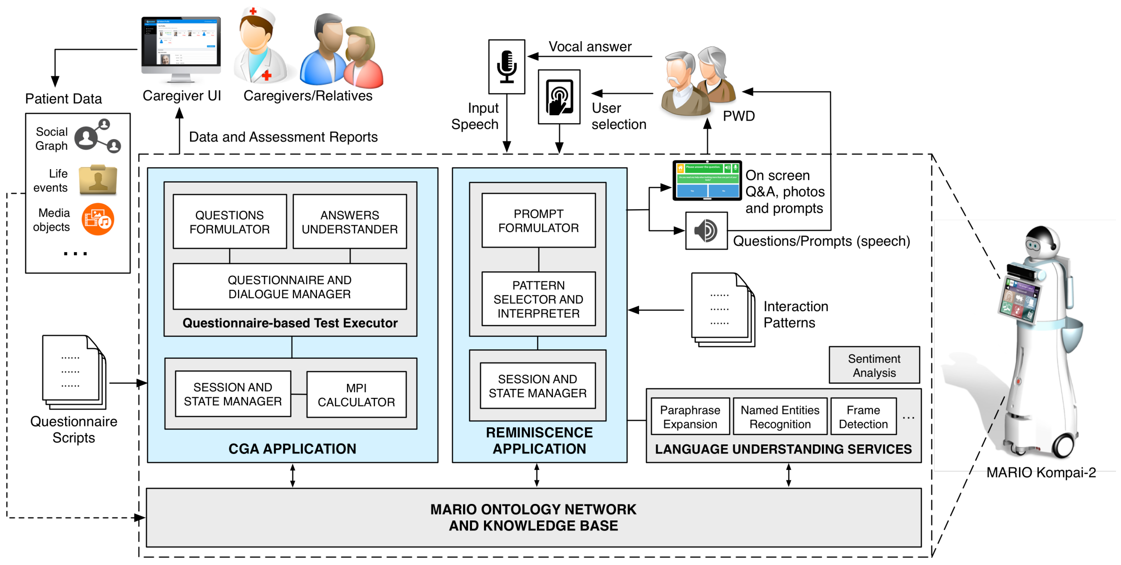


Figure 7.2: Architectural model of the CGA and Reminiscence applications

As part of the assessment tools and procedures, the patient is required to answer questions defined in standardized clinical questionnaires[7] (e.g., about his/her daily life and ability to autonomously perform specific activities). Depending on the answers, a score is given to the patient and evaluated according to a reference rating scale. The assessment enables the evaluation of a Multidimensional Prognostic Index (MPI), a prognostic tool that combines the scores resulting from the questionnaires to derive a single score able to synthetically represent patient's health status and define the severity grade of mortality risk in elderly subjects [168].

A CGA is typically carried out every 6 months and, on average, a questionnaire-based evaluation requires between 20 and 30 minutes per patient to be completed. As most of the total time available to the formal caregiver is consumed to collect information from the patient, the evaluation and definition of a personalized care plan is often performed under time pressure, in particular in the setting of an ambulatory

[7]A standard CGA includes eight assessment tools and scales: Co-habitation status, Medication use, Activities of Daily Living (ADL), Instrumental Activities of Daily Living (IADL), Short Portable Mental Status Questionnaire (SPMSQ), Exton-Smith Scale (ESS), Cumulative Illness Rating Scale (CIRS), and Mini Nutritional Assessment (MNA).

geriatric care unit. Nowadays health professionals increasingly use ICT supporting tools and devices (such as computers and tablets) during the multidimensional assessment phase for recording test results and calculate the corresponding scores. However, it has been observed that these devices and the need to interact with them to input information can represent a "communication barrier" between the caregiver and the patient during clinical interviews [66]. The lack of visual contact with the caregiver can further increase stress and anxiety in frail elderly patients undergoing a cognitive evaluation whose results may potentially impact on their autonomy.

The introduction of a robotic solution able of autonomously performing parts of a CGA is expected to reduce the direct involvement of health professionals in the time-consuming data collection tasks, as well as the perceived tiredness resulting from the performance of repetitive tests. This will enable them to concentrate their efforts on the interpretation of the results and the elaboration of personalized care plans. In the long term, the objective is to enable a continuous monitoring of patient's conditions (e.g., by increasing the frequency of CGA sessions), with an opportunity to early detect relevant changes in the health status. In this direction, the ASSESSTRONIC project[8] and the CLARC framework [20] are investigating robotic solutions for supporting the CGA process.

MARIO's CGA application, whose components are shown in Figure 7.2, aims at enabling the robot to autonomously perform and manage the execution of the questionnaire-based tests required in the CGA process, in order to assist the formal caregivers and physicians in the multidimensional assessment phase and facilitate the evaluation of the Multidimensional Prognostic Index. The CGA application is thus designed to undertake a dialogue-based interaction with the patient, by posing the defined questions and interpreting patient's answers to assign the corresponding scores. Moreover, by recording patient's answers and calculated tests scores, the application can generate health reports for the care staff, to allow them to access, analyze and review test results. The CGA application relies on the CGA ontology

[8]http://echord.eu/essential_grid/assesstronic/

(cf. Section 3.3.2).

In the CGA application (Figure 7.2), the *Session and State Manager* manages the overall execution and status of CGA sessions, coordinating the scheduling and performance of the configured tests. It operates on the basis of the user profile and test configuration settings defined by the formal caregiver and available in the knowledge base. In order to access the ontology and the corresponding data, the CGA module exploits the functionalities and API provided by the knowledge management system introduced in Chapter 5. As CGA tests are typically performed during a clinical encounter (e.g., when the patient is admitted to or discharged from the geriatric unit), a CGA session can be initiated by the caregiver either through the provided graphical interface or by vocally interacting with the robot, asking MARIO to perform an assessment of the patient.

When the application is activated, the *Session and State Manager* initiates and monitors the sequential execution of the specific tests to be performed. Specifically, the *Questionnaire-based Test Executor* is in charge of the execution of questionnaire-driven tests, and is thus responsible for engaging the patient in a dialogue-based interaction, with the aim of gathering information that enables the calculation of assessment scores and prognostic indexes. The dialogue flow is driven by the robot and unfolds on the basis of a continuous question-answer interaction pattern. To this end, the component relies on the speech-based communication capabilities provided by the MARIO framework and operates on the basis of scripted representations of the different questionnaires that are part of the CGA. Dialogue management is driven by the questionnaire structure, which acts as a blueprint for the question-answer interactions and provides the ordering and sequencing of the assessment questions. For a specific test, the corresponding questionnaire script is derived from its description and representation retrieved from the knowledge base.

Basically, the application gradually presents spoken questions to the patient and gathers her vocal responses to be interpreted. Each question formulated by the app and uttered by the robot is contextually shown on the touch screen. Depending on the question type (open-ended or closed-ended question), possible answers may be shown on the screen as well. This enables the patient to provide her answers by directly speaking to the robot or by interacting with the graphical interface. The application relies on natural language understanding capabilities for interpreting patient's utterances representing answers to the evaluation questions. A proper interpretation of provided answers ultimately results in the assignment of a score to each answer. The *Answers Understander* takes as input the textual representation of patient's utterances, as provided by the speech-to-text subsystem. The actual interpretation strategy directly depends on the question and corresponding answer type.

In the case of Yes-No questions (e.g., *"Do you need any help to wash or bathe yourself?"*), which cover most of the items in the CGA questionnaires, patient's answers are matched against regular expression patterns that aim at capturing both positive and negative answers. The patterns were built by exploiting existing linguistic resources, in particular the Paraphrase Database (PPDB)[9], an automatically extracted multilingual database of paraphrases. PPDB has been re-engineered in RDF and included as part of the knowledge base, according to the reference PPDB ontology[10] we defined. In the case of *Wh*-questions, which cover most of the items in the Short Portable Mental Status Questionnaire (e.g., *"What is the date today?"*, *"When were you born?"*, *"Who is the current Pope?"*), the understanding process maps to the task of comparing patient's answers with known properties of named entities, such as persons (including the patient herself, her parents, and well-known present and historical individuals) and dates. These properties can be directly retrieved or derived by querying the knowledge base (e.g., by accessing patient's profile to get her birth day or her mother's maiden name) and then compared with the provided answer. The matching process relies on specialized understanding capabilities that restrict the recognition and interpretation to specific domains, such as locations and numbers, used for example when the user is asked to perform basic math calculations as part of the SPMSQ questionnaire.

[9]http://paraphrase.org/

[10]http://w3id.org/ppdb/ontology/ppdb.owl

Finally, the *MPI Calculator* is responsible for calculating the overall Multidimensional Prognostic Index, taking into account the scores and rating scales resulting from the execution of the assessment tests.

7.4.1.2 Reminiscence Therapy

Reminiscence therapy is based on verbal interactions that focus on recalling positive memories about people, past activities, experiences and personal events, often with the support of materials such as photos that act as memory triggers. Reminiscence therapy thus targets and aims at stimulating long-term autobiographical memory, which is relatively unaffected by the disease. Reported effects range from increased socialization and self-esteem to improvements in cognition and mood, with a general positive impact on quality of life [129, 213].

As discussed in [120, 200], existing systems for supporting reminiscence aim at improving traditional practice and basically consist of software applications, deployed on desktop/laptop computers or tablets, that act as personalized multimedia systems for the storage and retrieval of digital reminiscence materials. Our approach focuses on robot-enabled delivery of so-called *simple reminiscence* [129], based on a conversational approach and highly focused verbal and visual memory triggers. The application, whose components are shown in Figure 7.2, is thus specifically designed to actively prompt the PWD and engage her in interactive and personalized reminiscence sessions, where dialogue-based interactions are complemented with multimedia content associated with relevant people, places and life events.

Supporting reminiscence requires the availability of user-specific factual knowledge, gathered in the form of a life history from family members and caregivers. In order to represent, structure, store and make available this heterogeneous information, specific ontology modules were defined as part of the MARIO Ontology Network. The ontology modules supporting reminiscence cover three main knowledge areas, i.e., personal sphere, life events and multimedia content. They address the need of representing persons and their basic biographic information, family and social relationships among them, life events, and multimedia objects along with their association with persons, places and life events.

While biographic information covers basic data (e.g., first/last name, birth date and hometown), family and social relationships enable the definition of a social graph for the PWD. User profiles can be further enriched with the definition of life events on the basis of a generalized representational schema, which includes the primary properties of a life event and relies on the *time-indexed situation* ontology design pattern. In addition to a title and a textual description, a life event is characterized by *(i)* a temporal dimension, to allow representing events that occurred in a specific date (e.g., a marriage) or over a period of time (e.g., attendance to college); *(ii)* a set of participants, to express the participation of potentially multiple persons in the event; *(iii)* a location where the event took place; *(iv)* a set of multimedia objects (photos, videos, etc.) associated with the event. Starting from this generic representational structure, the need to specialize life events to cover specific domains led us to narrow of the scope of the modeling approach and adopt a frame-based representational structure. Specific life events and their properties are modeled as *frames*, to cover typical domains including work and education (e.g., school attendance and working experiences), personal and family events (such as a marriage and the birth of a child), and living and travel experiences. A *frame* provides a schema for conceptualizing the description of an event type and its participants in terms of *frame elements* or *semantic roles* [81]. For example, a marriage involves two persons participating as *partners*, and takes place in a specific *location* and *date*. Similarly, a birth event includes an *offspring* (the person that was born) and involves two persons as *mother* and *father*, along with the *birth place* and *date*.

The association between media objects and other entities relies on a semantic tagging approach, as defined in a *tagging* ontology module (cf. Section 3.3.3) designed so that any object (including frames or even named graphs) can be used to categorize or describe the entity being tagged. This allows defining, for example, life events and persons as tags for an image, in addition to simple properties expressing when and where a photo was taken.

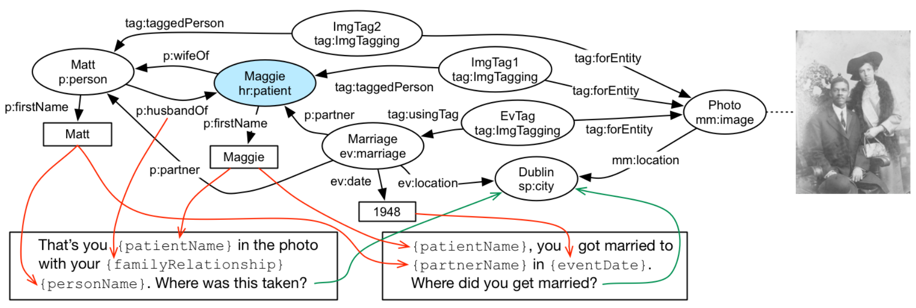


Figure 7.3: Example of prompting questions formulation from user-specific knowledge graph

User-specific knowledge is directly exploited by the application for engaging the patient in reminiscence sessions. A reminiscence session can be triggered as a result of a direct request issued by the user, either through the GUI provided by the MARIO framework and available on the touchscreen, or via vocal commands, exploiting the multimodal interaction capabilities provided by the robot. Specifically, a dialogue-based reminiscence session is driven by an extensible repertoire of *interaction patterns*, that allow the application to prompt the user through specific questions and triggers, associated with media objects such as images that are contextually shown on the touchscreen available onboard the robot.

An *interaction pattern* consists of: *(i)* a *precondition*, with constraints expressed as queries over the knowledge base, defining under which conditions the prompt can be instantiated and used; *(ii)* a parametric *prompting question* to be used for triggering reminiscence, represented as a partially-formulated prompt template containing variables to be instantiated with data from the knowledge base; *(iii)* a set of queries over the knowledge base providing a binding for the variables in the prompting question. On the basis of these patterns, the main step in the application logic consists in contextually identifying the applicable patterns, by accessing the knowledge base to evaluate their preconditions and instantiate the corresponding prompt. As visual memory triggers are fundamental for reminiscence, the patterns are always evaluated taking into account the availability of an image that will be shown to the user while the prompt is uttered by the robot through its text-to-speech capabilities.

Prompting questions are defined to cover the aforementioned knowledge elements, including life event types, people and tagged media objects. As informally shown in Figure 7.3, given a photo with information on where it was taken, and who appears in the picture, examples of parametric prompting questions that also exploit family/social relationships include *"Is that your* `{familyRelationship}` `{personName}` *in the photo with you?"* or *"That's you* `{patientName}` *in the photo with your* `{familyRelationship}` `{personName}`*. Where was this taken?"*. Similarly, the association between photos and life events can be exploited to formulate questions about the event. Assuming, for example, that there is a marriage event where the PWD is one of the partners, prompting questions such as *"*`{patientName}`*, you got married to* `{partnerName}` *in* `{eventDate}`*. Where did you get married?"* can be formulated.

