

Alma Mater Studiorum Università di Bologna Archivio istituzionale della ricerca

A reference architecture for social robots

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version: A reference architecture for social robots / Luigi Asprino, Paolo Ciancarini, Andrea Giovanni Nuzzolese, Valentina Presutti, Alessandro Russo. - In: JOURNAL OF WEB SEMANTICS. - ISSN 1570-8268. -ELETTRONICO. - 72:(2022), pp. 1-18. [10.1016/j.websem.2021.100683]

Availability: This version is available at: https://hdl.handle.net/11585/859663 since: 2022-02-16

Published:

DOI: http://doi.org/10.1016/j.websem.2021.100683

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/). When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

Asprino, L., Ciancarini, P., Nuzzolese, A. G., Presutti, V., & Russo, A. (2022). A reference architecture for social robots. Journal of Web Semantics, 72, 100683.

The final published version is available online at:

https://doi.org/10.1016/j.websem.2021.100683

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<u>https://cris.unibo.it/</u>)

When citing, please refer to the published version.

A Reference Architecture for Social Robots

Luigi Asprino^{a,*}, Paolo Ciancarini^{a,b}, Andrea Giovanni Nuzzolese^c, Valentina Presutti^{a,c}, Alessandro Russo^c

^a University of Bologna, via Zamboni, 32 - 40126 - Bologna ^b Innopolis University, Russia ^cSTLab (ISTC-CNR), Via San Martino della Battaglia, 44, 00185 Rome, Italy

Abstract

Social robotics poses tough challenges to software designers who are required to take care of difficult architectural drivers like acceptability, trust of robots as well as to guarantee that robots establish a personalized interaction with their users. Moreover, in this context recurrent software design issues such as ensuring interoperability, improving reusability and customizability of software components also arise. Designing and implementing social robotic software architectures is a time-intensive activity requiring multi-disciplinary expertise: this makes it difficult to rapidly develop, customize, and personalize robotic solutions. These challenges may be mitigated at design time by choosing certain architectural styles, implementing specific architectural patterns and using particular technologies. Leveraging on our experience in the MARIO project, in this paper we propose a series of principles that social robots may benefit from. These principles lay also the foundations for the design of a reference software architecture for social robots. The goal of this work is twofold: (i) Establishing a reference architecture whose components are unambiguously characterized by an ontology thus allowing to easily reuse them in order to implement and personalize social robots; (ii) Introducing a series of standardized software components for social robots architecture (mostly relying on ontologies and semantic technolo-

^{*}Corresponding author

Email addresses: luigi.asprino@unibo.it (Luigi Asprino),

paolo.ciancarini@unibo.it (Paolo Ciancarini), andreagiovanni.nuzzolese@cnr.it (Andrea Giovanni Nuzzolese), valentina.presutti@unibo.it (Valentina Presutti), alessandro.russo@istc.cnr.it (Alessandro Russo)

gies) to enhance interoperability, to improve explainability, and to favor rapid prototyping.

Keywords: Social Robots Design, Software Architectures, Architectural Patterns, Interoperability, Ontologies, Linked Open Data

1. Software Design in Social Robotics

Social robots¹ 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. Social robots provide valuable solutions for domains, such as education [1] or healthcare [2], where robots must have social skills to establish and preserve social relationships (even if their domain is dominated by non-social activities) [3]. In order to make robots able to establish social relationships, they must be designed so to favor *acceptability*, *trust* and to guarantee a *personalized interaction*¹ with their users. Moreover,

there exists a variety of challenges that arises in the implementation of a (social) robotics solution [4]: (i) How to guarantee the syntactic and semantic interoperability of data exchanged by software artifacts running on a robotic platform? (ii) How to integrate and ease the deployment of software modules in robotic architecture? (iii) How to ease the customization of robot's behavior? (iv) How to enhance the reusability of software components in robot's architectures?

Physical body, hardware and software components contribute to the development of robot social skills and have a role in mitigating social robotics challenges, and establishing and favoring the human-robot interaction [5, 6]. In this paper, we focus on robot's software layer by investigating how to organize software components in order to facilitate the development of robot's social skills as well as to enable robots to carry out a personalized interaction with their users and to increase acceptability, trust, and sociability. We discuss how the design of

¹The definitions of the terms "Social robot", "Acceptability", "Trust" and "Personalized interaction" are reported in Section 1.1.

the software architecture of the social robots may benefit of well-known architectural patterns and established technologies. We claim that the design of the

- robot's software architecture mitigates common challenges for social robotics at design time. Moreover, we propose a reference software architecture for social robots that, in our opinion, may favor the reuse of established technologies and standardized software components thus streamlining the development of social robotics solutions. This architecture: (i) defines a common vocabulary (i.e.
- an ontology whose terms unambigously characterize software components) that enables social robot designers to formally specify their software architectures; *(ii)* introduces a series of standardized software components for social robots architecture relying on ontologies and semantic technologies; *(iii)* serves as a template for developing systems; *(iv)* provides a generalization of 29 existing
- ³⁵ robotics solutions. Finally, the architecture is aimed at: (i) easing customizability and extensibility of robot's behavior and social skills; (ii) guaranteeing explainability and predicatability of robot's decisions; (iii) improving both inner (among architectural components) and outer (with external entities) interoperability; (iv) enabling a rapid prototyping of robotic applications; (v) enhancing reusability of architectural components.

Design Methodology. The proposed architecture has been developed with a bottom-up approach. We have elicited a set of architectural drivers (cf. Section 3) from the use cases [7] defined in the context of a Socially Assistive Robotics project (cf. Section 6) [8]. This generalization was aimed at formu-

- ⁴⁵ lating the architectural drivers in a way to capture both the Social Drivers discussed before and the major challenges for software architecture design in social robotics. The drivers led us to the selection of a set of architectural principles, namely, architectural styles and patterns social robotics architectures may benefit from. Furthermore, in order to generalize the design of the architecture we
- ⁵⁰ have analyzed 29 social robotics solutions in order to characterize a reference architecture for a social robot. These architectures were also evaluated with respect to the elicited architectural drivers. Drivers and principles have guided

the design of the architecture and the ontology presented in Section 5. As a result, the architecture proposed in Section 5 is an instance of the reference

architecture of a social robot discussed in Section 2 that also follows the architectural drivers elicited in Section 3 and the architectural principles of Section 4. Finally, we performed a scenario-based evaluation of the reference architecture with respect to scenarios elicited in the context of the MARIO project [8].

Paper Outline. The rest of the paper is organized as follows. Sections 1.1 and 1.2

- concludes the introduction to the paper by presenting some reference definitions and the case study (i.e. the MARIO project). An analysis of the existing software architectures for social robots is provided in Section 2. Section 3 discusses the drivers that guided the design of the architecture. A set of architectural and technological principles meeting the aforementioned drivers are presented
- ⁶⁵ in Section 4. Section 5 describes the proposed architecture and Section 6 gives an insight of how the architecture has been implemented in MARIO. Section 7 reports on the scenario-based evaluation of the reference architecture and the assessment of the surveyed architectures with respect to the elicited architectural drivers. Finally, Section 8 concludes the paper and outlines the future work.

1.1. Definitions

Social robot. Several definitions [9, 10, 11, 12, 13, 14] 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

- ⁷⁵ 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 [15, 16]).
- Social Drivers. Acceptability is described as "the demonstrable willingness within a user group to employ technology for the tasks it is designed to support" [17].

Trust is defined as "a belief, held by the trustor, that the trustee will act in a manner that mitigates the trustor's risk in a situation in which the trustor has put its outcomes at risk" [18]. Humans assess the reliability of a relation when

- interacting with another agent, hence it becomes critical for robots to act in a way to create and maintain a trusted relation. No matter how capable is an embodied agent, since its actions may entail risk for its users, they do not interact with it if they do not trust it. To be accepted in our society robots must show that they are worthy of trust [19]. A few studies [20, 21] have demonstrated that
- the ability of robots of personalizing the interaction with their users is one of the key features that reinforces people's rapport, cooperation, and engagement with a robot. Robots able to personalize the social interactions adapt their behavior and interaction policy in order to accommodate user preferences, needs, and emotional and psychological state.
- It is worth noticing that the elicited drivers are compliant with the European Ethics guidelines for trustworthy Artificial Intelligence² issued in 2019 and the Artificial Intelligence Act (AIA) proposed on April 2021³ which, among the other principles, claim that the trasparency, the technical robustness and the human agency are key requirements for an AI system to be deemed trustworthy.
- ¹⁰⁰ Since the architectural and technological choices discussed in this paper can be applied to any AI system, the benefits of this work are not limited to the social robotics but apply, with appropriate adaptions, to any AI system. However, the generalization of the results of this work to any AI system is out of the scope of this article and is left to future research.