In these examples, prompting questions take the form of targeted questions that assume a specific, known answer, from a simple positive/negative reply to the identification of specific persons, places, dates or events. In the case of prompts formulated as targeted questions, the interaction patterns are extended by defining the answer type (e.g., a yes/no answer, a person, a date, etc.), the actual expected answer (by referencing a concrete entity in the knowledge base, such as a specific person or location), and utterance templates that are used by the robot depending on whether user's reply matches the expected answer or not. These additional elements are used by the application in the user answer processing step, where the capabilities of the natural language understanding subsystem are used. Targeted questions with specific answers constrain the language interpretation domain: the interpretation maps to the task of named entity recognition and linking with respect to the knowledge base, to identify mentions of named entities (e.g., a person or a location) in user's utterance and check the correspondence with the entity representing the expected answer. Depending on the outcome of this step, the robot can reply with a confirmation and encouragement if the answer is correct, or otherwise provide the patient with intermediate hints or the expected answer.

As an approach based on repeated questions can create stress and anxiety and be inappropriate for people with cognitive impairments, prompting questions can also be defined as open-ended prompts that aim at stimulating conversation. So for example, considering again a picture related to a marriage event, the robot can use prompts like *"*`{patientName}`*, you got married to* `{partnerName}` *in* `{eventDate}`*. Tell me about you wedding day! What was it like?"*. Similarly, given a picture of one of patient's children, prompts like *"*`{patientName}`*, this is your* `{childRelationship}` `{childName}` *in this nice picture. What was* `{childName}` *like as a child?"*. When dealing with this type of prompts, the interpretation of user's replies adopts a different strategy and relies on *sentiment analysis* capabilities. Basically, the application attempts identify the polarity of user's utterances, to recognize whether the visual and verbal prompt is eliciting a positive, neutral or negative mood or reaction from the person. The interaction patterns are extended in this case by defining utterance templates for the different polarities, so that the robot can, e.g., encourage the user to tell him more about the subject if the reaction is positive, or otherwise propose to move to another picture.

The selection of the interaction patterns is thus a dynamic process, driven by patient's replies and reactions, and by traversing the links in the knowledge graph on the basis of the dialogue context and history. So, for example, a question about when a photo was taken can be followed by a question concerning a person that appears in the picture, and then move to a life event where the person participated in, and so on, exploiting the properties of and links between the entities in the knowledge base. Similarly, sentiment data can influence the selection process as well: for example, a negative reaction to a picture concerning an event or showing a specific person may lead to avoid subsequent prompts with images about the same event or with that person. Moreover, sentiment data emerging from the interactions can be associated with the concerned entities (pictures, people, events, etc.) and stored in the knowledge base. This knowledge is then used in subsequent reminiscence sessions so that, for example, photos that generated a positive reaction are favored in the selection process, whereas those causing negative reactions are less likely to be reproposed.

7.4.2 Evaluation

In this section we report the key results obtained from the trials undertaken within for the MARIO project. More details on the evaluation methodology and the results can be found in [48]. The evaluation involved 38 people with dementia, each engaged at least three times for 60 minutes, for a total of 195 engagements. The evaluation methodology combined two approaches, a quantitative approach involving the use of standardized assessment questionnaires, and a qualitative approach to capture the perceptions and experiences of key stakeholders.

7.4.2.1 Quantitative Evaluation

Each participant performed seven different assessment questionnaires before and after the introduction of the robot: *(i)* The mini mental state [76], a 30-point questionnaire that is used extensively in clinical and research settings to measure cognitive impairment; *(ii)* Cornell Scale for Depression in Dementia (CSDD) [3], a screening tool for depression in participants with dementia; *(iii)* Quality of life in Alzheimer's disease (QOL-AD) [132], a 13-item self- and caregiver-report measure that assesses quality of life across several domains; *(iv)* The Comprehensive Geriatric Assessment, a multidimensional diagnostic process intended to determine a person's medical, psychosocial, and functional capacity (cf. Section 7.4.1.1); *(v)* The Multidimensional Scale of Perceived Social Support (MSPSS) [215], a subjective, self-reporting measurement of social support; *(vi)* The 14 ITEM Resilience scale (RS-14) outlined in [47]; *(vii)* The Observational Measurement of Engagement (OME) [50], a tool for assessing interactions between people with dementia and environmental stimuli.

An overview of the result of the quantitative analysis is presented. The complete results are reported in details in the MARIO D8.3 deliverable [48]. In the residential care setting, no statistically significant difference was found in participant's scores pre and post MARIO as measured by the QoL-AD for patient, MSPSS, CSDD and the RS14. In the hospital setting, a significant improvement was observed in patient's depression (CSDD: $p=0.01$), resilience (RS-14: $p<0.0001$), and quality of life

(QoL-AD) scores (patient: p=0.04). However, there was no statistically significant difference found in the MPSS. In the community setting, only 2 participants completed the standardized assessment questionnaires, due to the small sample size involved these were not analysed on a single site basis. When scores across all sites were compared the combined participant's resilience scores pre-and-post MARIO were the only scores found to be significant (RS-14: p = 0.04).

A significant improvement was observed in the quality of life score of participants in the hospital setting (QoL-AD patient: p=0.04), indicating that engagement with MARIO had a positive impact. In MARIO study, there was some very limited evidence of impact on resilience levels in the hospital setting (RS-14: p<0.0001) and no statistically significant difference for other indicators emerged from the evaluation.

7.4.2.2 Qualitative Evaluation

Following [210], a qualitative interpretive descriptive design was used to gather and analyse data from participants. Interpretive description was designed to give participants a voice about their own experiences. Semi-structured interviews were developed from the literature and expertise of the researchers across all pilot sites.

In the following it is presented an overview of the results regarding the overall impact. The complete results are reported in details in the MARIO D8.3 deliverable [48]. The analysis of the results revealed five key themes, namely: *(i)* Perceptions/attitudes towards MARIO; *(ii)* Challenges to the use of social robots in the context of the real world of dementia care; *(iii)* Impact of MARIO on cognitive engagement, autonomy, loneliness, resilience, and quality of life; *(iv)* Utilization of the MARIO applications.

Perceptions/Attitudes towards MARIO. The data revealed that most participants across the three settings were accepting and had positive perceptions/attitude toward MARIO, and the deployment of social robots. People with dementia enjoyed their interactions with MARIO and they often referred to MARIO as he or she, conceptualizing him as an embodied presence and some referred to MARIO as "a friend" describing how they had developed a relationship with MARIO.

Challenges to the Use of Social Robots in the Context of Dementia Care. Whilst the majority of carers and managers were positive about MARIO, some expressed concerns regarding the deployment of robots in dementia care. These concerns related to the fact that robots should not be seen as a replacement for human interaction or care givers. Carers, managers, and relatives across all sites commented on the fact that the stage of dementia was an important consideration when deploying robots to work with people with dementia. They suggested that MARIO was most useful to people at the mild to moderate stage of dementia.

The impact of MARIO. Participants suggested that the main impacts were on increased cognitive engagement, autonomy, loneliness, resilience and quality of life. Carers/relatives across all three settings indicated that working with MARIO positively impacted the level of cognitive engagement of participants with dementia. Working with MARIO, particularly in the residential settings enhanced the autonomy of participants with dementia because it enabled them to make autonomous choices about what activities they wanted to do. It was also observed that people with dementia spent less time alone and more time socially engaged. MARIO was found to facilitate conversations and social engagement with staff and relatives and provided participants with the opportunity to converse about their own life. MARIO had a limited impact on resilicence, only a few participants reported that they felt that MARIO had impacted on their resilience. The provision of personalised activities, and the fact that MARIO provided entertainment and diversion that had a positive impact on quality of life of participants. It was also suggested that MARIO had an impact on quality of life because MARIO expanded social activities and facilitated discussions and conversations.

Utilization of Applications. All the MARIO applications were personalised to the individual. The most frequently used applications across all pilot sites were, in order of preference, My Music, My Memories (i.e. Reminiscence application) and My Games, My Chat and My Family and Friends. In the following report the utilization of the applications presented in Chapter 7.4.1.

The Reminiscence application described in Section 7.4.1.2 was the second most popular app and was widely used and selected by participants with dementia across all sites. Participants with dementia enjoyed using the app, and they emphasized they enjoyed looking at the materials on MARIO. Relatives/carers in all settings also commented on the importance of this application for the person with dementia and some described their enjoyment in helping to compile the materials for the application and how the application drew on preserved long-term memories, engaged, stimulated the participant with dementia and created enjoyment.

The Comprehensive Geriatric Assessment (CGA) application, described in Section 7.4.1.1 enabled MARIO to autonomously undertake a specific number of questions required in the CGA process. The average time of the CGA app sessions was 7.25 $\pm$ 2.55 minutes. Having a robot undertake the assessment process was found acceptable to participants, with dementia and to caregivers. In addition, the findings indicate that having MARIO conduct the CGA may optimise the “time care” of healthcare professionals, allowing them to focus on other more meaningful patient activities.

7.5 Discussion

This chapter presented a component-based software architecture which is aimed at supporting knowledge-intensive tasks performed by social robots with a platform implementing and orchestrating a set of off-the-shelf components. As far as RQ5 (cf. Section 1.1) is concerned, these components rely on contributions of this thesis and state-of-the-art semantic web technologies to provide developers with generic functionalities for structuring and accessing knowledge, exploiting linked open data, and sharing knowledge among architectural components.

This architecture benefits of both symbolic and subsymbolic techniques (cf.

Chapter 1). Subsymbolic techniques are used in perceptual tasks (such as translation of spoken language into text), whereas symbolic techniques are used for controlling the robot at an higher level [93]. In particular, subsymbolic subsystems of the robot transform low-level perception in symbols so to enable the symbolic processing of the control system. In this way the framework benefits of the state-of-the-art performance on perceptual tasks of subsymbolic techniques without compromising the possibility of having a system that is deterministic and able to explain its behavior and decisions (important requirements for the case study of this thesis, cf. Section 1.3). The other issue that prevented the use of subsymbolic techniques within the control subsystem is the lack of data. But, if more data on the action to undertake in a given situation is made available, then subsymbolic approaches would become viable solutions to exploit for controlling robot's behavior.

Moreover, this chapter presented a prototype which is aimed at demonstrating feasibility and benefits of such a architecture and two applications running on top of the architecture prototype. We claim that the prototype presented in this chapter is only an example of robotic systems that benefit of framework proposed in the thesis. This framework could potentially be integrated, with appropriate adaptions, with every autonomous agents (not limited to embodied systems). The prototype has been deployed on Kompaï-2 robots, evaluated during supervised trials in different dementia care environments.

The findings reveal that social robots do have an important role in the social care of people with dementia. A significant improvement was observed in the quality of life score of participants in the hospital setting, indicating that engagement with the prototype had a positive impact. A lighter impact on resilience level emerged from the quantitative analysis and no statistically significant difference was found in participants scores for other indicators. The data of the qualitative analysis revealed that most participants across the three settings were accepting and had positive perceptions/attitude toward the robot. Participants suggested that the main impacts were on increased cognitive engagement, autonomy, loneliness, resilience and quality of life.

Our ultimate goal is creating a robotic platform orchestrating a series of off-the-shelf components for *(i)* easing *customizability* and *extensibility* of robot's behavior and its social skills; *(ii)* improving both inner (among architectural components) and outer (with external entities) *interoperability*; *(iii)* enabling a *rapid prototyping* of robotic applications; *(iv)* enhancing *reusability* of architectural components. Architecture as well as other contributions of the thesis represent a progress towards this goal. However, there is still a long way to go. The Task Manager proposed for controlling robot's behavior is able to react to only a set of few predefined situations with a fixed set of behaviors. The Task Manager only demonstrated the feasibility of a limited affordance mechanism, while customizability and extensibility of robot's behavior need to be investigated in the future. Moreover, another line of research that could be investigated is the possibility of applying service-oriented techniques such as service composition and orchestration [124]. Such techniques give the opportunity to quickly realize new behaviors by easily assembling (with no need of code) existing behaviors and capabilities.

Similar considerations for the Convoluted Capability Manager. Architecture prototype developed in MARIO has been equipped with some fixed capabilities, such those exploited by behaviors presented in Section 7.4.1, but a Convoluted Capability Manager is missing. Such a component is essential for guaranteeing reusability and extensibility of robots' capabilities, and could potentially be paired with a repository of common capabilities from which behavior developers could easily download and install the capabilities they need.

Chapter 8

Conclusion and Future Work

In this thesis we investigated feasibility and benefits of engineering background knowledge of Social Robots with a framework based on Semantic Web technologies and Linked Data. This research has been supported and guided by a case study (presented in Section 1.3) that provided a proof of concept through a prototype tested in a real socially assistive context.

The thesis contributes to this goal by proposing a component-based architecture centered on the robots' knowledge base, namely, all the components contribute to and benefit from the knowledge base which is the cornerstone of the architecture. The knowledge base is structured by a set of interconnected and modularized ontologies, constituting the MARIO Ontology Network, which are meant to model information relevant for robots' activities in socially assistive context. The knowledge base is originally populated with lexical-linguistic, factual, and ontological information retrieved from the Linked Open Data, and integrated within Framester. The access to the knowledge base is guaranteed by Lizard, a tool that provides software components with an API for accessing RDF facts stored in the knowledge base in a programmatic way.

Moreover, this thesis proposed two methods for creating and manipulating knowledge needed by robots. *(i)* A novel method for automatically integrating knowledge coming from heterogeneous sources with a frame-driven approach. This method aims at evolving the robots' knowledge with information learned during robots' activities. *(ii)* A novel empirical method for assessing foundational distinctions over Linked Open Data entities from a common sense perspective (e.g. deciding if an entity inherently represents a class or an instance from a common sense perspective). This method realizes the first step of a more general procedure meant to automatically generate common sense knowledge by using Linked Open Data as empirical basis.

We also presented two examples of knowledge-intensive robotic applications that benefit of the framework described in this thesis. These two applications have been developed, deployed and tested in a real socially assistive scenario. These two applications enable Social Robots at: *(i)* Assessing the medical, psycho-social and functional status of a person by undertaking a dialogue-based interaction which is part of the Comprehensive Geriatric Assessment (CGA) diagnostic process; *(ii)* Delivering a reminiscence aimed at stimulating long-term autobiographical memory with verbal interactions that focus on recalling positive memories about people, past activities, experiences and personal events, often with the support of materials such as photos that act as memory triggers.

Finally, this thesis presented a quantitative and qualitative evaluation of a prototype of the framework tested in a real socially assistive context that involved 38 people with dementia. The findings reveal that social robots do have an important role in the social care of people with dementia. A significant improvement was observed in the quality of life score of participants in the hospital setting, indicating that engagement with the prototype had a positive impact. A lighter impact on resilience level emerged from the quantitative analysis and no statistically significant difference was found in participants scores for other indicators. The data of the qualitative analysis revealed that most participants across the three settings were accepting and had positive perceptions/attitude toward the robot. Participants suggested that the main impacts were on increased cognitive engagement, autonomy, loneliness, resilience and quality of life.

This thesis also presented some experiments for assessing whether the Web, and in particular Linked Open Data, provides an empirical basis to extract foundational distinctions, and if they match common sense. For testing the former, we adopt and compare two approaches, namely alignment-based methods and machine learning methods. For the latter we use crowdsourcing and compare the judgements of the crowd with those of experts'. For both questions we observed promising results and define a method that can be generalized to investigate additional distinctions.

8.1 Research Questions Revisited

The first chapter detailed several questions that we intended to explore. In this section we review these questions stating the conclusions this work has set forth.

RQ1: What kind of knowledge a robot needs to operate in socially assistive context? What exiting ontologies can be used to organize the robot's knowledge? What ontologies need to be advanced? What domains of interest in this context miss of a conceptualization?

To investigate these questions, this thesis presented a proof-of-concept, namely the MARIO Ontology Network. The most developed MON's knowledge areas are those related to the assistive and medical domain, social and multimedia contents, and user-related information (user data, user's life events etc.). The competency questions falling in the area of multimedia contents, user related information and transversal knowledge (e.g. temporal and spatial information) were mostly addressed by reusing and adapting existing ontologies. Instead, considerable effort was spent to meet the requirements related to the medical domain. In particular, there were no ontologies able to support a robot in performing a Comprehensive Geriatric Assessment of its users. MON filled this gap with the CGA Ontology (cf. Section 3.3.2) which enabled the robot to autonomously perform a Comprehensive Geriatric Assessment (cf. Section 7.4.1.1). Another innovative module of MON is the Affordance ontology (cf. Section 3.3.1) which enables a novel mechanism for arbitrating robot's actions. This model allows to define a series of situations a robot should react to and the most appropriate actions to perform in each situation. Concerning the "sociality" of a robot, much work still to be done to investigate to what extent the robot's knowledge impacts on acceptability, empathy and trustability of robots. Although Social Ontology [188] (i.e. the study of the nature and properties of the social world) is a developed field in philosophy, the results of this research area are closed off to robots. Therefore, future work should focus on encoding social theories in a machine-interpretable format.

RQ2: What Linked Data can provide background knowledge for social robots tasks?

Chapter 4 presented two different lines of research that investigate the possibility of providing Linked Open Data as background knowledge for supporting robots in their daily tasks. The two studies focused on linguistic and common sense knowledge respectively. The high accuracy of the extracted knowledge reveals that LOD is indeed a good source of knowledge for these domains. The generated knowledge supports the applications presented in Chapter 7. The positive impact of the prototype (assessed in three trials) motivates to further extend the investigation to other knowledge domains.

RQ3: How to provide robots with access to knowledge?

Chapter 5 presented Lizard, a framework which is aimed at easing the software development of knowledge-aware systems by filling the gap between Semantic Web technologies and object-oriented applications. Lizard allows application running on robotic frameworks to access RDF facts stored in the knowledge base in a programmatic way. Robotic applications presented in Chapter 7 demonstrate Lizard's benefits, feasibility and limitations when developed, deployed and tested in a real socially assistive scenario. In particular, a current limitation of Lizard is the poor triple-based interaction paradigm offered to applications. This is strongly limiting compared to querying a knowledge base with the possibility of joining several triple patterns in a single SPARQL query.

RQ4: How to integrate robot's knowledge with data coming from robot's experience?

Chapter 6 introduced a novel approach for ontology matching that uses frame semantics as cognitive model for representing the intensional meaning of ontology entities. This method allows to integrate data coming from robot's experience (encoded in an ontology) with information already contained in robot's knowledge base. Aligning ontologies with frames, in particular linguistic frames, has a second non-negligible benefit related to both understanding and generation of natural language. On the one hand, the combination of tools for frame-based semantic role labelling and an alignment mapping linguistic frames on robot's knowledge base schema, enables robot to directly ingest and integrate structured knowledge extracted from text. As far as natural language generation is concerned, it is worth noticing that a linguistic frame (such those defined in FrameNet) often comes with annotated sentences verbalizing some of its instances. We claim that aligning linguistic frames with ontologies could be the key for filling the gap between structured and unstructured knowledge. The proposed approach is being implemented and evaluated, therefore we cannot show results that directly demonstrate the feasibility of the proposed approach. But, good results on classification linked data entities with respect to foundational distinctions by using labels and descriptions of entities demonstrate that textual annotations carry the semantics of entities (cf. Section 4.2). Therefore, we hypothesize that textual annotations in ontology found on the Web are rich enough to be used for the frame ontology alignment task.

RQ5: How to Semantic Web technologies can be orchestrated to support robot tasks?

Chapter 7 presented a component-based software architecture which is aimed at supporting knowledge-intensive tasks performed by social robots with a platform implementing and orchestrating a set of off-the-shelf components. The components of the architecture rely on contributions of this thesis and state-of-the-art semantic web technologies to provide developers with generic functionalities for structuring and accessing knowledge, exploiting linked open data, and sharing knowledge among architectural components. We claim that the prototype presented in this chapter is only an example of robotic systems that benefit of framework proposed in the thesis. This framework could potentially be integrated, with appropriate adaptions, with every autonomous agents (not limited to embodied systems). A prototype of this architecture has been deployed on Kompaï-2 robots, evaluated during supervised trials in different dementia care environments. The findings revealed that the prototype had an important role in the social care of people with dementia and motivate to evaluate the prototype in other contexts.

8.2 Future Work

This thesis concludes with a discussion on possible lines of future work, some of which are underway at the time of this writing.

Facing other Contexts. Although trials in different healthcare settings confirm the validity of the approach, large part of the future activities shall be put in analyzing, adapting, deploying and evaluating the overall framework for other scenarios different from the healthcare context (e.g. education or entertainment). New modeling requirements for the architecture (e.g. new components to be integrated) and for the ontology network (e.g. new modules to be designed) could emerge from new scenarios. Novel information shall be possibly integrated within the knowledge base for supporting robot's activities in the new scenarios. Moreover, new contexts will provide the opportunity to assess feasibility and benefits of the architecture which could be assessed with techniques overviewed in [18].

Pattern-based Interaction with Knowledge Base. Lizard, like other tools for programmatically accessing knowledge bases, enables a triple-based interaction with triple stores, i.e. generated Java methods deal with a single triple at time. Instead, a valuable direction for improving the usability of the API and the interaction with the knowledge base could be enabling a pattern-based interaction. For example, static methods could be generated by Lizard to instantiate an ontology pattern by invoking a single method. In this direction, ontology modularization research field [54, 55, 189] could provide Lizard with techniques to identify patterns in input ontologies.

Extending Framester. Ongoing work is about integrating and linking Framester's linguistic information with other kinds of knowledge, so to provide robots with a richer human-like knowledge base. The following types of knowledge could be valuable for the robot activities: procedural knowledge (e.g [160]), physical knowledge (e.g. [78]), and open-domain common sense knowledge (such that produced in projects like ConceptNet [196] and NELL [148]). Another line of research is on improving linguistic coverage of Framester's frames in cataloging and describing situations. Methods like [176, 163] could be used to extend the lexical units associated with frames, but the main lack of FrameNet-based datasets is the scarcity of semantic roles and semantic types associated with frames. A possible solution could be analyzing statistical correlation (by using tools such as sense embedding [42, 41]) between occurrence of frames and ontology classes within a corpus. We hypothesize that classes statistically correlated with a frame **f** are the semantic types involved in situations described by **f**. Further work (e.g. involving crowdsourcing techniques) is needed to determine semantic roles that entities of the discovered semantic types play in situations described by **f**.

Frame-based Ontology Alignment. In this thesis we introduced a novel approach for ontology matching. This method exploits the frame semantics as cognitive model for representing the intensional meaning of ontology entities. The frame-based representation enabled finding correspondences between ontology entities abstracting from their logical type thus leading a step ahead the state of the art of ontology matching. The proposed approach is being implemented and evaluated. We are evaluating the resulting alignments in a both *direct* and *indirect* way. The benchmarks used for assessing ontology matching systems are mostly unable to evaluate the capability of finding correspondences among ontology entities with different logical types. In order to accomplish this purpose we are extending the existing benchmarks for ontology matching. An example in this direction is [214]. On the other hand, we are using the proposed approach in a question answering system for selecting relevant resources answering a given question. The frame occurrences in a question together with the frame-ontology alignment help in formulate the query over the linked data, hence identifying resources that answer the given question.

Common Sense Knowledge. This thesis reports a set of experiments for assessing whether the Web, and in particular Linked Open Data, provides an empirical basis to extract foundational distinctions, and if they match common sense. For testing the former, we adopt and compare two approaches, namely alignment-based methods and machine learning methods. For the latter we use crowdsourcing and compare the judgements of the crowd with those of experts'. For both questions we observe promising results and define a method that can be generalised to investigate additional distinctions. We plan experiments on other foundational distinctions (e.g. types of locations, objects that can serve as locations or containers, etc.) and with additional methods. In this direction methods proposed in this thesis could be combined with approaches in [23, 106] to extract relational common sense knowledge. The good precision of alignment-based methods (∼90% for both classifications) allows to hypothesize that SENECA output could provide valuable examples for training Machine Learning methods. Our ultimate goal is to advance the state of the art of AI tasks requiring common sense reasoning by designing a methodological framework that enables mass-production of common sense knowledge, and its injection into LOD. To this end effort should be payed on knowledge representation languages and (meta-)models to encode common sense. Challenges for knowledge representation filed come from the need of having suitable languages and models for encoding: *(i) agreement on*, *(ii) evidence of*, and *(iii) validity of* common sense facts. This information is essential for autonomous agents to take decision. Agreement (i.e. the number of people that agree with a stated fact) and evidence (i.e.

other facts that support a given fact) of facts guarantee the trustworthiness of data and the reliability of its decisions. The validity (i.e. where and when a fact is true) guarantees the suitability of context in which the fact is used. Validity and evidence together could enable an agent to assess: if a certain fact apply for the context at a hand, and the degree of similarity between a situation an agent should face and other situations where the fact has proven to be true. Future work shall focus on investigating to what extent available languages and models are suitable to these purposes.

Deeper Evaluation. Various contributions of this thesis need of a deeper evaluation. Robotic applications developed in the case study provided a feedback on the MARIO Ontology Network but more evidences could be gathered to assess the validity of proposed design solutions. As well the methodology developed for designing the MARIO Ontology Network needs of a larger scale evaluation with experts. Concerning Framester, a methodology (hopefully based on crowdsourcing techniques) is needed to assess the linking structure of the resource. We also plan to assess the usability of ontology API generated by Lizard. Experiments shall be performed with application developers both having experience or not with Semantic Web technologies. A possible test could be asking developers to interact with a knowledge base with and without the support of Lizard, and then evaluating limits and benefits with a survey.

Appendix A

Code Generated by Lizard

The complete project generated by Lizard for the ontology Person[1] is available on line at[2].

A.1 Interface

```
public static Action get(String entityURI) {
  Action _entity = null;
  Model model = RuntimeJenaLizardContext.getContext().getModel();
  if (model.contains(
      ModelFactory.createDefaultModel().createResource(entityURI),
      RDF.type,
      ModelFactory.createDefaultModel()
    .createResource(
    "http://www.ontologydesignpatterns.org/ont/mario/action.owl#Action"
    ))) {

  _entity = new ActionJena(
      ModelFactory.createDefaultModel()
          .createResource(entityURI));

  }
```

[1]http://ontologydesignpatterns.org/ont/mario/person.owl

[2]http://etna.istc.cnr.it/mario/lizard/person_ont.zip

```java
  return _entity;
}
```

Frame A.1: The Java code implementing the static method get(String entityURI).

```java
public static Set<Action> getByAction_byAgent() {
  Set<Action> ret = new HashSet<Action>();
  Property predicate = ModelFactory.createDefaultModel()
    .createProperty(ACTION_BY_AGENT);
  Model model = RuntimeJenaLizardContext.getContext().getModel();
  StmtIterator stmtItTemp = model.listStatements(
    null, predicate, ((RDFNode) null));
  Model tempModel = ModelFactory.createDefaultModel().add(stmtItTemp);
  StmtIterator stmtIt = tempModel.listStatements();
  while (stmtIt.hasNext()) {
    Statement stmt = stmtIt.next();
    Resource subj = stmt.getSubject();
    Action individual = new ActionJena(subj);
    ret.add(individual);
  }
  return ret;
}
```

Frame A.2: The Java code implementing the method getByAction_byAgent().

```java
public static Set<Action> getByAction_byAgent(LizardInterface value) {
  Set<Action> ret = new HashSet<Action>();
  Property predicate = ModelFactory.createDefaultModel()
    .createProperty(ACTION_BY_AGENT);
  Model model = RuntimeJenaLizardContext.getContext().getModel();
  StmtIterator stmtItTemp =
    model.listStatements(null, predicate, value.getIndividual());
  Model tempModel = ModelFactory.createDefaultModel().add(stmtItTemp);
  StmtIterator stmtIt = tempModel.listStatements();
  while (stmtIt.hasNext()) {
    Statement stmt = stmtIt.next();
    Resource subj = stmt.getSubject();
    Action individual = new ActionJena(subj);
    ret.add(individual);
  }
  return ret;
}
```

Frame A.3: The Java code implementing the method getByAction_byAgent(LizardInterface value).