105 1.2. Case Study

A case study for this work has been provided by the H2020 European Project MARIO⁴ [8]. This project has investigated the use of autonomous compan-

3A52021PC0206

²https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai ³https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1623335154975&uri=CELEX%

⁴MARIO project, http://www.mario-project.eu/portal/

ion robots as cognitive stimulation tool for persons with dementia (PWDs). The MARIO robot and its abilities were specifically designed to provide sup-

- port to PWDs, their caregivers, and related healthcare professionals. Among its abilities, MARIO can help caregivers in the patient assessment process by autonomously performing Comprehensive Geriatric Assessment (CGA) evaluations, it is able to deliver reminiscence therapy through personalized interactive sessions and to entertain its users by playing music or making them reading newspapers or playing videogames. The overall framework has been deployed
- on Kompaï-2 robots (showed in Figure 7), 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).

¹²⁰ 2. Analysis of the Existing Software Architectures for Social Robots

Despite the different application domains and the intended functions, software architectures of Social Robots have a common underlying structure. This common underlying structure has been synthesized in the Reference Architecture showed in Figure 1. This architecture is the result of the analysis of 29 papers describing social robotics systems [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50] and it also leverages on our experience in the MARIO project [8].

These works have been selected with Systematic Mapping Study (SMS) method [51] that comprised three steps:

(i) Planning a study. The study has the objective of giving an overview of software architectures of social robots by highlighting principles, requirements and design choices on which they are based on. Therefore, in order to be selected in the study, an article must meet the following criteria, it must: (a) present a social robot; (b) provide all the details of the software architecture of the robot.



Figure 1: Reference Architecture of a Social Robot.

(ii) Data collection. The collection of the articles leveraged a previous analysis [52] which identified 10 papers meeting the selection criteria. We enriched this collection by carrying out a systematic analysis of all the issues of the International Journal of Social Robotics (IJSR). We analyzed IJSR papers in order to select all the papers presenting social robotics solutions. This selection, which resulted in 25 papers, considered papers' title and abstract only. A further deeper analysis of these papers was needed in order to keep only the papers that discuss the architectural design of the robot. This step restricted the set of papers of IJSR papers to 12. Additional 4 papers were identified by analyzing the references of the selected papers. Finally, we ran a search on Web of Science database with keywords "architecture" AND "social" AND "robot". We ordered 543 results by relevance and we analyzed the first 100 results. Since most of the relevant articles were already selected by the above criteria or didn't provide a clear description of the software architecture, only 3 papers were included in the study.

145

140

- (iii) Analyzing the results. A qualitative analysis of the selected 29 papers is presented in this section. Moreover, An assessment of these architectures with respect to the architectural drivers is discussed in Section 7.
- All of the existing architectures define a layering of their components. Almost all of them differentiate the robot's "deliberative" from the "automatic" level (terminology borrowed from [53] but also adopted in [34] which refers to a *behavioral* layer rather than *automatic*). The former is meant to decide the next actions the robot has to perform. The latter allows the robot to perceive the environment through its sensors (such as lasers, cameras, touch sensors, microphones etc.), to manage the actuators (e.g. wheels engines, speakers etc.) and to provide other basic facilities such as: speech to text (S2T), text to speech (T2S), motion controller and HCI (the human control interface, i.e. a software component that provides a set of APIs that are meant to manage embodied devices, such as a tablet or buttons, that can be used by users to command the

robot). Software components running on the automatic level are often provided by the standard development kit of robots.

An alternative terminology, proposed by Wood et al. [49], separates sensors, which constitute the "sense" layer, from actuators, which form the "act" layer, and the deliberative layer is called "think" layer.

Although most of the systems run entirely on robot, there also exist examples of distributed robotic architectures, such as [23, 39, 40, 29], in which the deliberative layer is hosted on a remote server and the automatic runs on the robot. In such architectures, the deliberative layer can also control multiple robots at the same time. Figure 1 indicates the group of components running on the robot as "Agent" and the components that can run remotely as "Server".

Deliberative and automatic layers usually communicate through a semantic bus. A semantic bus manager is a software component implementing the publish/subscribe pattern for supporting a loosely coupled communication through-

¹⁸⁰ out the system (e.g. topics in ROS [54]). The semantic bus manager allows

175

software component to establish semantic buses that are N-to-N communica-

tion channels in which strictly-typed messages flow from the publishers to the subscribers of the channel.

Peculiarities of individual systems apart, all the architectures broadly converge on a deliberative layer constituted by a *Behavior Manager*, a *Knowledge Base Manager*, a set of predefined *Behaviors*, a set of *Capabilities* and, if necessary, a *Supervisory Controller*. The Behavior Manager is a software component that gathers information from perceptual components and knowledge base in order to decide the next actions the robot has to perform. Specifically, it de-

tects from the acquired information, the current state of the world and then it activates one of its predefined Behaviors. A Behavior is a procedure that makes the robot perform actions (e.g. movements, reproducing sounds etc.). Examples of Behaviors are entertaining the user, reminding something, charging its battery etc.

Some existing architectures [26, 35, 43, 22] distinguish general purpose capabilities of robots from their behaviors. In such architectures, capabilities used by multiple behaviors (such as face detection, person tracking, dialogue managing, etc.) form a layer apart that enables the reuse of capabilities among Behaviors. These capabilities complement basic capabilities provided by the automatic level, such as S2T, T2S, HCI, etc.

A Knowledge Manager is a software component that provides APIs to store information for supporting the robot's behaviors, tracing the users' activities or preferences, and collecting from the operating environment (e.g. maps). Many existing architectures [25, 26, 28, 31, 32, 39, 40, 48, 43] have a centralized Knowledge Base (sometimes articulated into two databases, such as [40]) all the components can contribute to/benefit from.

205

Some architectures, such as [25, 26, 31], explicitly define a Supervisory Controller that enables to remotely govern the robot. Such an interface is part of one of the most common HRI experimental techniques called Wizard of Oz [15, 16].

²¹⁰ In this setting, subjects interacting with the robot believe that it is autonomous but it is actually being operated by an unseen human being.

3. Elicited Architectural Drivers

In this section, we introduce the main drivers that lead the design of the proposed architecture. The formulation of these drivers aimed at capturing: (i) the use cases [7] defined in the context of the MARIO project [8]; (ii) the Social Drivers (discussed in Section 1); (iii) the major challenges for software architecture design in social robotics. We classified the drivers into functional (Section 3.1) and non-functional (Section 3.2) drivers.

3.1. Functional Drivers

Functional drivers are defined as the "functionalities" that a system must implement in order to satisfy a requirement. Therefore, functional drivers may vary a lot depending on the objectives of the specific system. In this section, we discuss the general functional drivers that a social robot should meet. It is worth noticing that the functionalities described in this section might require both capabilities and behaviors to be implemented.

Perceiving/Interacting/Moving within the Environment (Perceiving). A social robot should be able to perceive, to move itself within and physically interact with its operating environment [55]. These drivers 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 may overlook these drivers if the interaction with humans is limited to non-physical modalities (e.g. spoken language) and if it doesn't need to perceive the external environment. In such cases, the robots still have a physical body but they don't use it for interacting with humans or the environment. Examples of this kind of robots are the personal assistants (e.g. Amazon Echo, Google
 - Home etc.).

Interacting with Humans (Interacting). Interaction, between robots and humans may take several forms depending on human-robot proximity (cf. [6]). For a social interaction it is important that humans and robots are co-located in the

- ²⁴⁰ 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 stress on the most important abilities that enable interaction with humans and increase the social acceptability of robots. *Dialoguing* is a form of interaction where two or
- ²⁴⁵ 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 involves capabilities related to speech and natural language processing such as: (i) Speech recognition, i.e. the ability of recognizing and translating spoken lan-
- ²⁵⁰ guage 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
- ²⁵⁵ 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 [56]. For example, facial gestures [57] and motion behaviors [58] may facilitate to develop a social relationship with a robot.

Learning and Memorizing Knowledge (Learning). In order to increase its social acceptability and to evolve its social skills, a social robot must be able to learn (e.g. facts, rules, norms etc.) [59]. 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.