A.2 Jena Class

```
public ActionJena(String individual) {
 super(
  ModelFactory.createDefaultModel().createResource(individual),
  ModelFactory.createOntologyModel()
  .createOntResource(
 "http://www.ontologydesignpatterns.org/ont/mario/action.owl#Action"));

 Model model = RuntimeJenaLizardContext.getContext().getModel();

 model.add(
 ((Resource) super.individual),
 RDF.type,
 ModelFactory.createOntologyModel()
 .createOntResource(
 "http://www.ontologydesignpatterns.org/ont/mario/action.owl#Action"));

}
```

Frame A.4: The constructor of the class *Action_ActionJena*.

```
public Set<Agent> getAction_byAgent() {
 Set<Agent> retValue = new HashSet<Agent>();
 Property predicate =
 ModelFactory.createDefaultModel()
 .createProperty(
 "http://www.ontologydesignpatterns.org/ont/mario/action.owl#byAgent");
 Model model = RuntimeJenaLizardContext.getContext().getModel();
 StmtIterator stmtItTemp = model.listStatements(
   ((Resource) super.individual),
   predicate,
   ((RDFNode) null));
 Model tempModel = ModelFactory.createDefaultModel().add(stmtItTemp);
 StmtIterator stmtIt = tempModel.listStatements();
 while (stmtIt.hasNext()) {
  Statement stmt = stmtIt.next();
  RDFNode object = stmt.getObject();
  Agent obj = new AgentJena(object);
  retValue.add(obj);
 }
 return retValue;
}
```

Frame A.5: The method getAction_byAgent() as implemented by the class *Action_ActionJena*.

```
public void setAction_byAgent(Set<Agent> agent) {
 Property predicate =
 ModelFactory.createDefaultModel()
 .createProperty(
 "http://www.ontologydesignpatterns.org/ont/mario/action.owl#byAgent");
 Model model = RuntimeJenaLizardContext.getContext().getModel();
 removeAllAction_byAgent(getAction_byAgent());
 for (Agent object: agent) {
  model.add(((Resource) super.individual), predicate,
     object.getIndividual());
 }
}
```

Frame A.6: The method setAction_byAgent(Set<Agent> agents) as implemented by the class *Action_ActionJena*.

```
public void addAllAction_byAgent(Set<Agent> agent) {
 Property predicate =
  ModelFactory.createDefaultModel()
  .createProperty(
  "http://www.ontologydesignpatterns.org/ont/mario/action.owl#byAgent");
 Model model = RuntimeJenaLizardContext.getContext().getModel();
 for (Agent object: agent) {
  model.add(((Resource) super.individual), predicate,
    object.getIndividual());
 }
}
```

Frame A.7: The method addAllAction_byAgent(Set<Agent> agents) as implemented by the class *Action_ActionJena*.

```
public void removeAllAction_byAgent(Set<Agent> agent) {
 Property predicate = ModelFactory.createDefaultModel()
 .createProperty(
 "http://www.ontologydesignpatterns.org/ont/mario/action.owl#byAgent");
 Model model = RuntimeJenaLizardContext.getContext().getModel();
 for (Agent object: agent) {
  model.remove(((Resource) super.individual), predicate,
      object.getIndividual());
 }
}
```

Frame A.8: The method removeAllAction_byAgent(Set<Agent> agents) as implemented by the class *Action_ActionJena*.

```
public ActionBean asBean() {
 Model model =
 RuntimeJenaLizardContext.getContext().getModel();
 Query query = QueryFactory.create((
  "DESCRIBE <"+(super.individual.asResource().getURI()+">")));
 QueryExecution qexec =
  RuntimeJenaLizardContext.getContext()
    .createQueryExecution(query, model);
 Model m = qexec.execDescribe();
 ActionBean actionJena = new ActionBean();
```

```java
 actionJena.setId(super.individual.asResource().getURI());
 actionJena.setIsCompleted(true);
 actionJena.setSpa_hasPlace(this.getSpa_hasPlace(m, false));
 actionJena.setAction_executesTask(
   this.getAction_executesTask(m, false));
 actionJena.setAction_byAgent(this.getAction_byAgent(m, false));
 actionJena.setTime_atTime(this.getTime_atTime(m, false));
 actionJena.setAction_hasParticipant(
   this.getAction_hasParticipant(m, false));
 return actionJena;
}
```

Frame A.9: The method asBean() as implemented by the class *Action_ActionJena*.

```java
public ActionBean asMicroBean() {
 Model model = RuntimeJenaLizardContext.getContext().getModel();
 Query query = QueryFactory.create(
  ("DESCRIBE <"+(super.individual.asResource().getURI()+">")));
 QueryExecution qexec = RuntimeJenaLizardContext.getContext()
  .createQueryExecution(query, model);
 Model m = qexec.execDescribe();
 ActionBean actionJena = new ActionBean();
 actionJena.setId(super.individual.asResource().getURI());
 actionJena.setIsCompleted(true);
 actionJena.setSpa_hasPlace(this.getSpa_hasPlace(m, true));
 actionJena.setAction_executesTask(
    this.getAction_executesTask(m, true));
 actionJena.setAction_byAgent(this.getAction_byAgent(m, true));
 actionJena.setTime_atTime(this.getTime_atTime(m, true));
 actionJena.setAction_hasParticipant(
    this.getAction_hasParticipant(m, true));
 return actionJena;
}
```

Frame A.10: The method asMicroBean() as implemented by the class *Action_ActionJena*.

```
private Set<Agent>
  getAction_byAgent(Model model, Boolean isForMicroBean) {
 Set<Agent> retValue = new HashSet<Agent>();
 Property predicate =
 ModelFactory.createDefaultModel()
 .createProperty(
  "http://www.ontologydesignpatterns.org/ont/mario/action.owl#byAgent");
 StmtIterator stmtIt = model.listStatements(
   ((Resource) super.individual), predicate, ((RDFNode) null));
 while (stmtIt.hasNext()) {
  Statement stmt = stmtIt.next();
  RDFNode object = stmt.getObject();
  Agent obj = null;
  if (isForMicroBean) {
    obj = new AgentBean(object);
    obj.setId(object.asResource().getURI());
    obj.setIsCompleted(false);
  } else {
    obj = new AgentJena(object);
  }
  retValue.add(obj);
 }
 return retValue;
}
```

Frame A.11: The private service method getAction_byAgent of the class *Action_ActionJena*.

A.3 Bean Class

```
public class ActionBean extends InMemoryLizardClass
   implements Action, Thing {

private String id;
private Boolean isCompleted;
private Set<SpatialThing> spa_hasPlace;
private Set<TemporalEntity> time_atTime;
private Set<Agent> action_byAgent;
private Set<Task> action_executesTask;
private Set<Agent> action_hasParticipant;

public ActionBean() {
  spa_hasPlace = new HashSet<SpatialThing>();
  action_executesTask = new HashSet<Task>();
  action_byAgent = new HashSet<Agent>();
  time_atTime = new HashSet<TemporalEntity>();
  action_hasParticipant = new HashSet<Agent>();
}
....
@Override
public void removeAllAction_byAgent(Set<Agent> agent) {
  action_byAgent.removeAll(agent);
}
@Override
public void setAction_byAgent(Set<Agent> agent) {
  action_byAgent = agent;
}
@Override
public Set<Agent> getAction_byAgent() {
  return action_byAgent;
}
@Override
public void addAllAction_byAgent(Set<Agent> agent) {
  action_byAgent.addAll(agent);
}
....
}
```

Frame A.12: An excerpt of the Bean Class for `action:Action`.

A.4 REST API Description

```
{
  “basePath”: “/org_ontologydesignpatterns_ont_mario_action_owl”,
  “paths”: {
    “/action_action”: {
      “get”: {
        “summary”: “Retrieve individuals that belong to action”,
        ....
      }},
    “/action_action/action_byAgent”: {
      “post”: {
      “summary”: “Add the value to the action_byAgent of the object
      identified by the iri passed as parameter.”,
      .....
      “parameters”: [
      {
        “in”: “query”,
        “name”: “iri”,
        “description”: “iri”,
        “type”: “string”,
        “required”: true
      },{
        “in”: “query”,
        “name”: “value”,
        “description”: “value to be set”,
        “type”: “string”,
        “required”: true }]
    },
      “get”: {
        “summary”: “Get the action_byAgent of the individual identified
        by the iri passed as parameter.”,
        “responses”: {
          “200”: {
            “schema”: {
              “type”: “array”,
              "items": {"$ref": "#/definitions/Agent"}
              },
           "description": "successful operation"
         },
        "404": {"description": "Not found"}
        },
        ...
    },
      "delete": {
        "summary": "Remove the value from the action_byAgent of the
         object identified by the the iri passed as parameter.",
         ....
      },
      "put": {
        "summary": "Set the value as the action_byAgent of the object
         identified by the the iri passed as parameter.",
         ...
       }
     },
    "/action_action/having/action_byAgent": {
       ....
    },
    "/action_action/action_executesTask": {
      ...
    },
    "/action_action/having/action_executesTask": {
      ...
    },
    "/action_task": {
      ....
    }
  "definitions": {
    "Action": {
      "properties": {
        "action_byAgent": {
          "uniqueItems": true,
            ”type”: “array”,
            “items”: {“$ref”: “#/definitions/Agent”}
          },
        “time_atTime”: {...},
        “action_executesTask”: {...},
        }}
...
}}
....
}}
```

Frame A.13: An excerpt of the REST API description (in Swagger language) for Action ontology module of the Mario Ontology Network.

Bibliography

[1] Eneko Agirre and Aitor Soroa. "Personalizing PageRank for Word Sense Disambiguation". In: *Proceedings of the 12th conference of the European chapter of the Association for Computational Linguistics (EACL 2009).* (Athens, Greece). Edited by Alex Lascarides, Claire Gardent and Joakim Nivre. The Association for Computer Linguistics, 2009, pages 33–41.

[2] Jose M. Alcaraz Calero, Jorge Bernal Bernabé, Juan M. Marin Perez, Diego Sevilla Ruiz, Felix J. Garcia Clemente and Gregorio Martínez Pérez. "Towards an Architecture to bind the Java and OWL languages". In: *Journal of Research and Practice in Information Technology* 44.1 (2012), pages 17–41.

[3] George S. Alexopoulos, Robert C. Abrams, Robert C. Young and Charles A. Shamoian. "Cornell scale for depression in dementia". In: *Biological psychiatry* 23.3 (1988), pages 271–284.

[4] Carlo Allocca, Mathieu d'Aquin and Enrico Motta. "DOOR - Towards a Formalization of Ontology Relations". In: *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD 2009).* (Funchal - Madeira, Portugal). Edited by Jan L. G. Dietz. 2009, pages 13–20.

[5] Ronald C. Arkin. *Behavior-based robotics.* MIT press, 1998.

[6] Lora Aroyo and Chris Welty. "Truth Is a Lie: Crowd Truth and the Seven Myths of Human Annotation". In: *AI Magazine* 36.1 (2015), pages 15–24.

[7] Luigi Asprino. “Addressing Knowledge Integration with a Frame-Driven Approach”. In: *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management and Satellite Events, EKM and Drift-a-LOD (EKAW 2016).* (Bologna, Italy). Edited by Paolo Ciancarini, Francesco Poggi, Matthew Horridge, Jun Zhao, Tudor Groza, Mari Carmen Suarez-Figueroa, Mathieu d’Aquin and Valentina Presutti. Springer International Publishing, 2016, pages 205–210. DOI: 10.1007/978-3-319-58694-6_32.

[8] Luigi Asprino, Valerio Basile, Paolo Ciancarini and Valentina Presutti. “Empirical Analysis of Foundational Distinctions in Linked Open Data”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 18).* Edited by Jérôme Lang. International Joint Conferences on Artificial Intelligence, 2018, pages 3962–3969. DOI: 10.24963/ijcai.2018/551.

[9] Luigi Asprino, Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Diego Reforgiato Recupero and Alessandro Russo. “Autonomous Comprehensive Geriatric Assessment”. In: *Proceedings of the 1st International Workshop on Application of Semantic Web technologies in Robotics co-located with 14th ESWC (ANSWER 2017).* (Portroz, Slovenia). Edited by Emanuele Bastianelli, Mathieu d’Aquin and Daniele Nardi. 2017, pages 41–45.

[10] Luigi Asprino, Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti and Alessandro Russo. “A Knowledge Management System for Assistive Robotics”. In: *Proceedings of the 1st International Workshop on Application of Semantic Web technologies in Robotics co-located with 14th ESWC (ANSWER 2017).* (Portroz, Slovenia). Edited by Emanuele Bastianelli, Mathieu d’Aquin and Daniele Nardi. 2017, pages 46–50.

[11] Luigi Asprino, Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti and Alessandro Russo. “Knowledge-driven Support for Reminiscence on Companion Robots”. In: *Proceedings of the 1st International Workshop on Application of Semantic Web technologies in Robotics co-located with 14th*

ESWC (ANSWER 2017). (Portroz, Slovenia). Edited by Emanuele Bastianelli, Mathieu d'Aquin and Daniele Nardi. 2017, pages 51–55.

[12] Luigi Asprino, Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero and Alessandro Russo. "Ontology-based Knowledge Management for Comprehensive Geriatric Assessment and Reminiscence Therapy on Social Robots". In: *Data Science for Healthcare: Methodologies and Applications*. Edited by Sergio Consoli, Diego Reforgiato Recupero and Milan Petkovic. Springer, 2018, (to appear).

[13] Luigi Asprino, Andrea Giovanni Nuzzolese, Alessandro Russo, Aldo Gangemi, Valentina Presutti and Stefano Nolfi. "An Ontology Design Pattern for supporting behaviour arbitration in cognitive agents". In: *Advances in Ontology Design and Patterns*. Edited by Karl Hammar, Pascal Hitzler, Adila Krisnadhi, Agnieszka Ławrynowicz, Andrea Giovanni Nuzzolese and Monika Solanki. IOS Press, 2017, pages 85–95.

[14] Luigi Asprino, Valentina Presutti and Aldo Gangemi. "Matching Ontologies Using a Frame-Driven Approach". In: *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management and Satellite Events, EKM and Drift-a-LOD (EKAW 2016)*. (Bologna, Italy). Edited by Paolo Ciancarini, Francesco Poggi, Matthew Horridge, Jun Zhao, Tudor Groza, Mari Carmen Suarez-Figueroa, Mathieu d'Aquin and Valentina Presutti. Springer International Publishing, 2016, pages 101–104. DOI: 10.1007/978-3-319-58694-6_9.

[15] Luigi Asprino, Valentina Presutti, Aldo Gangemi and Paolo Ciancarini. "Frame-Based Ontology Alignment". In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence and the Twenty-Ninth Innovative Applications of Artificial Intelligence Conference (AAAI 2017)*. (San Francisco, California USA). Edited by Satinder P. Singh and Shaul Markovitch. AAAI Press, Palo Alto, California, 2017, pages 4095–4096.

[16] Mark van Assem, Aldo Gangemi and Guus Schreiber. "Conversion of WordNet to a standard RDF/OWL representation". In: *Proceedings of the 5th International Conference on Language Resources and Evaluation, (LREC 2006).* Edited by Nicoletta Calzolari, Khalid Choukri, Aldo Gangemi, Bente Maegaard, Joseph Mariani, Jan Odijk and Daniel Tapias. 2006, pages 237–242.