3.2. Non-Functional Drivers

Non-Functional Drivers specify general properties of a system, rather than its
 specific functional behavior. This section summarizes the general non-functional requirements that a social robot should meet.

Flexibility/Modifiability/Extensibility (Extensibility). One of the major challenges in robotics concerns the design of software architectures to support the development of the robot behaviors as plug-and-play applications [60]. The

- 275 robot software architecture should offer an extensibility mechanism to support the composition of new robot behaviors by combining the existing ones 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 the robot's hardware) to the component aimed at understanding what the user says (i.e. components that perform high-level tasks). Moreover, extensibility of robot's behavior makes sure that the robot is able to easily extend its capabilities to meet users' requirements, thus increasing its accept-ability.

Customizability (Customizability). Customizing robot's behavior and social skills on the basis of users' needs is crucial in order to improve robot acceptability [61, 62]. Therefore, robotic software applications should either provide an easy customization interface available for behavior's designers or guarantee that the robot is able to autonomously learn how to modify its behavior to meet the

user needs.

290

295

Predictability/Explainability (Explainability). To instill trust to its users, a robot must be able to explain and justify its actions, and its behavior should be predictable. Without a satisfactory explanation for robots' actions, users are not able to assess robots' decisions thus reducing its trustworthiness [63].

Interoperability (Interoperability). Generally speaking, interoperability is the ability of a system to interact and work together with other systems [64]. In order to interoperate, the systems have to agree on a common data format, an unambiguous meaning for the exchanged data, and a protocol for exchange-

- ing information. In robotics we can distinguish two kinds of interoperability: inner and outer. The inner interoperability is defined as the ability of a software component to interoperate with another component running on the same robot. The outer interoperability is the ability of a robot (i.e. one of its software component) to intoperate with another robot [65]. Interoperability is not directly related to social robotics challenges but it is a transversal requirement
- that enables software components to be easily integrated, reused and deployed.

Rapid Prototyping and Reusability of Software Components (Reusability). One of the major problems for the design of a social robot is the definition of the user requirements, which, in turn, is critical to the adoption of the robot

- itself. An accurate early elicitation of the requirements and early feedback on the robotic solution may save from costly changes at a later stage. Similar problems are experienced by software engineers which usually mitigate these issues with rapid prototyping. Rapid prototyping refers to a class of techniques that are meant to create and test system prototypes at a very early stage with the aim of
- ³¹⁵ getting feedback from users as early as possible. Rapid prototyping techniques have long been adopted in robotics too (e.g. [66, 67]) and, since user experience is even more critical than in other domains, we believe that rapid prototyping techniques should be extensively applied in social robotics. A common strategy to enable rapid prototyping is to compose systems by reusing off-the-shelf soft-
- ware components available on a trusted repository (e.g. apt or yum for Linux). Therefore, we advocate for the establishment of a repository of software components that can be easily used to quickly build robotic solutions, thus letting robot designers to focus only on personalising robot's behavior. Finally, rapid prototyping is not only a matter of software but also of background knowledge.
- $_{325}$ In fact, social robots need a wide range of heterogeneous background knowledge

for their tasks which should be available for robots in the same way as software components are available.

4. A Selection of Architectural and Technological Choices

This section provides an overview of the principles that led to the design of the architecture. Firstly, we choose a robotic approach, namely *behavioral robotics* [68], which we consider the most appropriate to fulfill the requirements elicited in Section 3. Then, we select the architectural style, i.e. service-oriented architecture, which best fits with this robotic approach. Finally, we develop the architecture by applying a set of architectural patterns (i.e. reflection, hot deployment and black board) and by selecting the most suitable technologies with respect to the requirements (semantic web, linked open data). The remaining of this section briefly presents these principles and the reasons that motivate

such choices.

Behavioral robotics. Behavioral robotics [68] is a robotic approach in which ³⁴⁰ robot's behavior is built in a bottom-up fashion, starting from simple behaviors which are the basic building blocks to realize robot's behavior. Robot behaviors can run in general simultaneously and are situated (i.e. do not need complex abstract world models to operate). Behavior-based robotics is mostly oriented to reactive behaviors in which there is a direct coupling between sensors and actu-

- ator. The architecture presented in this article generalizes the classic behavioral robotics approach by combining output of sensors with (symbolic) knowledge of the robot. Sensors' data and symbolic knowledge are combined in symbolic rules that activate robot's behaviors (not directly actuators). This strategy increases the customizability and personizability since the robot's knowledge base
- can store users' preferences or habits that can be used to personalize robot's behavior. Moreover, since robots' behaviors are activated by symbolic rules, they provide an explanation of the behavior thus favoring the predictability. Finally, this approach enhances rapid prototyping and reusability since behaviors, implemented as modular software components, can be (re)used as building

355 blocks.

Service-Oriented Computing. Service-Oriented Computing (SOC) [69] is a computing paradigm in which services are the fundamental elements for developing applications. Services are applications that perform certain functions. A service is (i) invocable through a platform (e.g. the Web); (ii) self-contained;
(iii) technology neutral; (iv) and, its behavior is described by a formal specification. One of the most common technologies for implementing SOC paradigm is REST. REST is a software architectural style used for creating Web applications. REST services provide its resources in a textual form and allow clients applications to read and modify them with a stateless protocol (which relies

- on HTTP). Services can be easily composed in order to realize composite functions. These features make service-oriented architectures an ideal candidate to implement an architecture inspired to behavioral robotics principles. In this solution, robots provide a platform able to host pluggable applications that either realize a robotic behavior or offer some functionality to other applications.
- Different benefits might be achieved through the adoption of service-oriented principles for designing robotic software architectures: (i) SOC's neutrality with respect to a specific technology guarantees interoperability of software components. (ii) Compositionality of services ensures the rapid development of robot's behaviors.
- Hot Deployment. Hot-deployable services [70] are services that can be added to or removed on-the-fly from a running server. This mechanism allows to change what is currently deployed on the platform without redeploying the whole platform. In particular, hot deployment may enable software architectures of social robots to be dynamically extended with new components or to easily customize the deployed ones.

Blackboard Systems. Blackboard systems [71] are systems constituted by independent components that cooperate to solve a problem using a blackboard (i.e. a shared knowledge base) as workplace for developing the solution. In these systems each component is specialized at solving a certain task of the overall

- problem (this realizes what is called modularization of expertise). Components are activated either when a change in the blackboard occurs (e.g. addition or removal of information) or when they receive an external events. Components are at the same time contributors and beneficiary of the blackboard, namely, their behavior is influenced by the status of blackboard and they record infor-
- ³⁹⁰ mation on the blackboard. Software architectures of social robots may benefit of a blackboard-system-like design for several reasons. With such an architecture software components benefit from and contribute to robot's knowledge hence making robot's behavior conditioned by its knowledge. Since changing the robot's knowledge would affect its behavior, this kind of architecture would
- ³⁹⁵ increase the robot's customizability, personalisability, predictability and adaptability over time with respect to the user's habits and preferences. This solution also pushes software components to adopt a shared data model, thus favoring the inner syntactic and semantic interoperability at data level.
- Semantic Web, Linked Open Data and Ontologies. The Semantic Web [72] is
 an extension of the Web which is aimed at providing a set of standards that allows data to be shared and reused across application boundaries. The Semantic Web standards mainly include: XML, RDF, RDFs, OWL, and SPARQL. XML which is a uniform data-exchange format thus providing a common syntax for exchange data. RDF⁵ is a framework for modeling information in form
 of triples (i.e. subject, predicate, object). RDFs⁶ provides a data-modeling vocabulary for RDF data. OWL⁷ 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. SPARQL⁸ defines a query

⁵RDF, W3C Recommendation https://www.w3.org/TR/rdf11-concepts/

⁶RDFs, W3C Recommendation https://www.w3.org/TR/rdf-schema/

⁷OWL, W3C Recommendation https://www.w3.org/TR/owl-ref/

⁸SPARQL, W3C Recommendation https://www.w3.org/TR/rdf-sparql-query/

- ⁴¹⁰ language and a protocol for retrieving and manipulating data stored in RDF format. Linked Data is structured data that is shared across the internet using Semantic Web standards. Linked data distributed with an open license are called Linked Open Data (LOD). Ontologies are explicit conceptualizations of a domain of interest that can be specified in Semantic Web format. Robots' archi-
- ⁴¹⁵ tectures can profoundly benefit of the Semantic Web technologies, the Linked Open Data paradigm and the adoption of ontologies: (i) The semantic interoperability among software components might be guaranteed from the adoption of ontologies which provide a shared conceptualization of the domain on which the component operate on. (ii) Since such technologies rely on symbolic ap-
- ⁴²⁰ proaches to Artificial Intelligence, they enable the development of predictable decision making mechanisms (subsystems that use the robot's knowledge and a set of rules defined on top it to decide which action to undertake). *(iii)* Linked Data provides a mechanism that allows robots to mutually share knowledge, thus increasing outer interoperability. *(iv)* Linked Open Data paradigm lets
- ⁴²⁵ to easily reuse existing external datasets so to bootstrap knowledge base with relevant information for robots' activities, and consequently enabling a rapid prototyping of robotic applications.

5. Proposed Architecture Design

This section presents the the design of the reference architecture for social robots. We first introduce the architecture by presenting its layers (Section 5.1), then we describe the components of each layer (Section 5.2). Finally, Section 5.3 introduces an ontology for specifying social robots architectures.

5.1. Architectural Layering

In order to enhance the separation of concerns among components, the soft-⁴³⁵ ware architecture has been organized into four concentric layers. Figure 2 depicts the architecture layering. These layers are complemented by a cross-layers group of components that provide architecture with hot-deploy support. Software components on a layer use components of its nested layers (therefore, the



Figure 2: The layered structure of the software architecture. It is worth noticing that the layering organization of the architecture reflects the structure of the architecture presented in Figure 1. The mapping between the two architectures is emphasized by layers' colors.

nesting structure depends on the usage relations among the software compo-

- ⁴⁴⁰ nents not the relation of the components with the users or the environment). Every layer can exist without the layers above it, and requires the layers inside it to function. Differently, the layer supporting the hot-deploy mechanism is used by its adjacent layers. This section provide an overview of the different concerns of the levels.
- The base layer is made up of the robot's knowledge and the bus subsystems which constitute the blackboard of the system. Software components may use the knowledge base to store persistent information and the semantic bus to asynchronously exchange messages or to generate and react to events. Each of these subsystem is split into two sub-levels, one providing the resources to
- ⁴⁵⁰ be accessed (i.e. the knowledge base and the semantic bus) and the other providing the APIs to access them. The access to blackboard level is guaranteed by the object-ontology mapping API, which provides software components with facilities for accessing the knowledge base, and the semantic bus API that allows components to create, subscribe to, or publish messages on a message queue (or to the base).
- 455 topic).

Looking at the higher levels, software components are classified into behaviors and capabilities. We define capabilities as software components that give to robots human-like abilities such as: listening, speaking, understanding natural language, moving etc. These capabilities are typically domain independent and

- therefore such components can be re-used in different robotic applications. Similarly to [13], capabilities are classified into basic and convoluted capabilities. Basic capabilities deliver primitive robot functionalities (e.g. reproducing or capturing sounds, speech to text, text to speech etc.). Convoluted capabilities are higher level functionalities (e.g. making phone calls) built on top of the basic
- ones. From a developer point of view, both classes of capabilities correspond to functionalities provided by the robot platform. The difference between the two classes is subtle: (i) Convoluted capabilities typically depend on basic ones (e.g. natural language understanding relies on speech to text); (ii) Convoluted capabilities are services that can be included in the robotic platform at run-
- time using a hot-deploy mechanism (e.g. if the robot has a camera, an object recognition tool can be installed at runtime); *(iii)* Basic capabilities are tightly related to hardware devices (e.g. text to speech component uses robot's speaker to reproduce text) and therefore can only be included in the architecture at design time.
- Software components that belong to the behavior level are meant to implement the high level behavior of the robot. The robot's behaviors are defined as sequences 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 such as: "approaching" and "conversing" with the user, "showing" videos, "reproducing" music and so on. Actions require some robot capabilities like: "moving", "speaking", "listening", "understanding", "showing images" and so on.

Except for the base layer and the basic capabilities, each level has an hotdeploy manager that allows to deploy new components. This means that the ⁴⁸⁵ architecture enables (at run-time) to deploy new robots behaviors, new robotcapabilities (that do not require new hardware components) and to extend the



Figure 3: The UML component diagram of the reference software architecture for social robots. The components are colored according to their layer. Specifically, C:Behavior and C:BehaviorScheduler belong to Behaviors level (purple-colored); C:Text2Speech, C:HCIMa nager, C:Speech2Text, and C:PerceptionAndMotionController are part of the Capabilities level (red-colored); C:ObjectOntologyMappingModule and C:KnowledgeBase are two sub-levels of the knowledge level (yellow-colored) which together with the C:SemanticBus (light-blue-colored) constitute the base layer; the software components providing hot-deploy support are green-colored.

structure of the knowledge base by instantiating new ontology-object mappers. This feature allows to incrementally develop the robot's architecture.

5.2. Architectural Components

The Figure 3 shows the UML component diagram of the reference architecture. Each box represents a software component of the architecture. The assembly connector bridges a component's *required* interface, which is depicted as a socket, with the *provided* interface of another component which is represented by a ball. When the communication between the two components is asynchronous (namely, it is mediated by the Semantic Bus), sockets and ball are gray-highlighted. The rest of the section is dedicated to the presentation of the design of the architecture and the description of its software components.

5.2.1. Behaviors Layer

A C:Behavior⁹ is a software artifact that aims to realize a specific goal. Ex-⁵⁰⁰ amples of Behaviors include "entertain the user", "locate a user", "take user to a place" etc. The C:Behaviors rely on perceptual capabilities of the robot that provide sensor data, perform potentially complex processing (e.g. involving retrieving knowledge from and adding knowledge to the knowledge base), and control robots' actuators and devices in order to operate on the environment and in-

- teract with the user. Each C:Behavior maintains and updates an internal state, and it decides the actions to perform based on sensor data, its state, the general state of the robot and its internal behavior-specific logic. The C:Behaviors expose an interface (i.e. the I:BehaviorControlInterface¹⁰) that allows the C:BehaviorScheduler to control (i.e. start and stop) them. Moreover, in order
- to implement an affordance-based behavior arbitration (the details of this mechanism are provided in the next section), the I:BehaviorControlInterface allows the C:BehaviorScheduler to retrieve the situations that can be managed by the C:Behavior. Using the I:BehaviorControlInterface, the C:Behavior Scheduler can also grants the access to robot's capabilities to the C:Behavior.
- Once granted the use to robot's capabilities: (i) the C:Behavior can use the I:S2TInterface component to make the robot speak; (ii) by using I:HCI ManagerInterface the C:Behaviors can show to users pictures and videos (if the robot is equipped with a tablet) or to receive notifications when a button is pressed by a user; (iii) the C:Behavior can subscribe to the C:Speech2Text
- topic to retrieve what the user says; (iv) the C:Behavior can use the I:Perception AndMotionControllerInterface to retrieve sensor data, and make the robot move in its operating environment. Therefore, the C:Behaviors delegate to this components all the operations needed for making the robot move (e.g. map construction and geometric reasoning).
- 525

The C:Behaviors are also able to store/retrieve knowledge from the C:Know

⁹C: prefix is used for software components.

¹⁰I: prefix is used for software interfaces.

ledgeBase through an C:ObjectOntologyMappingModule. A C:Behavior can also extend (at both intensional and extensional level) the C:KnowledgeBase by generating a new C:ObjectOntologyMappingModule with the I:ManageObject OntologyMappingInterface. In other words, C:Behaviors adopt an ontol-

- ogy¹¹ to structure their data in the C:KnowledgeBase, and, in order to interact with the C:KnowledgeBase they generate an C:ObjectOntologyMappingModu le that complies with the adopted ontology. This solution guarantees that C:Behaviors are decoupled from the technology adopted for the C:Knowledge Base. A C:Behavior may use robot's convoluted capabilities. The architecture
- depicted in Figure 3 shows a single C:ConvolutedCapabilityModule which has to be intended as a placeholder for any possible software components implementing a capability. This module realizes an interface (i.e. I:Convoluted CapabilityModuleInterface) that allows C:Behaviors to use the capability. A C:Behavior can also extend robot's convoluted capabilities by registering
- new components using the I:ManageConvolutedCapabilityModuleInterface of the C:ConvolutedCapabilityManager. The new C:ConvolutedCapability Module is then deployed by the C:HotDeployManager and becomes available to the other C:Behaviors. Since the C:Behaviors communicate with other components either through the C:SemanticBus or via REST APIs, there is no
 restriction on the technology to realize the C:Behaviors. This solution increases the modularity of the architecture thus also favoring the rapid prototyping of the robot. Finally, it is worth noticing that the Supervisory Controller (cf. Section 2) can be seen as a special C:Behavior that enable the remote control of

the robot.