[17] Franz Baader, Ian Horrocks and Ulrike Sattler. "Description Logics". In: edited by Frank van Harmelen, Vladimir Lifschitz and Bruce W. Porter. Volume 3. Foundations of Artificial Intelligence. Elsevier, 2008, pages 135–179. ISBN: 978-0-444-52211-5. DOI: 10.1016/S1574-6526(07)03003-9.

[18] Muhammad Ali Babar and Ian Gorton. "Comparison of Scenario-Based Software Architecture Evaluation Methods". In: *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC 2004).* 2004, pages 600–607. DOI: 10.1109/APSEC.2004.38.

[19] Collin F. Baker, Charles J. Fillmore and John B. Lowe. "The Berkeley FrameNet Project". In: *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998).* (Montréal, Quebec, Canada). Edited by Christian Boitet and Pete Whitelock. Morgan Kaufmann Publishers / ACL, 1998, pages 86–90.

[20] Antonio Bandera, Juan Pedro Bandera, Pablo Bustos, Luis V. Calderita, Álvaro Dueñas, Fernando Fernández, Raquel Fuentetaja, Angel García-Olaya, Francisco Javier García-Polo, Jose Carlos González, Ana Iglesias, Luis J. Manso, Rebeca Marfil, José Carlos Pulido, Christian Reuther, Adrián Romero-Garcés and Cristina Suárez. "CLARC: a Robotic Architecture for Comprehensive Geriatric Assessment". In: *17th Workshop of Physical Agents (WAF 2016).* 2016, pages 1–8.

[21] Christoph Bartneck and Jodi Forlizzi. "A design-centred framework for social human-robot interaction". In: *Proceedings of the 13th IEEE International*

Workshop on Robot and Human Interactive Communication (ROMAN 2004). IEEE. 2004, pages 591–594.

[22] Valerio Basile, Elena Cabrio and Claudia Schon. "KNEWS: Using Logical and Lexical Semantics to Extract Knowledge from Natural Language". In: *Proceedings of the European Conference on Artificial Intelligence (ECAI 2016)*. 2016.

[23] Valerio Basile, Soufian Jebbara, Elena Cabrio and Philipp Cimiano. "Populating a Knowledge Base with Object-Location Relations Using Distributional Semantics". In: *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2016)*. (Bologna, Italy). Edited by Eva Blomqvist, Paolo Ciancarini, Francesco Poggi and Fabio Vitali. Springer International Publishing, 2016, pages 34–50. DOI: 10.1007/978-3-319-49004-5_3.

[24] Emanuele Bastianelli, Mathieu d'Aquin and Daniele Nardi, editors. *Proceedings of the 1st International Workshop on Application of Semantic Web technologies in Robotics co-located with 14th ESWC (ANSWER 2017)*. (Portroz, Slovenia). 2017.

[25] Bénédicte Batrancourt, Michel Dojat, Bernard Gibaud and Gilles Kassel. "A core ontology of instruments used for neurological, behavioral and cognitive assessments". In: *Proceedings of the 6th International Conference on Formal Ontology in Information Systems (FOIS 2010)*. (Toronto, Canada). 2010, pages 185–198.

[26] Wouter Beek, Laurens Rietveld, Stefan Schlobach and Frank van Harmelen. "LOD Laundromat: Why the Semantic Web Needs Centralization (Even If We Don't Like It)". In: *IEEE Internet Computing* 20.2 (2016), pages 78–81. DOI: 10.1109/MIC.2016.43.

[27] Roger Bemelmans, Gert Jan Gelderblom, Pieter Jonker and Luc de Witte. "Socially Assistive Robots in Elderly Care: A Systematic Review into Effects

and Effectiveness". In: *Journal of the American Medical Directors Association* 13.2 (2012), pages 114–120. DOI: 10.1016/j.jamda.2010.10.002.

[28] Tim Berners-Lee, James Hendler and Ora Lassila. "The semantic web". In: *Scientific american* 284.5 (2001), pages 34–43.

[29] David L. Bisset. *D1.1 MARIO System Specification for Pilot 1*. 2015. URL: http://www.mario-project.eu/portal/images/deliverables/D1.1-MARIO-System-Specification-for-Pilot%201.pdf.

[30] Christian Bizer. "D2R MAP - A Database to RDF Mapping Language". In: *Proceedings of the 12th International World Wide Web Conference (WWW 2003) - Poster*. (Budapest, Hungary). Edited by Irwin King and Tamás Máray. 2003.

[31] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak and Sebastian Hellmann. "DBpedia - A crystallization point for the Web of Data". In: *International Journal of Web Semantics* 7.3 (2009), pages 154–165. DOI: 10.1016/j.websem.2009.07.002.

[32] Mary R. Bliss, Rhoda McLaren and Arthur N. Exton-Smith. "Mattresses for preventing pressure sores in geriatric patients". In: *Monthly bulletin of the Ministry of Health and the Public Health Laboratory Service* 25 (1966), pages 238–268.

[33] Eva Blomqvist, Valentina Presutti, Enrico Daga and Aldo Gangemi. "Experimenting with eXtreme Design". In: *Proceedings of the 17th International Conference on Knowledge Engineering and Management by the Masses (EKAW 2010)*. (Lisbon, Portugal). Edited by Philipp Cimiano and Helena Sofia Pinto. Volume 6317. Lecture Notes in Computer Science. Springer, 2010, pages 120–134. DOI: 10.1007/978-3-642-16438-5_9.

[34] Olivier Bodenreider. "The unified medical language system (UMLS): integrating biomedical terminology". In: *Nucleic acids research* 32.suppl 1 (2004), pages 267–270.

[35] Matt-Mouley Bouamrane, Alan Rector and Martin Hurrell. “Gathering Precise Patient Medical History with an Ontology-Driven Adaptive Questionnaire”. In: *Proceedings of 21st IEEE International Symposium on Computer-Based Medical Systems*. 2008, pages 539–541. DOI: 10.1109/CBMS.2008.24.

[36] Matt-Mouley Bouamrane, Alan Rector and Martin Hurrell. “Development of an ontology for a preoperative risk assessment clinical decision support system”. In: *Proceedings of 22nd IEEE International Symposium on Computer-Based Medical Systems*. 2009, pages 1–6.

[37] Cynthia Breazeal. “Toward sociable robots”. In: *Robotics and Autonomous Systems* 42.3 (2003), pages 167–175. ISSN: 0921-8890. DOI: 10.1016/S0921-8890(02)00373-1.

[38] Cynthia Lynn Breazeal. “Sociable machines: Expressive social exchange between humans and robots”. PhD thesis. Massachusetts Institute of Technology, 2000.

[39] Joost Broekens, Marcel Heerink and Henk Rosendal. “Assistive social robots in elderly care: a review”. In: *Gerontechnology* 8.2 (2009), pages 94–103.

[40] Jeen Broekstra, Arjohn Kampman and Frank Van Harmelen. “Sesame: A generic architecture for storing and querying rdf and rdf schema”. In: *Proceedings of the 1st International Semantic Web Conference (ISWC 2002)*. Springer, 2002, pages 54–68.

[41] José Camacho-Collados and Mohammad Taher Pilehvar. “From Word to Sense Embeddings: A Survey on Vector Representations of Meaning”. In: *Journal of Artificial Intelligence Research* (2018), to appear.

[42] José Camacho-Collados, Mohammad Taher Pilehvar and Roberto Navigli. “Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities”. In: *Artificial Intelligence* 240 (2016), pages 36–64. ISSN: 0004-3702. DOI: 10.1016/j.artint.2016.07.005.

[43] Lola Canamero and Jakob Fredslund. “I Show You How I Like You: Emotional Human-Robot Interaction through Facial Expression and Tactile Stimulation”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 31.5 (2001), pages 454–459. DOI: 10.1109/3468.952719.

[44] Hoang-Long Cao, Pablo Gomez Esteban, Albert De Beir, Ramona Simut, Greet Van De Perre and Bram Vanderborght. “A platform-independent robot control architecture for multiple therapeutic scenarios”. In: *Proceedings of 5th International Symposium on New Frontiers in Human-Robot Interaction (NF-HRI 2016).* (Sheffield, United Kingdom). Edited by Maha Salem, Astrid Weiss, Paul Baxter and Kerstin Dautenhahn. The Association for Computer Linguistics, 2016.

[45] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr. and Tom M. Mitchell. “Toward an Architecture for Never-Ending Language Learning”. In: *Proceedings of the 24th Conference on Artificial Intelligence (AAAI 2010).* (Atlanta, Georgia, USA). AAAI Press, 2010, pages 1306–1313.

[46] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne and Kevin Wilkinson. “Jena: implementing the semantic web recommendations”. In: *Proceedings of the 13th international World Wide Web conference (WWW 2004) - Alternate track papers & posters.* 2004, pages 74–83. DOI: 10.1145/1013367.1013381.

[47] Dympna Casey, Timur Beyan, Mary Kagkoura, Thomas Messervey, Andy Bleaden, Grazia D’Onofrio and Daniele Sancarlo. *D1.3 Assessment Methodology.* EU Project Deliverable. 2015.

[48] Dympna Casey, Megan Burke, Tanja Kovacic, Keith Cortis, Kathy Murphy, Aisling Dolan, Grazia D’Onofrio, Daniele Sancarlo, Francesco Ricciardi, Aimee Teare, Massimiliano Raciti and Rubén Alonso. *D8.3 Evidence of Service Robots Benefits.* EU Project Deliverable. 2018.

[49] Abdelghani Chibani, Yacine Amirat, Samer Mohammed, Eric Matson, Norihiro Hagita and Marcos Barreto. "Ubiquitous robotics: Recent challenges and future trends". In: *Robotics and Autonomous Systems* 61.11 (2013), pages 1162–1172. ISSN: 0921-8890. DOI: 10.1016/j.robot.2013.04.003.

[50] Jiska Cohen-Mansfield, Maha Dakheel-Ali and Marcia S. Marx. "Engagement in persons with dementia: the concept and its measurement". In: *The American journal of geriatric psychiatry* 17.4 (2009), pages 299–307.

[51] Yeates Conwell, Nicholas T. Forbes, Christopher Cox and Eric D. Caine. "Validation of a measure of physical illness burden at autopsy: the Cumulative Illness Rating Scale". In: *Journal of the American Geriatrics Society* 41.1 (1993), pages 38–41.

[52] Francesco Corcoglioniti, Marco Rospocher, Alessio Palmero Aprosio and Sara Tonelli. "PreMOn: a Lemon Extension for Exposing Predicate Models as Linked Data". In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016).* (Portorož, Slovenia). Edited by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk and Stelios Piperidis. European Language Resources Association (ELRA), 2016.

[53] Anneli Dahlström, Fredrik Heintz, Martin Jacobsson, Johan Thapper and Martin Öberg. "The NOAI Team Description". In: *RoboCup 2000: Robot Soccer World Cup IV.* Edited by Peter Stone, Tucker R. Balch and Gerhard K. Kraetzschmar. Volume 2019. Springer, 2000, pages 413–416. ISBN: 3-540-42185-8. DOI: 10.1007/3-540-45324-5_49.

[54] Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt and Marta Sabou. "Ontology Modularization for Knowledge Selection: Experiments and Evaluations". In: *Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA 2007).* (Regensburg, Germany). Edited by Roland R. Wagner, Norman Revell and Günther Pernul. Volume 4653.

Lecture Notes in Computer Science. Springer, 2007, pages 874–883. ISBN: 978-3-540-74467-2. DOI: 10.1007/978-3-540-74469-6_85.

[55] Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt and Marta Sabou. "Criteria and Evaluation for Ontology Modularization Techniques". In: edited by Heiner Stuckenschmidt, Christine Parent and Stefano Spaccapietra. Volume 5445. Lecture Notes in Computer Science. Springer, 2009, pages 67–89. ISBN: 978-3-642-01906-7. DOI: 10.1007/978-3-642-01907-4_4. URL: https://doi.org/10.1007/978-3-642-01907-4.

[56] Kerstin Dautenhahn. "The art of designing socially intelligent agents: Science, fiction, and the human in the loop". In: *Applied artificial intelligence* 12.7-8 (1998), pages 573–617.

[57] Kerstin Dautenhahn, Sarah Woods, Christina Kaouri, Michael L. Walters, Kheng Lee Koay and Iain Werry. "What is a robot companion - friend, assistant or butler?" In: *International Conference on Intelligent Robots and Systems (IROS 2005).* (Edmonton, Alberta, Canada). 2005, pages 1192–1197. DOI: 10.1109/IROS.2005.1545189.

[58] Donald Davidson. "The Logical Form of Action Sentences". In: *The Logic of Decision and Action.* Edited by Nicholas Rescher. University of Pittsburgh Press, 1967.

[59] Ernest Davis and Gary Marcus. "Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence". In: *Communications of the ACM* 58.9 (2015), pages 92–103. ISSN: 0001-0782. DOI: 10.1145/2701413.

[60] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi and Riccardo Rosati. "Planning with Sensing for a Mobile Robot". In: *Proceedings of the 4th European Conference on Planning (ECP 1997).* Edited by Sam Steel and Rachid Alami. Volume 1348. Springer, 1997, pages 156–168. ISBN: 3-540-63912-8. DOI: 10.1007/3-540-63912-8_83.

[61] Brian R. Duffy. "The Social Robot". PhD thesis. University College Dublin, 2000.

[62] Brian R. Duffy. “Anthropomorphism and the social robot”. In: *Robotics and Autonomous Systems* 42.3 (2003), pages 177–190. DOI: 10.1016/S0921-8890(02)00374-3.

[63] Brian R. Duffy, Gina Joue and John Bourke. “Issues in assessing performance of social robots”. In: *Proceedings of the 2nd WSEAS International Conference (RODLICS 2002)*. 2002.

[64] Brian R. Duffy, Colm Rooney, Greg MP O’Hare and Ruadhan O’Donoghue. “What is a Social Robot?” In: *Proceedings of the 10th Irish Conference on Artificial Intelligence & Cognitive Science*. 1999.

[65] Maud Ehrmann, Francesco Cecconi, Daniele Vannella, John Philip McCrae, Philipp Cimiano and Roberto Navigli. “Representing Multilingual Data as Linked Data: the Case of BabelNet 2.0”. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*. (Reykjavik, Iceland). Edited by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk and Stelios Piperidis. 2014, pages 401–408. ISBN: 978-2-9517408-8-4.

[66] European Coordination Hub for Open Robotics Development (ECHORD++). *Robotics for the Comprehensive Geriatric Assessment (CGA) Challenge*. 2015. URL: http://echord.eu/portal/ProposalDocuments/download/9.

[67] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Berlin, Germany: Springer, 2013. DOI: 10.1007/978-3-642-38721-0.