¹¹The choice of which ontology to use depends on the task to perform and, therefore, is left to the behavior developer. For example, the CGA and reminiscence application mentioned in Section 6 operate according two different ontologies of the MARIO Ontology Network.

550 5.2.2. Behavior Scheduler

555

The C:BehaviorScheduler is a special C:Behavior that actively coordinates the other C:Behaviors and manages the functional capabilities of the robot. Specifically, the C:BehaviorScheduler is responsible for: (i) Processing incoming data/events and reasoning over the actual state and available knowledge in order to detect situations that require to activate a C:Behavior;

- (*ii*) Coordinating, scheduling and prioritizing C:Behavior execution; (*iii*) Activating, suspending, resuming and terminating C:Behaviors, as a result of a continuous decision making process; (*iv*) Monitoring C:Behavior executions, to detect successful C:Behavior completions as well as abnormal terminations,
- failures and exceptions. In case of abnormal termination or failure C:Behavior, the C:BehaviorScheduler takes back the control and may either execute a recovery procedure or activate another C:Behavior.

Behavior Arbitration Mechanism. The C:BehaviorScheduler implements a hybrid strategy for arbitrating the C: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 over complex, abstract world models. For example, let the user make a phone call or reminding the user to take his pills does not require a complex control strategy. The

⁵⁷⁰ purely reactive strategy has proven to be effective for a variety of problems that can be completely specified at design-time with simple rules [73]. However, it is inflexible at run time 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 the purely reactive strategy is a behavior-based approach relying on the notion of *affordance*. The notion of affordance has been introduced by Gibson [74] who devised a theory of how animals perceive opportunities for actions. Gibson called this opportunity *affordance*. He suggested that the en-

- vironment offers the agents (people or animals) opportunities for actions. For instance, a door can have the affordance of "openability". The behavior arbitration strategy implemented by the C:BehaviorScheduler exploits and goes beyond the notion of affordance introduced by Gibson. This mechanism is based on the assumption that not only physical 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. More precisely, in our model situations afford (i.e. contribute to select) robot's goals which are then pursued by behaviors which carry out a series of actions to achieve the desired state of the world.
- While a situation can be seen as the fulfillment of certain conditions at a ⁵⁹⁰ certain time, a goal can be interpreted as a certain state of the world the robot desires to achieve. Both may involve temporal aspects (e.g. lunchtime may afford the task to remind 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
- ⁵⁹⁵ user is sitting on a chair for a long while may afford the task entertain the user). The C:BehaviorScheduler continuously checks the current state of the world, and, whenever a condition is satisfied, it retrieves the goals afforded by the fulfilled situations, it selects one of these goals and it then activates a behavior which synthesizes and executes a plan of how to achieve that goal. In
- the context of the H2020 MARIO project, this notion has been formalized as Ontology Design Pattern (ODP) using the MON's Affordance Ontology¹² [75]. Here, we briefly summarize the main ingredients of this pattern which is depicted in Figure 4. The affordance is formalised as a weighted relation (i.e. a specialization of dul:Relation¹³ from the DOLCE foundational ontology),
- which connects a stereotyped situation (i.e. a dul:Description) to either the task to be performed in the situation (i.e. a dul:Task) or to the goal that the agent has to achieve in a given situation (i.e. dul:Goal). This solution

¹²https://w3id.org/MON/affordance.owl

¹³dul: prefix stands for http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

enables two (mutually exclusive) behavior arbitration strategies: (i) A purely-reactive strategy in which the C:Behavior is directly activated at the oc-

currence of a certain situation. For example, if the battery is in critical level, 610 then the robot has to recharge it. (ii) A goal-oriented strategy in which the occurrence of a certain stereotyped situation causes the robot to set a goal, and the choice of the *sequence* of C:Behaviors to activate for achieving the goal is delegated to the C:BehaviorScheduler. Once the goal is set, the C:Behavior

Scheduler activates the C:Behaviors needed for achieving the task. For ex-615 ample, the goal *entertaining the user* might require multiple actions (including approaching the user, and entertaining the user with different activities such as playing musing, showing videos etc.) scheduled according to a certain sequence. The mutual exclusivity between these two strategy is enforced by the

axiom $(\exists hasGoal.dul:Goal) \sqcap (\exists hasTask.action:Task) \sqsubseteq owl:Nothing. The$ 620 affordanceStrength allows to define a prioritization among the affordance relations. For example, if the battery is in critical level, then the robot should not activate the goal "entertaining the user". This can be achieved by assigning to <battery in critical level, recharge> a strength higher than <user annoved, entertain the user>. 625

Interestingly, the affordance relation can be personalised, i.e. each robot may have its own affordances, and it may be adapted over time to best fit with user preferences. Moreover, it is important to note that the arbitration mechanism provides the robot with practical reasoning capabilities [76] (intended as

- the "matter of weighing conflicting considerations for and against competing 630 options"). In fact, a parallelism with Bratman's BDI (Belief-Desire-Intention) model can be done. The robot believes the facts stored in the knowledge base and the data received from its sensors. It uses this information to figure out the current state of the world and to *deliberate* what goals it *desires* to achieve.
- Then the robot commits to the goal to achieve (i.e. intention) and builds a plan 635 to this end.



Figure 4: The Affordance Ontology Design Pattern

Hot-Deployment of Behaviors. The C:BehaviorScheduler provides the robot's designers with the functionalities for easily hot-deploying C:Behaviors on the architecture. Particularly, this component enables the robot's designers to
choose which C:Behaviors (possibly taken from a software repository) to equip the robot with. The chosen C:Behaviors are then effectively installed on the architecture by the C:BehaviorManager. The robot may use the MON's Action Ontology¹⁴ for cataloging C:Behaviors available for the robot and tracing their execution. This ontology, which is depicted in Figure 5, is meant to implement

the Task Execution Ontology Design Pattern¹⁵ in order to keep track of the actions performed either by the robot or by its users (with the indication of possible participants to the Action). C:Behaviors can specified as Tasks. Each Task can be associated with a set of parameters (i.e. the information needed to execute the task).

It is worth noting that the C:BehaviorScheduler guarantees the extensibility, customizability and predictability of the robot's behavior. The extensibility is guaranteed by the fact that the architecture is open to new C:Behaviors that can be dynamically added/removed at runtime (by using the I:Manage BehaviorInterface). In order to customize the robot's behavior, a design just need to modify affordance relations stored in the C:KnowledgeBase. The robot's behavior can be predicted from affordance relations between situations

and C:Behaviors.

¹⁴https://w3id.org/MON/action.owl

 $^{^{15} \}tt http://ontologydesignpatterns.org/wiki/Submissions:TaskExecution$



Figure 5: The MON's Action Ontology.

5.2.3. Basic Capabilities

- This Section describes the components providing the basic robot capabilities. The capabilities enabled by these components are the most significant with respect to the requirements for a social robot (cf. Section 3.1 and 3.2) and that were actually deployed on MARIO (cf. Section 6). The Section omits components providing general purpose services (e.g. network connectivity).
- Text to Speech and Speech to Text. The C:Text2Speech component aims at converting natural language text into speech. The C:Text2Speech implements an interface (i.e. I:T2SInterface) that allows C:Behaviors to synthesize and to reproduce synthesized speech. The C:Speech2Text component converts spoken language into digital-encoded text. The C:Speech2Text component creates a topic for publishing the converted text, the C:Behaviors that need to recognize what users say subscribe to this topic, and, they receive a message whenever the text is converted (by means of the I:S2TInterface). Similarly, the C:Behaviors that need to make the robot speak have to send a mes-
- sage to C:Text2Speech message queue. The asynchronous interaction with
- C:Speech2Text and C:Text2Speech avoids the C:Behaviors to busy wait until
- $_{\rm 675}$ $\,$ text/speech is generated, hence decoupling the C:Behaviors from these tools.

Human Control Interface Manager (HCIManager). Most of the social robots are equipped with some control buttons (e.g. emergency button) and one or more (touch-)screens in order to complement 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 C:Speech2Text and C:Text2Speech components, whereas visual communication is ensured by the C:HCIManager. The C:HCIManager component aims at providing C:Behaviors with facilities for managing the robot's human control interfaces (like buttons or tablets). The component realizes an interface (i.e. the I:HCIManagerInterface) that allows C:Behaviors to show widgets

on the screen or to receive a feedback whenever the user interacts with such widgets or presses a button. Similarly to C:Speech2Text and C:Text2Speech components, the C:Behaviors communicate with the C:HCIManager through the C:SemanticBus. The use of a common C:HCIManager may standardise the interaction between HCIs and C:Behaviors thus increasing the interoperability and the reusability of software components, and favoring the rapid prototyping.

Perception and Motion Controller. The C:PerceptionAndMotionController provides functional capabilities for supporting human-robot and environmentrobot interactions. It includes a set of software components that enable the robot to perform a series of motion routines (e.g., approaching the user, following the user, recharging, driving the user to a destination, etc.). The I:Perception AndMotionControllerInterface provides C:Behaviors with the access to data

coming from several sensors such as: (i) RFID in order to detect a list of tagged objects; (ii) Camera for recognizing users using face and posture recognition, and extracting his/her relative position and distance; (iii) Laser for perceiving and identifying dynamic objects/persons that were not included in the static map (SLAM system). Finally, this component provides C:Behaviors with high-

⁷⁰⁵ level functionalities such as: go to X (where X is a point within the robot's operating area), give me user's position etc. As for the other components providing basic capabilities, the C:PerceptionAndMotionController communicates with the C:Behaviors through the C:SemanticBus. This solution increases the modularity and the interoperability of the software components and favors the rapid prototyping.

5.2.4. Convoluted Capabilities

710

Most of the software applications (not only the robotics ones) that verbally interact with users rely on (general purpose) natural language processing tools to interpret and to reply to users' utterances. Since such services ⁷¹⁵ rely on basic capabilities (i.e. speech-to-text and text-to-speech) for acquiring and reproducing utterances they can be classified as *Convoluted Capabilities*. These services may require non-negligible computational resources, therefore it is desirable to optimize their use as much as possible. In other words, the C:Behaviors should be enabled to share these services instead of re-deploying

- them. To this end, we have applied the hot-deploy pattern (already seen at behavior level). Specifically, C:ConvolutedCapabilityModules are services that can be installed in the robotic platform at runtime by the C:Behaviors that intend to make available new functionalities for other C:Behaviors. The C:ConvolutedCapabilityManager is responsible for the dynamic deployment
- of new C:ConvolutedCapabilityModule. It realizes an interface (i.e. I:Manage ConvolutedCapabilityInterface) which accepts as input an application module realizing the new capability to deploy. Once received a software module, the C:ConvolutedCapabilityManager uses the C:HotDeployManager to deploy it.

5.2.5. Knowledge Management Framework

This Section presents the architectural components responsible for the management of the robot's knowledge. The Knowledge Management Framework consists of two components, namely the C:KnowledgeBase and the C:Object OntologyMappingModule.

Knowledge Base. The C:KnowledgeBase is the component intended to store the robot's knowledge in a structured format. The reference data model for the C:KnowledgeBase is RDF. In our architectural model the C:Knowledge Base provides facilities that allow to create, read, updated, and delete facts. The C:KnowledgeBase doesn't have a predefined and fixed schema, but it is dynamically defined by the C:Behaviors. Finally, the C:KnowledgeBase also

- ⁷⁴⁰ provides a reasoning engine that is able to infer logical consequences (i.e. entailed facts) from the stored facts. It is worth noticing that the adoption of RDF as reference data model increase the interoperability with other Semantic Web compliant systems, and, since it enables the reuse of Linked Open Data datasets, RDF favors also the rapid prototyping.
- Object-Ontology Mapping Module. An C:ObjectOntologyMappingModule is a REST service that provides software components with the access to the C:Know ledgeBase. Given an ontology as input, the C:ObjectOntologyMappingMana ger generates a module that provides software components with REST APIs for accessing the RDF facts stored in the C:KnowledgeBase in a way that:
- (i) reflects the semantics of the ontology, (ii) and, follows the Object-Oriented paradigm. It is easy to recognize the Hot-Deployment pattern already seen for C:Behaviors and Convoluted Capabilities. The Object-Ontology Mapping paradigm avoids software components to deal with RDF, OWL or SPARQL. The Object-Ontology Mapping solution, rather than a direct access to the C:Know
- rss ledgeBase, aims to increase the decoupling between the C:Behaviors and the C:KnowledgeBase.

5.2.6. Semantic Bus

The C:SemanticBus is meant to provide the components with a messagebased asynchronous communication mechanism according to the publish/sub-⁷⁶⁰ scribe paradigm. The C:SemanticBus exposes two interfaces, namely: (i) the I:TopicManagementInterface which allows software components to create new topics (also called messages queues); (ii) the I:PubSubInterface which allows software components to publish messages on/subscribing to a topic. This mechanism decouples the software components.

765 5.2.7. Hot-Deploy Manager

The C:HotDeployManager allows to extend the architecture by dynamically deploying new software components. The C:HotDeployManager aims at providing an OSGi-like¹⁶ platform for the robot's software architecture enabling a dynamic component model. In OSGi, applications come in the form of bundles¹⁷. A bundle can be installed, started, stopped, updated, and uninstalled without requiring a reboot. These features ensure the flexibility and the extensibility of the software architecture.

5.3. RASR Ontology

A long standing tradition in software engineering is to formally specify and document software architectures by means of ontologies [77, 78]. These approaches aim at formally capturing knowledge about a specific architectural solution in order to favor its reuse. Following this line of work, we designed the *RASR Ontology* which defines a model for specifying and documenting social robot architectures. The model comply with the design presented in previous sections. The ontology extends the Ocè ontology [78] by introducing terms to be instantiated for documenting social robot architectures. The ontology is available online at ¹⁸ and is depicted in Figure 6.

Following the Ocè ontology¹⁹ we introduced three kinds of classes, namely subsystems (rasr:Subsystem), components (rasr:Component) and interfaces (rasr:Interface), which respectively represent layers, components and interfaces of the reference architecture. These classes are the main architectural elements that a specific social robot architecture might want to instantiate. To guarantee compliance with Ocè ontology rasr:Subsystem, rasr:Component and rasr:Interface are declared as sub-classes of oce:Subsystem, oce:Component

¹⁶OSGi, https://www.osgi.org/

¹⁷Bundle 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.

¹⁸https://w3id.org/MON/rasr.owl

¹⁹https://kadegraaf.nl/oce-ontology.owl



Figure 6: The RASR Ontology. For brevity the diagram uses the following prefix declaration:
- rasr: https://w3id.org/MON/rasr.owl#

- L: https://w3id.org/MON/rasr.owl#Layer:
- C: https://w3id.org/MON/rasr.owl#Component:
- I: https://w3id.org/MON/rasr.owl#Interface:
- ⁷⁹⁰ and oce:Interface. The ontology defines two object properties: (*i*) provides which relates a component with the interface it provides; (*ii*) includes which links a subsystem (a layer) with the components within the layer. Finally, for each class we defined a restriction which is meant to constraint extension of the properties provides and includes. Specifically:
- Each component class *must* provide the designed interfaces (according the reference architecture). For example, C:Behavior ⊑ ∃ provides.I:BehaviorControlInterface.
 - Each subsystem class is constrained to include the appropriate class of components only. For example, C:Behaviors ⊑ ∀ includes. (C:Behavior or C:BehaviorScheduler).



Figure 7: The MARIO Kompaï-2 robot and the instantiation of the RASR for supporting the CGA and Reminiscence behaviors. The components are colored according to the layer they belong to. Gray boxes indicate behavior-specific software components.

6. An Instantiation of the Reference Architecture

MARIO software architecture is an instantiation of the reference architecture proposed in this article. Specifically, examples of C:Behaviors in MARIO's architecture are the CGA [79] and the Reminiscence [80] behaviors (briefly described in Sections 6.1 and 6.2). These behaviors use MARIO's Basic Capabilities made available through the integration of tools such as Nuance Dragon Naturally Speaking for the C:Speech2Text, Google Speech to Text APIs as C:Text2Speech, and Karto²⁰ as motion control system. Other customized Basic Capabilities were introduced to let C:Behaviors easily control the robot's HCI interface. C:Behaviors share Convoluted Capabilities providing common

NLP services (e.g. Stanford's CoreNLP [81]) for dialoguing with users [82].