[68] Joerg Evermann and Yair Wand. “Ontology based object-oriented domain modelling: fundamental concepts”. In: *Requirements Engineering* 10.2 (2005), pages 146–160. DOI: 10.1007/s00766-004-0208-2.

[69] Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit and George J. Pappas. “Temporal logic motion planning for dynamic robots”. In: *Automatica* 45.2 (2009), pages 343–352. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2008.08.008.

[70] Jing Fan, Dayi Bian, Zhi Zheng, Linda Beuscher, Paul A. Newhouse, Lorraine C. Mion and Nilanjan Sarkar. "A Robotic Coach Architecture for Elder Care (ROCARE) Based on Multi-User Engagement Models". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.8 (2017), pages 1153–1163. ISSN: 1534-4320. DOI: 10.1109/TNSRE.2016.2608791.

[71] Juan Fasola and Maja J. Matarić. "A Socially Assistive Robot Exercise Coach for the Elderly". In: *Journal of Human-Robot Interaction* 2.2 (2013), pages 3–32. ISSN: 2163-0364. DOI: 10.5898/JHRI.2.2.Fasola.

[72] David Feil-Seifer and Maja J. Matarić. "Socially Assistive Robotics". In: *IEEE Robotics & Automation Magazine* 18.1 (2011), pages 24–31. ISSN: 1070-9932. DOI: 10.1109/MRA.2010.940150.

[73] Javier D. Fernández, Wouter Beek, Miguel A. Martínez-Prieto and Mario Arias. "LOD-a-lot - A Queryable Dump of the LOD Cloud". In: *Proceedings of the 16th International Semantic Web Conference (ISWC 2017), Part II.* (Vienna, Austria). Edited by Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange and Jeff Heflin. Springer, 2017, pages 75–83. ISBN: 978-3-319-68203-7. DOI: 10.1007/978-3-319-68204-4_7.

[74] Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez, Axel Polleres and Mario Arias. "Binary RDF Representation for Publication and Exchange (HDT)". In: *International Journal of Web Semantics* 19 (2013), pages 22–41. DOI: 10.1016/j.websem.2013.01.002.

[75] Charles J. Fillmore. "Frame semantics". In: *Linguistics in the Morning Calm.* Edited by Linguistic Society of Korea. Hanshin Publishing Co., 1982, pages 111–137. DOI: 10.1016/B0-08-044854-2/00424-7.

[76] Marshal F. Folstein, Susan E. Folstein and Paul R. McHugh. ""Mini-mental state": a practical method for grading the cognitive state of patients for the clinician". In: *Journal of psychiatric research* 12.3 (1975), pages 189–198.

[77] Terrence Fong, Illah R. Nourbakhsh and Kerstin Dautenhahn. “A survey of socially interactive robots”. In: *Robotics and Autonomous Systems* 42.3-4 (2003), pages 143–166. DOI: 10.1016/S0921-8890(02)00372-X.

[78] Maxwell Forbes and Yejin Choi. “Verb Physics: Relative Physical Knowledge of Actions and Objects”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017).* (Vancouver, Canada). Edited by Regina Barzilay and Min-Yen Kan. Association for Computational Linguistics, 2017, pages 266–276. ISBN: 978-1-945626-75-3. DOI: 10.18653/v1/P17-1025.

[79] Jannik Fritsch, Marcus Kleinehagenbrock, Axel Haasch, Sebastian Wrede and Gerhard Sagerer. “A Flexible Infrastructure for the Development of a Robot Companion with Extensible HRI-Capabilities”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005).* (Barcelona, Spain). IEEE, 2005, pages 3408–3414. DOI: 10.1109/ROBOT.2005.1570637.

[80] Aldo Gangemi. “What’s in a Schema?” In: *Ontology and the Lexicon: A Natural Language Processing Perspective.* Cambridge, UK: Cambridge University Press, 2010, pages 144–182.

[81] Aldo Gangemi, Mehwish Alam, Luigi Asprino, Valentina Presutti and Diego Reforgiato Recupero. “Framester: A Wide Coverage Linguistic Linked Data Hub”. In: *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2016).* (Bologna, Italy). Edited by Eva Blomqvist, Paolo Ciancarini, Francesco Poggi and Fabio Vitali. Springer International Publishing, 2016, pages 239–254. DOI: 10.1007/978-3-319-49004-5_16.

[82] Aldo Gangemi, Nicola Guarino, Claudio Masolo and Alessandro Oltramari. “Sweetening WORDNET with DOLCE”. In: *AI Magazine* 24.3 (2003), pages 13–24.

[83] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari and Luc Schneider. “Sweetening Ontologies with DOLCE”. In: *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002).* (Sigüenza, Spain). Edited by Asunción Gómez-Pérez and V. Richard Benjamins. Springer Berlin Heidelberg, 2002, pages 166–181. ISBN: 978-3-540-45810-4. DOI: 10.1007/3-540-45810-7_18.

[84] Aldo Gangemi and Peter Mika. “Understanding the Semantic Web through Descriptions and Situations”. In: *Proceedings of Confederated International Conferences, CoopIS, DOA, and ODBASE.* (Catania, Sicily, Italy). Edited by Robert Meersman, Zahir Tari and Douglas C. Schmidt. 2003, pages 689–706. ISBN: 3-540-20498-9. DOI: 10.1007/978-3-540-39964-3_44.

[85] Aldo Gangemi, Roberto Navigli and Paola Velardi. “The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet”. In: *Proceedings of Confederated International Conferences, CoopIS, DOA, and ODBASE.* (Catania, Sicily, Italy). Edited by Robert Meersman, Zahir Tari and Douglas C. Schmidt. 2003, pages 3–7. ISBN: 3-540-20498-9.

[86] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti and Paolo Ciancarini. “Automatic Typing of DBpedia Entities”. In: *Proceedings of the 11th International Semantic Web Conference (ISWC 2012), Part I.* (Boston, MA, USA). Edited by Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein and Eva Blomqvist. Volume 7649. Lecture Notes in Computer Science. Springer, 2012, pages 65–81. DOI: 10.1007/978-3-642-35176-1_5.

[87] Aldo Gangemi, Domenico Pisanelli and Geri Steve. “Ontology integration: Experiences with medical terminologies”. In: *Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS 1998).* (Rome,Italy). Volume 46. 1998, pages 98–94.

[88] Aldo Gangemi and Valentina Presutti. “Ontology Design Patterns”. In: *Handbook on Ontologies.* Edited by Steffen Staab and Rudi Studer. Springer, 2009, pages 221–243. ISBN: 978-3-540-70999-2. DOI: 10.1007/978-3-540-92673-3_10.

[89] Aldo Gangemi and Valentina Presutti. “Towards a pattern science for the Semantic Web”. In: *Semantic Web* 1.1-2 (2010), pages 61–68. DOI: 10.3233/SW-2010-0020.

[90] Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio and Misael Mongiovì. “Semantic Web Machine Reading with FRED”. In: *Semantic Web* 8.6 (2017), pages 873–893. DOI: 10.3233/SW-160240.

[91] James Gibson. “The theory of affordances”. In: *Perceiving, acting, and knowing: Toward an ecological psychology.* Edited by Robert Shaw and John Bransford. Lawrence Erlbaum Associates, 1977, pages 67–82.

[92] Ben Goertzel. “Perception Processing for General Intelligence: Bridging the Symbolic/Subsymbolic Gap”. In: *Artificial General Intelligence.* Edited by Joscha Bach, Ben Goertzel and Matthew Iklé. Springer Berlin Heidelberg, 2012, pages 79–88.

[93] Ben Goertzel. “Artificial General Intelligence: Concept, State of the Art, and Future Prospects”. In: *Journal of Artificial General Intelligence* 5.1 (2014), pages 1–48.

[94] Horst-Michael Gross, Christof Schroeter, Steffen Mueller, Michael Volkhardt, Erik Einhorn, Andreas Bley, Tim Langner, Christian Martin and Matthias Merten. “I’ll keep an eye on you: Home robot companion for elderly people with cognitive impairment”. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics.* 2011, pages 2481–2488. DOI: 10.1109/ICSMC.2011.6084050.

[95] Horst-Michael Gross, Christof Schroeter, Steffen Müller, Michael Volkhardt, Erik Einhorn, Andreas Bley, Tim Langner, Matthias Merten, Claire A. G. J. Huijnen, Herjan van den Heuvel and Andreas van Berlo. “Further progress

towards a home robot companion for people with mild cognitive impairment". In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2012).* (Seoul, South Korea). 2012. DOI: 10.1109/ICSMC.2012.6377798.

[96] Horst-Michael Gross, Christof Schröter, Steffen Müller, Michael Volkhardt, Erik Einhorn, Andreas Bley, Christian Martin, Tim Langner and Matthias Merten. "Progress in developing a socially assistive mobile home robot companion for the elderly with mild cognitive impairment". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011).* 2011, pages 2430–2437. ISBN: 978-1-61284-454-1. DOI: 10.1109/IROS.2011.6094770.

[97] Thomas R. Gruber. "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2 (1993), pages 199–220. DOI: 10.1006/knac.1993.1008.

[98] Michael Grüninger and Mark S. Fox. In: *Benchmarking – Theory and Practice.* Edited by Asbjørn Rolstadås. Springer, 1995. Chapter The Role of Competency Questions in Enterprise Engineering, pages 22–31. DOI: 10.1007/978-0-387-34847-6_3.

[99] Barbara Hammer and Pascal Hitzler, editors. *Perspectives of Neural-Symbolic Integration.* Volume 77. Springer, 2007.

[100] Patrick Hayes. "The Second Naive Physics Manifesto". In: *Readings in qualitative reasoning about physical systems.* Edited by Daniel S. Weld and Johan de Kleer. San Francisco, CA, USA: Morgan Kaufmann, 1989. ISBN: 0-262-62101-0.

[101] Jochen Hirth, Norbert Schmitz and Karsten Berns. "Towards Social Robots: Designing an Emotion-Based Architecture". In: *International Journal of Social Robotics* 3.3 (2011), pages 273–290. DOI: 10.1007/s12369-010-0087-2.

[102] Matthew Horridge and Sean Bechhofer. "The owl api: A java api for owl ontologies". In: *Semantic Web* 2.1 (2011), pages 11–21.

[103] "Human–robot interaction: a survey". In: *Foundations and Trends in Human–Computer Interaction* 1.3 (2008), pages 203–275. DOI: 10.1561/1100000005.

[104] Julie Huschilt and Laurie Clune. "The Use of Socially Assistive Robots for Dementia Care". In: *Journal of Gerontological Nursing* 38.10 (2012), pages 15–19. DOI: 10.3928/00989134-20120911-02.

[105] Chandimal Jayawardena, I-Han Kuo, Elizabeth Broadbent and Bruce A. MacDonald. "Socially Assistive Robot HealthBot: Design, Implementation, and Field Trials". In: *IEEE Systems Journal* 10.3 (2016), pages 1056–1067. DOI: 10.1109/JSYST.2014.2337882.

[106] Soufian Jebbara, Valerio Basile, Elena Cabrio and Philipp Cimiano. "Extracting common sense knowledge via triple ranking using supervised and unsupervised distributional models". In: *Semantic Web* (2018), to appear. DOI: 10.3233/SW-180302.

[107] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. "LogMap: Logic-Based and Scalable Ontology Matching". In: *Proceedings fo the 10th International Semantic Web Conference (ISWC 2011), Part I.* (Bonn, Germany). Edited by Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy and Eva Blomqvist. Volume 7031. Lecture Notes in Computer Science. Springer, 2011, pages 273–288. DOI: 10.1007/978-3-642-25073-6_18.

[108] Aditya Kalyanpur, Daniel Jiménez Pastor, Steve Battle and Julian A. Padget. "Automatic Mapping of OWL Ontologies into Java". In: *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE 2004).* (Banff, Alberta). Edited by Frank Maurer and Günther Ruhe. 2004, pages 98–103.

[109] Sidney Katz, Amasa B. Ford, Roland W. Moskowitz, Beverly A. Jackson and Marjorie W. Jaffe. "Studies of illness in the aged: the index of ADL: a standardized measure of biological and psychosocial function". In: *Jama* 185.12 (1963), pages 914–919. DOI: 10.1001/jama.1963.03060120024016.

[110] John F. Kelley. “An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications”. In: *ACM Transactions on Information Systems* 2.1 (1984), pages 26–41. DOI: 10.1145/357417.357420.

[111] Dan J. Kim, John Hebeler, Victoria Yoon and Fred Davis. “Exploring Determinants of Semantic Web Technology Adoption from IT Professionals’ Perspective: Industry Competition, Organization Innovativeness, and Data Management Capability”. In: *Computers in Human Behavior* 86 (2018), pages 18–33. ISSN: 0747-5632. DOI: 10.1016/j.chb.2018.04.014.

[112] Hak Lae Kim, Simon Scerri, John G. Breslin, Stefan Decker and Hong Gee Kim. “The state of the art in tag ontologies: a semantic model for tagging and folksonomies”. In: *Proceedings of the International Conference on Dublin Core and Metadata Applications.* (Berlin, Germany). Edited by Jane Greenberg and Wolfgang Klas. 2008, pages 128–137.

[113] Michel Klein. “Combining and relating ontologies: an analysis of problems and solutions”. In: *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing.* (Seattle, USA). Edited by Asunción Gomez Perez, Michael Gruninger, Heiner Stuckenschmidt and Uschold Mike. 2001, pages 53–62.

[114] Marius Kloetzer and Calin Belta. “Temporal Logic Planning and Control of Robotic Swarms by Hierarchical Abstractions”. In: *IEEE Transactions on Robotics* 23.2 (2007), pages 320–330. DOI: 10.1109/TRO.2006.889492.

[115] Torben Knerr. *Tagging ontology-towards a common ontology for folksonomies.* 2006. URL: https://tagont.googlecode.com/files/TagOntPaper.pdf.

[116] Maddalen Lopez De Lacalle, Egoitz Laparra and German Rigau. “Predicate Matrix: Extending SemLink through WordNet mappings”. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014).* (Reykjavik, Iceland). Edited by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani,

Asunción Moreno, Jan Odijk and Stelios Piperidis. 2014, pages 903–909. ISBN: 978-2-9517408-8-4.

[117] Egoitz Laparra and German Rigau. "eXtended WordFrameNet". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010).* (Valletta, Malta). Edited by Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner and Daniel Tapias. European Language Resources Association, 2010. ISBN: 2-9517408-6-7.

[118] Craig Larman. *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development.* Prentice Hall, 2004.

[119] M. Powell Lawton and Elaine M. Brody. "Assessment of older people: self-maintaining and instrumental activities of daily living". In: *Gerontologist* 9.3 (1969), pages 179–186.