²⁰https://wiki.ros.org/karto

C:Behaviors and Capabilities rely on a Knowledge Management Framework [52, 83] consisting of a triple store (accessible through Apache Jena²¹ and initially populated with Linguistic and Common Sense Knowledge [84, 85]) and an

⁸¹⁵ Object-Ontology Mapper called Lizard²² [52]. Finally, the C:SemanticBus is based on the Event Admin Service provided by Apache Felix²³ which also provides the OSGi implementation that enables the Hot-Deploy mechanism.

Evaluation of the RASR Instantiation. We briefly report on the key results of the evaluation undertaken in the MARIO project (more details can be found

- in [86] and [8]). The evaluation involved 38 PWDs of the three different dementia care environments: residential care, hospital and community. The evaluation methodology relied on the use of standardized questionnaires and interviews. In the hospital setting, a significant improvement was observed in patient's depression, resilience, and quality of life. The results also indicated that most
- participants across the three settings were accepting and had positive perceptions/attitude toward MARIO, and the deployment of social robots. PWDs enjoyed their interactions with the robot and they often referred to MARIO with a pronoun (he or she) or "a friend" thus indicating how they had developed a relationship with MARIO. Some of the carers expressed concerns related
- to the fact that robots should not be seen as a replacement for human interaction or care givers. Carers and relatives indicated that working with MARIO positively impacted the level of cognitive engagement of PWDs, since they spent less time alone and more time socially engaged.

6.1. CGA Behavior

835

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. To gather

²¹https://jena.apache.org/

²²https://github.com/anuzzolese/lizard

²³https://felix.apache.org/

information about the patient, physicians rely on a set of widely accepted, internationally validated formal assessment tools (which mostly include question-

- ⁸⁴⁰ naires). The assessment enables the evaluation of a Multidimensional Prognostic Index (MPI), a prognostic tool that derives a single score able to synthetically represent patient's health status. Nowadays, health professionals increasingly use Information Communication Technology (ICT) supporting tools and devices during the assessment phase for recording test results and calculate the
- ⁸⁴⁵ corresponding scores. Moreover, 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.
- MARIO's CGA behavior, whose components are shown in Figure 7, 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 caregivers and physicians in the assessment phase and facilitate the evaluation of the Multidimensional Prognostic Index. The CGA behavior 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 behavior relies on the CGA
- ontology²⁴ of the MARIO Ontology Network. 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. The ontology is modular, namely, the
- ⁸⁶⁵ CGA ontology consists of: (i) a main module²⁴ which defines high-level conceptual characteristics shared by all the tests composing the assessment; (ii) a

²⁴https://w3id.org/MON/cga.owl

module for each test of the assessment procedure for representing their pecularities (as for the MARIO project, ontologies have been developed for the following tests ADL and IADL²⁵, SPMSQ²⁶, MNA²⁷, ESS²⁸, Co-habitation status²⁹ and Medication Use^{30}).

870

Results of the evaluation. An autonomous CGA assessment undertaken by a robot was found acceptable to participants (both PWDs and caregivers). The findings also indicated that an autonomous assessment allows healthcare professionals to focus on other care activities and thus optimizing their time.

6.2. Reminiscence Behavior 875

880

890

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. MARIO's approach focused on robot-enabled delivery of so-called *simple remi*-

niscence [87], based on a conversational approach and highly focused verbal and visual memory triggers. The application, whose main components are shown in Figure 7, is thus specifically designed to actively prompt the Person With Dementia (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. 885

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 modules supporting the reminiscence therapy are:

²⁵https://w3id.org/MON/capabilityassessment.owl

²⁶https://w3id.org/MON/spmsq.owl

 $^{^{27} \}rm https://w3 id.org/MON/mna.owl$

²⁸https://w3id.org/MON/ess.owl

²⁹https://w3id.org/MON/cohabitationstatus.owl

³⁰https://w3id.org/MON/medicationuse.owl

- *Person*³¹ which addresses the need of representing basic biographic information, family and social relationships among them;
- *Personal Events*³² which allows to specify PWD's life events (such as, marriage, employment etc.) according to a series of dimension (i.e. time, location, participants to the event, multimedia objects associated with the event);
- The association between media objects and other entities relies on a semantic tagging approach, as defined in a *tagging* ontology module³³ 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.

Results of the evaluation. The Reminiscence Behavior was widely used and selected by PWDs across all the care settings. PWDs enjoyed using the Behavior particularly in looking at the materials on MARIO. Relatives and carers in all settings also commented on the importance of this application for the PWD.

7. Evaluation

895

900

In this section we conduct a scenario-based evaluation of the reference architecture. The scenario-based approaches are considered the most mature ⁹¹⁰ methodologies for assessing architectural solutions [88, 89, 90] and the Architecture Tradeoff Analysis Method (ATAM) [91] is one of the most used. We adopted a lightweight version of ATAM [92] which consists of five steps: *(i)* Present the architectural drivers; *(ii)* Present the architecture; *(iii)* Identify architectural approaches; *(iv)* Generate a utility tree; *(v)* Analyze the architectural approaches with respect to the defined scenarios. Steps *(i)*, *(ii)* and *(iii)* have

³¹https://w3id.org/MON/person.owl

³²https://w3id.org/personalevents.owl

³³https://w3id.org/MON/tagging.owl

been discussed in the above sections. In this section, we focus on the steps (iv) and (v). Concerning the step (iv), a utility tree aims at refining the overall *utility* (an expression for indicating the overall "goodness" of a system) into specific quality attributes (i.e. the architectural drivers presented in Section 3)

⁹²⁰ and associating each quality attribute with one or more scenarios. A scenario is a natural language description of how users will interact with the system. Then, the last step assesses how well a software architecture satisfies the scenarios.

7.1. Scenarios

From the use cases of the MARIO project [7], we elicited 16 scenarios involving the Robot and two kinds of robot's users the Person With Dementia (PWD) and the Care Staff. We assume that the Robot provides an authentication mechanism for distinguishing the Care Staff from the PWD. The scenarios are listed below.

- S.LinguisticData ³⁴ The Robot uses linguistic data to strengthen its natural language un-930 derstanding skills.
 - **S.POI** The Care Staff identifies and names the points of interest (PoI) on the internal map of the operating environment captured by the Robot.
 - S.Help The PWD tells the Robot "I'm in trouble" and the Robot calls a help line.
 - S.TækeMeTo The PWD says "Take me to X", where X is a PoI. The Robot uses the labelled map of its operating environment to take the PWD to X.
 - S.Recognize The Robot identifies the PWD in the immediate area or searches for him/her in its operating environment, and approaches the PWD.
 - S.Questions The Robot informs the PWD with information about people and locations.
 For example, the Robot is able to answer simple questions of the form: "Who am I?", "Who are you?", "Where am I?"

 $^{^{34}}$ S. prefix is used for scenarios.

- S.Games The Robot is equipped with a range of predefined games, but the PWD wants to play a game not provided by default. The Robot allows the Care Staff to install new games. The Care Staff can also choose the most suitable games for the PWD.
- S.Music The Robot plays some music for the PWD and learns from his/her requests the favorite songs of the PWD. The Robot is able to explain why the songs proposed to the PWD are considered the favorite ones.
- S.Entertain The Robot offers to entertain the PWD by proposing to play games, listen to some music, read a book, show a video, etc. The Robot detects the right moment to offer entertainment to the PWD and the offerings are based on what the PWD has interacted with in the past. The Robot records PWD preferences during use in order to propose doing the most suitable activities. The Robot is able to explain why the proposed activity is the most suitable for the PWD and why is an appropriate time for it.
- S.PersonalData The Robot allows the Care Staff to load and edit data about the PWD. The Robot uses this data (such as the PWD's name, birthday and data about friends and family including contact details) to adapt its behavior and to become more personable and friendly.
 - S.Actaivities The Care Staff defines and uploads to the Robot a set of sequences of daily living tasks to prompt for each PWD. The Robot prompts the activities as defined by the Care Staff.
- S.Reminiscence The Robot uses the personal data of the PWD (e.g. photos, videos etc.) for reminiscing events, people and places from their past lives. The Robot ⁹⁶⁵ may use materials precedently loaded by the Care Staff as well as items from generic knowledge stores that match the PWD's era. The Robot records PWD preferences during use in order to trigger the PWD with the most suitable material. The Robot is able to explain why it considers the proposed material as the most suitable for the PWD.