[120] Amanda Lazar, Hilaire Thompson and George Demiris. "A Systematic Review of the Use of Technology for Reminiscence Therapy". In: *Health Education & Behavior* 41.1 (2014), pages 51–61. DOI: 10.1177/1090198114537067.

[121] Olivier Lebeltel, Pierre Bessière, Julien Diard and Emmanuel Mazer. "Bayesian Robot Programming". In: *Autonomous Robots* 16.1 (2004), pages 49–79. DOI: 10.1023/B:AURO.0000008671.38949.43.

[122] Iolanda Leite, Carlos Martinho and Ana Paiva. "Social Robots for Long-Term Interaction: A Survey". In: *International Journal of Social Robotics* 5.2 (2013), pages 291–308. DOI: 10.1007/s12369-013-0178-y.

[123] Séverin Lemaignan, Raquel Ros, Lorenz Mösenlechner, Rachid Alami and Michael Beetz. "ORO, a knowledge management platform for cognitive architectures in robotics". In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2010).* (Taipei, Taiwan). IEEE, 2010, pages 3548–3553. ISBN: 978-1-4244-6674-0. DOI: 10.1109/IROS.2010.5649547.

[124] Angel Lagares Lemos, Florian Daniel and Boualem Benatallah. “Web Service Composition: A Survey of Techniques and Tools”. In: *ACM Computing Surveys* 48.3 (2015), 33:1–33:41. DOI: 10.1145/2831270.

[125] Douglas B. Lenat. “CYC: A Large-Scale Investment in Knowledge Infrastructure”. In: *Communications of the ACM* 38.11 (1995), pages 32–38. DOI: 10.1145/219717.219745.

[126] Juanzi Li, Jie Tang, Yi Li and Qiong Luo. “RiMOM: A Dynamic Multistrategy Ontology Alignment Framework”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.8 (2009), pages 1218–1232. DOI: 10.1109/TKDE.2008.202.

[127] Gi Hyun Lim, Il Hong Suh and Hyowon Suh. “Ontology-Based Unified Robot Knowledge for Service Robots in Indoor Environments”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41.3 (2011), pages 492–509. DOI: 10.1109/TSMCA.2010.2076404.

[128] Freddy Limpens, Alexandre Monnin, David Laniado and Fabien Gandon. “NiceTag Ontology: tags as named graphs”. In: *Proceeding of the 1st International Workshop on Social Networks Interoperability SNI*. 2009.

[129] Yen-Chun Lin, Yu-Tzu Dai and Shiow-Li Hwang. “The Effect of Reminiscence on the Elderly Population: A Systematic Review”. In: *Public Health Nursing* 20.4 (2003), pages 297–306. DOI: 10.1046/j.1525-1446.2003.20407.x.

[130] Hugo Liu and Push Singh. “ConceptNet –A Practical Commonsense Reasoning Tool-Kit”. In: *BT Technology Journal* 22.4 (2004), pages 211–226. ISSN: 1358-3948. DOI: 10.1023/B:BTTJ.0000047600.45421.6d.

[131] Giorgia Lodi, Luigi Asprino, Andrea Giovanni Nuzzolese, Valentina Presutti, Aldo Gangemi, Diego Reforgiato Recupero, Chiara Veninata and Annarita Orsini. “Semantic Web for Cultural Heritage Valorisation”. In: *Data Analytics in Digital Humanities*. Edited by Shalin Hai-Jew. Springer International Publishing, 2017, pages 3–37. DOI: 10.1007/978-3-319-54499-1_1.

[132] Rebecca G. Logsdon, Laura E. Gibbons, Susan M. McCurry and Linda Teri. "Assessing quality of life in older adults with cognitive impairment". In: *Psychosomatic medicine* 64.3 (2002), pages 510–519.

[133] Steffen Lohmann, Paloma Díaz and Ignacio Aedo. "MUTO: the modular unified tagging ontology". In: *Proceedings of the 7th International Conference on Semantic Systems (I-SEMANTICS 2011).* (Graz, Austria). Edited by Chiara Ghidini, Axel-Cyrille Ngonga Ngomo, Stefanie N. Lindstaedt and Tassilo Pellegrini. 2011, pages 95–104. DOI: 10.1145/2063518.2063531. URL: http://doi.acm.org/10.1145/2063518.2063531.

[134] Martin Lötzsch, Joscha Bach, Hans-Dieter Burkhard and Matthias Jüngel. "Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL". In: *RoboCup 2003: Robot Soccer World Cup VII.* Edited by Daniel Polani, Brett Browning, Andrea Bonarini and Kazuo Yoshida. Volume 3020. Springer, 2003, pages 114–124. ISBN: 3-540-22443-2. DOI: 10.1007/978-3-540-25940-4_10.

[135] Wing-Yue Geoffrey Louie, Tiago Stegun Vaquero, Goldie Nejat and J. Christopher Beck. "An autonomous assistive robot for planning, scheduling and facilitating multi-user activities". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014).* (Hong Kong, China). 2014, pages 5292–5298. DOI: 10.1109/ICRA.2014.6907637.

[136] Pattie Maes. "The Dynamics of Action Selection". In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI).* (Detroit, MI, USA). Edited by N. S. Sridharan. Morgan Kaufmann, 1989, pages 991–997. ISBN: 1-55860-094-9.

[137] Ester Martinez-Martin and Angel P. del Pobil. "Personal Robot Assistants for Elderly Care: An Overview". In: *Personal Assistants: Emerging Computational Technologies.* Edited by Angelo Costa, Vicente Julian and Paulo Novais. 2018, pages 77–91. DOI: 10.1007/978-3-319-62530-0_5.

[138] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino and Alessandro Oltramari. *WonderWeb Deliverable D18 - Ontology Library (Final Version)*. EU IST project deliverable. 2003.

[139] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly, 2011.

[140] Maja J. Mataric. "Behaviour-based control: examples from navigation, learning, and group behaviour". In: *Journal of Experimental & Theoretical Artificial Intelligence* 9.2-3 (1997), pages 323–336. DOI: 10.1080/095281397147149.

[141] Nikolaos Mavridis and Deb Roy. "Grounded Situation Models for Robots: Where words and percepts meet". In: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS 2006)*. IEEE, 2006, pages 4690–4697. ISBN: 1-4244-0258-1. DOI: 10.1109/IROS.2006.282258.

[142] John McCrae, Christiane Fellbaum and Philipp Cimiano. "Publishing and Linking WordNet using lemon and RDF". In: *Proceedings of the 3rd Workshop on Linked Data in Linguistics*. 2014.

[143] Ross Mead, Eric Wade, Pierre Johnson, Aaron St. Clair, Shuya Chen and Maja J. Mataric. "An architecture for rehabilitation task practice in socially assistive human-robot interaction". In: *Proceedings of 19th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2010)*. (Viareggio, Italy). Edited by Carlo Alberto Avizzano and Emanuele Ruffaldi. IEEE, 2010, pages 404–409. DOI: 10.1109/ROMAN.2010.5598666.

[144] Larry R. Medsker. "Design and development of hybrid neural network and expert systems". In: *Proceedings of IEEE International Conference on Neural Networks (ICNN - 1994)*. Volume 3. 1994, pages 1470–1474. DOI: 10.1109/ICNN.1994.374503.

[145] George A. Miller. "WordNet: A Lexical Database for English." In: *Communications of the ACM* 38.11 (1995), pages 39–41. ISSN: 0001-0782.

[146] George A. Miller and Florentina Hristea. "WordNet Nouns: Classes and Instances". In: *Computational Linguistics* 32.1 (2006), pages 1–3. ISSN: 0891-2017. DOI: 10.1162/coli.2006.32.1.1.

[147] Marvin Minsky. *A Framework for Representing Knowledge.* Technical report. Cambridge, MA, USA, 1974. URL: http://hdl.handle.net/1721.1/6089.

[148] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves and Joel Welling. "Never-Ending Learning". In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015).* (Austin, Texas, USA). Edited by Blai Bonet and Sven Koenig. AAAI Press, 2015, pages 2302–2310. ISBN: 978-1-57735-698-1.

[149] Elaine Mordoch, Angela Osterreicher, Lorna Guse, Kerstin Roger and Genevieve Thompson. "Use of social commitment robots in the care of elderly people with dementia: A literature review". In: *Maturitas* 74.1 (2013), pages 14–20. DOI: 10.1016/j.maturitas.2012.10.015.

[150] Roberto Navigli and Simone Paolo Ponzetto. "BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network". In: *Artificial Intelligence* 193 (2012), pages 217–250. DOI: 10.1016/j.artint.2012.07.001.

[151] Allen Newell. "The knowledge level". In: *Artificial intelligence* 18.1 (1982), pages 87–127.

[152] Nils J. Nilsson. *Shakey the robot.* Technical report. SRI International, 1984.

[153] Andrea Giovanni Nuzzolese, Aldo Gangemi and Valentina Presutti. "Gathering Lexical Linked Data and Knowledge Patterns from FrameNet". In:

Proceedings of the sixth international conference on Knowledge Capture (K-CAP). (Banff, AB, Canada). Edited by Mark A. Musen and Óscar Corcho. ACM, 2011, pages 41–48. ISBN: 978-1-4503-0396-5. DOI: 10.1145/1999676.1999685.

[154] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti and Paolo Ciancarini. "Semion: A Smart Triplification Tool". In: *Proceedings of the 17th International Conference on Knowledge Engineering and Management by the Masses (EKAW 2010) - Poster and Demo Track*. (Lisbon, Portugal). Edited by Johanna Völker and Óscar Corcho. 2010.

[155] Tetsuya Ogata and Shigeki Sugano. "Emotional communication robot: WAMOEBA-2R emotion model and evaluation experiments". In: *Proceedings of the International Conference on Humanoid Robots*. 2000.

[156] Alex Oliver and Timothy Smiley. "Multigrade Predicates". In: *Mind* 113.452 (2004), pages 609–681. DOI: 10.1093/mind/113.452.609.

[157] Eyal Oren, Benjamin Heitmann and Stefan Decker. "ActiveRDF: Embedding Semantic Web data into object-oriented languages". In: *Journal of Web Semantics* 6.3 (2008), pages 191–202. DOI: 10.1016/j.websem.2008.04.003.

[158] Jens Ortmann and Desiree Daniel. "An ontology design pattern for referential qualities". In: *Proceedings fo the 10th International Semantic Web Conference (ISWC 2011), Part I*. (Bonn, Germany). Edited by Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy and Eva Blomqvist. Volume 7031. Lecture Notes in Computer Science. Bonn, Germany: Springer, 2011, pages 537–552. DOI: 10.1007/978-3-642-25073-6_34.

[159] Jens Ortmann, Giorgio De Felice, Dong Wang and Desiree Daniel. "An egocentric semantic reference system for affordances". In: *Semantic Web* 5.6 (2014), pages 449–472.

[160] Paolo Pareti, Ewan Klein and Adam Barker. “Linking Data, Services and Human Know-How”. In: *Proceedings of the 13th European Semantic Web Conference (ESWC 2016).* (Heraklion, Crete, Greece). Edited by Harald Sack, Eva Blomqvist, Mathieu d’Aquin, Chiara Ghidini, Simone Paolo Ponzetto and Christoph Lange. Springer International Publishing, 2016, pages 505–520. ISBN: 978-3-319-34129-3. DOI: 10.1007/978-3-319-34129-3_31.

[161] Marius Pasca. “Finding Needles in an Encyclopedic Haystack: Detecting Classes Among Wikipedia Articles”. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW 2018).* (Lyon, France). Edited by Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas and Panagiotis G. Ipeirotis. ACM, 2018, pages 1267–1276. DOI: 10.1145/3178876.3186025.

[162] Heiko Paulheim and Aldo Gangemi. “Serving DBpedia with DOLCE – More than Just Adding a Cherry on Top”. In: *Proceedings of the 14th International Semantic Web Conference (ISWC 2015), Part I.* (Bethlehem, PA, USA). Edited by Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d’Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan and Steffen Staab. Springer International Publishing, 2015, pages 180–196. ISBN: 978-3-319-25007-6. DOI: 10.1007/978-3-319-25007-6_11.

[163] Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze and Benjamin Van Durme. “FrameNet+: Fast Paraphrastic Tripling of FrameNet”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, (ACL 2015) - Volume 2: Short Papers.* (Beijing, China). 2015, pages 408–413.

[164] R. S. Visser Pepijn, M. Jones Dean, T. J. M. Bench-capon and M. J. R. Shave. “An analysis of ontological mismatches: Heterogeneity versus interoperabil-

ity". In: *Proceedings of AAAI Spring Symposium on Ontological Engineering.* (Stanford, USA). 1997, pages 164–172.

[165] Silvio Peroni, Giorgia Lodi, Luigi Asprino, Aldo Gangemi and Valentina Presutti. "FOOD: FOod in Open Data". In: *Proceedings of the 15th International Semantic Web Conference (ISWC 2016), Part II.* (Kobe, Japan). Edited by Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck and Yolanda Gil. Springer International Publishing, 2016, pages 168–176. DOI: 10.1007/978-3-319-46547-0_18.

[166] Eric Pfeiffer. "A short portable mental status questionnaire for the assessment of organic brain deficit in elderly patients". In: *Journal of the American Geriatrics Society* 23.10 (1975), pages 433–441.

[167] Mohammad Taher Pilehvar, David Jurgens and Roberto Navigli. "Align, disambiguate and walk: A unified approach for measuring semantic similarity". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013).* (Sofia, Bulgaria). Edited by Massimo Poesio Hinrich Schuetze Pascale Fun. Association for Computational Linguistics, 2013, pages 1341–1351. ISBN: 978-1-937284-50-3.

[168] Alberto Pilotto, Daniele Sancarlo, Francesco Panza, Francesco Paris, Grazia D'Onofrio, Leandro Cascavilla, Filomena Addante, Davide Seripa, Vincenzo Solfrizzi, Bruno Dallapiccola, Marilisa Franceschi and Luigi Ferrucci. "The Multidimensional Prognostic Index (MPI), based on a comprehensive geriatric assessment predicts short- and long-term mortality in hospitalized older patients with dementia". In: *Journal of Alzheimer's disease* 18.1 (2009), pages 191–199. DOI: 10.3233/JAD-2009-1139.

[169] Steven Pinker and Jacques Mehler. *Connections and symbols.* Volume 28. Mit Press, 1988.

[170] Domenico M Pisanelli. *Ontologies in medicine.* Volume 102. IOS Press, 2004.

[171] Edson Prestes, Joel Luis Carbonera, Sandro Rama Fiorini, Vitor A. M. Jorge, Mara Abel, Raj Madhavan, Angela Locoro, Paulo J. S. Gonçalves, Marcos E. Barreto, Maki K. Habib, Abdelghani Chibani, Sébastien Gérard, Yacine Amirat and Craig Schlenoff. "Towards a core ontology for robotics and automation". In: *Robotics and Autonomous Systems* 61.11 (2013), pages 1193–1204. DOI: 10.1016/j.robot.2013.04.005.

[172] Valentina Presutti, Eva Blomqvist, Enrico Daga and Aldo Gangemi. "Pattern-Based Ontology Design". In: *Ontology Engineering in a Networked World.* Edited by Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta and Aldo Gangemi. Springer, 2012, pages 35–64. ISBN: 978-3-642-24793-4. DOI: 10.1007/978-3-642-24794-1_3.

[173] Valentina Presutti, Enrico Daga, Aldo Gangemi and Eva Blomqvist. "eXtreme Design with Content Ontology Design Patterns". In: *Proceedings of the Workshop on Ontology Patterns (WOP 2009).* (Washington, DC, USA). Edited by Eva Blomqvist, Kurt Sandkuhl, François Scharffe and Vojtech Svátek. Volume 516. CEUR Workshop Proceedings. CEUR-WS.org, 2009.

[174] Valentina Presutti, Giorgia Lodi, Andrea Nuzzolese, Aldo Gangemi, Silvio Peroni and Luigi Asprino. "The Role of Ontology Design Patterns in Linked Data Projects". In: *Proceedings of the 35th International Conference on Conceptual Modeling (ER 2016).* (Gifu, Japan). Edited by Isabelle Comyn-Wattiau, Katsumi Tanaka, Il-Yeol Song, Shuichiro Yamamoto and Motoshi Saeki. Springer International Publishing, 2016, pages 113–121. DOI: 10.1007/978-3-319-46397-1_9.

[175] James Pustejovsky. *The Generative Lexicon.* MIT Press, 1998.

[176] Pushpendre Rastogi and Benjamin Van Durme. "Augmenting FrameNet Via PPDB". In: *Proceedings of the 2nd Workshop on EVENTS: Definition, Detection, Coreference, and Representation (EVENTS@ACL 2014).* (Baltimore, Maryland). Edited by Teruko Mitamura, Eduard H. Hovy and Martha Palmer.

Association for Computational Linguistics, 2014, pages 1–5. ISBN: 978-1-941643-14-3. DOI: 10.3115/v1/W14-2901.

[177] David Riaño, Francis Real, Fabio Campana, Sara Ercolani and Roberta Annicchiarico. “An Ontology for the Care of the Elder at Home”. In: *Proceedings of the 12th Conference on Artificial Intelligence in Medicine AIME.* (Verona, Italy). Edited by Carlo Combi, Yuval Shahar and Ameen Abu-Hanna. 2009, pages 235–239.

[178] Laurel D. Riek. “Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines”. In: *Journal of Human-Robot Interaction* 1.1 (2012), pages 119–136. ISSN: 2163-0364. DOI: 10.5898/JHRI.1.1.Riek.

[179] Michael Ringgaard, Rahul Gupta and Fernando C. N. Pereira. “SLING: A framework for frame semantic parsing”. In: *CoRR* (2017). URL: http://arxiv.org/abs/1710.07032.

[180] Dominique Ritze, Christian Meilicke, Ondřej Šváb-Zamazal and Heiner Stuckenschmidt. “A pattern-based ontology matching approach for detecting complex correspondences”. In: *Proceedings of the 4th International Workshop on Ontology Matching (OM 2009) collocated with the 8th International Semantic Web Conference (ISWC 2009).* (Chantilly, USA). Edited by Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natasha Noy and Arnon Rosenthal. CEUR-WS.org, 2009, pages 25–36.

[181] Hayley Robinson, Bruce MacDonald and Elizabeth Broadbent. “The Role of Healthcare Robots for Older People at Home: A Review”. In: *International Journal of Social Robotics* 6.4 (2014), pages 575–591. DOI: 10.1007/s12369-014-0242-2.

[182] Sebastian Rockel, Bernd Neumann, Jianwei Zhang, Krishna Sandeep Reddy Dubba, Anthony G. Cohn, Stefan Konecny, Masoumeh Mansouri, Federico Pecora, Alessandro Saffiotti, Martin Günther, Sebastian Stock, Joachim Hertzberg, Ana Maria Tomé, Armando J. Pinho, Luís Seabra Lopes, Stephanie von Riegen and Lothar Hotz. “An Ontology-based Multi-level Robot Architecture for

Learning from Experiences". In: *Proceedings of the AAAI Spring Symposium.* (Palo Alto, California, USA). AAAI, 2013, pages 52–57.

[183] Silvia Rossi, Enrico Leone, Michelangelo Fiore, Alberto Finzi and Francesco Cutugno. "An extensible architecture for robust multimodal human-robot communication". In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2013).* (Tokyo, Japan). IEEE, 2013, pages 2208–2213. DOI: 10.1109/IROS.2013.6696665.

[184] Jacobo Rouces, Gerard de Melo and Katja Hose. "FrameBase: Representing N-ary Relations using Semantic Frames". In: *Proceedings of the 12th European Semantic Web Conference (ESWC 2015).* (Portoroz, Slovenia). Edited by Fabien Gandon, Marta Sabou, Harald Sack, Claudia d'Amato, Philippe Cudré-Mauroux and Antoine Zimmermann. Springer, 2015, pages 505–521. ISBN: 978-3-319-25638-2. DOI: 10.1007/978-3-319-18818-8_31.

[185] Laurence Z. Rubenstein, Andreas E. Stuck, Albert L. Siu and Darryl Wieland. "Impacts of geriatric evaluation and management programs on defined outcomes: overview of the evidence". In: *Journal of the American Geriatrics Society* 39.S1 (1991), pages 8–16. DOI: 10.1111/j.1532-5415.1991.tb05927.x.

[186] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra Kumar Misra and Hema Swetha Koppula. "RoboBrain: Large-Scale Knowledge Engine for Robots". In: *CoRRs* (2014). URL: http://arxiv.org/abs/1412.0691.

[187] Karin Kipper Schuler. "VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon". PhD thesis. University of Pennsylvania, 2006.

[188] John R Searle. "Social ontology: Some basic principles". In: *Anthropological theory* 6.1 (2006), pages 12–29.

[189] Jaydeep Sen, Ashish R. Mittal, Diptikalyan Saha and Karthik Sankaranarayanan. "Functional Partitioning of Ontologies for Natural Language Query Completion in Question Answering Systems". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 18).* Edited by Jérôme

Lang. 2018, pages 4331–4337. ISBN: 978-0-9992411-2-7. DOI: 10.24963/ijcai.2018/602.

[190] Robert Shaw, Michael T Turvey and William Mace. “Ecological psychology: The consequence of a commitment to realism”. In: *Cognition and the symbolic processes* 2 (1982), pages 159–226.

[191] Pavel Shvaiko and Jerome Euzenat. “Ontology Matching: State of the Art and Future Challenges”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.1 (2013), pages 158–176. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.253.

[192] Elena Paslaru Bontas Simperl, Malgorzata Mochol and Tobias Bürger. “Achieving Maturity: the State of Practice in Ontology Engineering in 2009”. In: *International Journal of Computer Science and Applications (IJCSA)* 7.1 (2010), pages 45–65. ISSN: 0972-9038.

[193] Push Singh. “The public acquisition of commonsense knowledge”. In: *Proceedings of AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access.* (Palo Alto, CA, USA). Edited by Jussi Karlgren. AAAI Press, 2002.

[194] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins and Wan Li Zhu. “Open Mind Common Sense: Knowledge Acquisition from the General Public”. In: *Proceedings of the DOA/CoopIS/ODBASE Confederated International Conferences.* (Irvine, California, USA). Edited by Robert Meersman and Zahir Tari. Springer, 2002, pages 1223–1237. ISBN: 3-540-00106-9. DOI: 10.1007/3-540-36124-3_77.

[195] Paul Smolensky. “Connectionist AI, symbolic AI, and the brain”. In: *Artificial Intelligence Review* 1.2 (1987), pages 95–109. ISSN: 1573-7462. DOI: 10.1007/BF00130011.

[196] Robert Speer, Joshua Chin and Catherine Havasi. “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”. In: *Proceedings of the 31st AAAI*

Conference on Artificial Intelligence and the Twenty-Ninth Innovative Applications of Artificial Intelligence Conference (AAAI 2017). (San Francisco, California USA). Edited by Satinder P. Singh and Shaul Markovitch. AAAI Press, Palo Alto, California, 2017, pages 4444–4451.

[197] Robert Speer and Catherine Havasi. “Representing General Relational Knowledge in ConceptNet 5”. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC.* (Istanbul, Turkey). Edited by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk and Stelios Piperidis. European Language Resources Association (ELRA), 2012, pages 3679–3686. ISBN: 978-2-9517408-7-7.

[198] Thomas A. Stoffregen. “Affordances as properties of the animal-environment system”. In: *Ecological Psychology* 15.2 (2003), pages 115–134.

[199] Anselm L. Strauss and Juliet M. Corbin. *Basics of Qualitative Research: Techniques and Procedures for developing Grounded Theory.* 1998. DOI: 10.4135/9781452230153.

[200] Ponnusamy Subramaniam and Bob Woods. “Towards the Therapeutic Use of Information and Communication Technology in Reminiscence Work for People with Dementia: a Systematic Review”. In: *International Journal of Computers Healthcare* 1.2 (2010), pages 106–125. DOI: 10.1504/IJCIH.2010.037457.

[201] Fabian M. Suchanek, Gjergji Kasneci and Gerhard Weikum. “Yago: A Core of Semantic Knowledge”. In: *Proceedings of the 16th International Conference on World Wide Web (WWW 2007).* (Banff, Alberta, Canada). Edited by Carey Williamson, Mary Ellen Zurko, Peter Patel-Schneider and Prashant Shenoy. ACM, 2007, pages 697–706. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242667.

[202] Vojtech Svátek, Martin Homola, Jan Kluka and Miroslav Vacura. “Metamodeling-Based Coherence Checking of OWL Vocabulary Background Models”. In:

Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013). (Montpellier, France). Edited by Mariano Rodriguez-Muro, Simon Jupp and Kavitha Srinivas. 2013.

[203] Swabha Swayamdipta, Sam Thomson, Chris Dyer and Noah A. Smith. "Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold". In: *CoRR* (2017). URL: http://arxiv.org/abs/1706.09528.

[204] Cui Tao, Wei-Qui Wei, Harold R. Solbrig, Guergana Savova and Christopher G. Chute. "CNTRO: A Semantic Web Ontology for Temporal Relation Inferencing in Clinical Narratives". In: *Proceedings of the Annual Symposium of American Medical Informatics Association (AMIA 2010).* (Washington DC, USA). 2010, pages 787–791.

[205] Moritz Tenorth and Michael Beetz. "KNOWROB - knowledge processing for autonomous personal robots". In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2009).* (St. Louis, Missouri, USA). IEEE, 2009, pages 4261–4266. ISBN: 978-1-4244-3803-7. DOI: 10.1109/IROS.2009.5354602.

[206] Moritz Tenorth and Michael Beetz. "KnowRob: A knowledge processing infrastructure for cognition-enabled robots". In: *The International Journal of Robotics Research* 32.5 (2013), pages 566–590. DOI: 10.1177/0278364913481635.

[207] Moritz Tenorth and Michael Beetz. "Representations for robot knowledge in the KnowRob framework". In: *Artificial Intelligence* 247 (2017), pages 151–169. DOI: 10.1016/j.artint.2015.05.010.

[208] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz and Michael Beetz. "Representation and Exchange of Knowledge About Actions, Objects, and Environments in the RoboEarth Framework". In: *IEEE Transactions on Automation Science and Engineering* 10.3 (2013), pages 643–651. DOI: 10.1109/TASE.2013.2244883.

[209] Michael Thielscher. “Representing the Knowledge of a Robot”. In: *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR 2000).* (Breckenridge, Colorado, USA). Edited by Anthony G. Cohn, Fausto Giunchiglia and Bart Selman. Morgan Kaufmann, 2000, pages 109–120.

[210] Sally Thorne, Sheryl Reimer Kirkham and Katherine O’Flynn-Magee. “The Analytic Challenge in Interpretive Description”. In: *International Journal of Qualitative Methods* 3.1 (2004), pages 1–11. DOI: 10.1177/160940690400300101.

[211] Elena Torta, Franz Werner, David O. Johnson, James F. Juola, Raymond H. Cuijpers, Marco Bazzani, Johannes Oberzaucher, John Lemberger, Hadas Lewy and Joseph Bregman. “Evaluation of a Small Socially-Assistive Humanoid Robot in Intelligent Homes for the Care of the Elderly”. In: *Journal of Intelligent and Robotic Systems* 76.1 (2014), pages 57–71. DOI: 10.1007/s10846-013-0019-0.

[212] Bruno Vellas, Yves Guigoz, Philip J. Garry, Fati Nourhashemi, David Bennahum, Sylvie Lauque and Jean-Louis Albarede. “The Mini Nutritional Assessment (MNA) and its use in grading the nutritional state of elderly patients”. In: *Nutrition* 15.2 (1999), pages 116–122.

[213] Bob Woods, Laura O’Philbin, Emma M. Farrell, Aimee E. Spector and Martin Orrell. “Reminiscence therapy for dementia”. In: *Cochrane Database of Systematic Reviews* 3 (2009), pages 1–36. DOI: 10.1002/14651858.CD001120.pub3.

[214] Lu Zhou, Michelle Cheatham, Adila Krisnadhi and Pascal Hitzler. “A Complex Alignment Benchmark: GeoLink Dataset”. In: *Proceedings of the 17th International Semantic Web Conference (ISWC 2018) - Part II.* (Monterey, CA, USA). Edited by Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee and Elena Simperl. Volume 11137. Lecture Notes in Computer Science.

Springer, 2018, pages 273–288. ISBN: 978-3-030-00667-9. DOI: 10.1007/978-3-030-00668-6_17.

[215] Gregory D. Zimet, Nancy W. Dahlem, Sara G. Zimet and Gordon K. Farley. "The multidimensional scale of perceived social support". In: *Journal of personality assessment* 52.1 (1988), pages 30–41.

[216] Cäcilia Zirn, Vivi Nastase and Michael Strube. "Distinguishing between Instances and Classes in the Wikipedia Taxonomy". In: *Proceedings of the 5th European Semantic Web Conference (ESWC 2008).* (Tenerife, Canary Islands, Spain). Edited by Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann and Manolis Koubarakis. Springer, 2008, pages 376–387. ISBN: 978-3-540-68233-2. DOI: 10.1007/978-3-540-68234-9_29.