- S.Reminders The Robot reminds the PWD about events and daily living schedules (e.g. meal times, bed time, etc.). The schedules are pre-loaded by the Care Staff.
- S.Knowledge The Robot is able to simple questions of the form "What is X?", "What happened on this day in history?".
- S.Assessment The Care Staff defines a questionnaire for assessing the cognitive, medical, psychosocial and functional status of the PWD. The Robot automatically performs the questionnaire and reports the results to the Care Staff.
- S.Prompting The Robot proactively proposes doing the most suitable activities to its PWD. The Robot is able to explain why it considers the activities as the most suitable for the PWD.

Each scenario may be associated with one or more drivers. The association of the scenarios with the architectural drivers is shown in Table A.1. All the scenarios require the robot to be able to interact (Interacting) with either the Care Staff or the PWD. Therefore, if the Robot gathers and uses existing linguis-

- ⁹⁸⁵ tic data (Interoperability) to strengthen its understanding capabilities (cf. S.LinguisticData), then, all the scenarios would benefit from them. S.POI implies that the Robot can perceive, move within (Perceiving), and learn (Learning) about its operating environment environment. These abilities are reused in the scenario S.TakeMeTo, S.Recognize and S.Help which also builds
- ⁹⁹⁰ upon the abilities of understanding simple PWDs requests (cf. S.Questions which also benefits of the ability of perceiving the environment) and perceiving the environment (cf. S.POI). S.Games requires the ability of *(i)* extending the architecture with new components in order to customize the robot according to the PWD's preferences (Customizability); *(ii)* reusing existing software
- ⁹⁹⁵ components (Reusability); (iii) interoperating with other systems in order to dynamically integrate new software artefacts (Interoperability). In the S.Music and S.Entertain scenario, the robot is supposed to be able to learn (Learning) the favorite songs or entertaining activities of the PWD so that it

 ${\tt can} \ (deterministically\ {\tt - Explainability}) \ adapt \ the \ playlist \ according \ to \ her/his$

- 1000 preferences (Customizability). The robot can also adapt (Customizability)
 its behavior by reusing PWD's data (S.PersonalData), sequence of activities
 (S.Activities) and the reminders (S.Reminders) loaded by the Care Staff.
 In order to satisfy the S.Reminiscence scenario, the robot should be able to
 (i) learn (Learning) from the past what the PWD likes; (ii) adapt the proposed
- norm material according to the PWD's preferences (Customizability); (iii) explain the rationale of the chosen of a proposed material (Explainability); (iv) re-use (Reusability) general knowledge possibly gathered from the web (Interoper ability). The ability of reusing (Reusability) general knowledge gathered from the web (Interoperability) is also required for the S.Knowledge sce-
- nario. S.Assessment scenario builds upon the ability of interacting with the environment (Perceiving) (some tests assess the mobility of the PWD) and of performing a suitable test for each PWD (Customizability). The S.Prompting scenario is satisfied as long as the C:Behaviors interoperate (Interoperability) for collecting information about PWD's preferences and the robot is able to explain why a certain activity is prompted (Explainability).

7.2. Analysis of the Architectural Approaches

We conduct two kinds of analysis aiming at assessing (i) what components of the reference architecture are involved in which scenario; (ii) how the architecture presented in Section 2 satisfy the architectural drivers.

- Analysis of the architectural components with respect to the scenarios. Table A.2 shows how the architectural components are involved in the scenarios. In general, we note that all the components play a role in at least two scenarios. As already observed in Section 7.1, all the scenarios require the ability of (verbally) interacting with either the PWD or the Care Staff. This implies
- that C:Text2Speech, C:Speech2Text and (consequently) the C:SemanticBus are involved in all the scenarios. Moreover, in all the scenarios at least a C:Behavior must be activated by the C:BehaviorScheduler. We also observe that both the C:KnowledgeBase and the C:ObjectOntologyMappingMo

dule are involved in almost all the scenarios (14 out of 16), thus witness-

ing the centrality of the C:KnowledgeBase in the architecture. These components are used for integrating data coming from external sources (S.Lingui sticData), for storing and retrieving information about the operating environment (S.POI and S.TakeMeTo), for accessing general knowledge (S.Questions, S.Reminiscence, S.Knowledge), for registering PWD's data (S.PersonalData

- and S.Assessment), schedules (S.Activities and S.Reminders) and preferences (S.Music, S.Games, S.Prompting and S.Entertain). The C:Knowledge Base is not involved in the purely reactive scenarios such as S.Recognize and S.Help. C:BehaviorManager, C:ConvolutedCapabilityManager, C:Object OntologyMappingManager and C:HotDeployManager play a role in the scenarios
- that require to extend the architecture with new applications (S.Games) or capabilities (S.LinguisticData). The C:HCIManager takes part to the scenarios in which the robot engages an interaction with the Care Staff. Finally, the C:Perce ptionAndMotionController is involved in all the scenarios in which the robot wander around the operating environment (S.POI, S.Help and S.TakeMeTo).
- Analysis of the surveyed architectures with respect to the architectural drivers. Evaluating the existing architectures with respect to MARIO's use cases would unjustifiedly penalize software architectures designed for addressing different scenarios. Therefore, we compared the existing architectures with respect to the architectural drivers defined in Section 3 (which indirectly reflect the scenarios).
- ¹⁰⁵⁰ To do so, for each architecture, we analyzed the description provided in the related reference paper in order to figure out whether the architecture fulfill the requirement. The result of this analysis is shown in Table A.3. This Table gives us an overview of how the architectural drivers are addressed in the software architectures for social robots.
- We observe that all the architectures satisfy both the Perceiving and the Interacting drivers. This testifies the centrality of the interacting capabilities in social robotics. Customizing robot's behavior on the basis of users' needs is crucial in half of the surveyed architectures and others (e.g. Dehkordi2015)

plan to address this requirement in future. However, by analyzing the corre-

lation between the Customizability and the Learning driver (the Pearson's coefficient scores 0.29), we can deduce that most of social robots don't adapt their behavior automatically but require the human intervention. Twelve architectures have a mechanism for extending the robot behavior. In some cases (e.g. Breazeal2004), the extension does not imply the deployment of a new software

- 1065 component at runtime, but the learning of a new ability by the robot. One third of the architectures consider Reusability as a core requirement. Inter operability and Explainability are irrelevant requirements for most of the surveyed architectures. Specifically, only three architectures (Bonaccorsi2016, Sarabia2011 and Coronado2021) consider Interoperability as an impor-
- tant quality to be addressed by the architecture, but the Interoperability is mainly intended at software level only. At data level, none of the surveyed architectures adopt standard languages for syntactic and semantic interoperability. We note that these technologies are also poorly adopted in Robotics (only a few examples exist [93, 94]).
- Discussion. The proposed software architecture addresses all the elicited drivers and copes with all the collected scenarios. In particular, the architecture guarantees the interoperability and explainability of social robots by-design, thus significantly advancing the state-of-the-art. In fact, despite the relevance of these requirements (as the recent European AI regulations also remark see
 Section 1.1), the existing architectures barely address the interoperability driver

and completely ignore explainability issues (cf. Table A.3).

The utility of the architecture has been assessed against a collection of social robotics scenarios coming from the assistive context, therefore the generalisability of the proposed solution is to be further assessed with respect to additional

scenarios. Moreover, it has to be noted that the elicited drivers mainly focus on the desired features of the robot and to a limited extent on the characteristics of its users. This could limit the usability of the robot by categories of users (e.g. people with disabilities). Although the usability requirements mainly influence the development of the user interface, investigating how usability affects

the design of the software architecture is a direction worth further research. In light of this consideration, the Universal Design principles [95], i.e. the design of products and environments to be usable to the greatest extent possible by people of all ages and disabilities, is relevant for further developing the reference architecture towards a better support to "universal" interfaces. It is worth noticing that Universal Design has already been applied to robot design (e.g. [96]), but implications on software architecture of such design choices are less studied and they are worth further investigation.

8. Conclusions and Future Work

This paper proposed a reference software architecture for social robots. This architecture aims at improving the acceptability of robots, guaranteeing that robots establish a personalized relationship with their users, and to make robots more trustworthy. We have adopted a bottom-up approach for defining general architectural requirements starting from specific use cases defined in the context of MARIO project. These requirements led us to the selection of a set of technologies, architectural styles and patterns, that guided the design of the reference architecture. The reference architecture introduces a series of standardized software components that are meant to increase the extensibility, customizability, predictability, interoperability of the software architecture as well as to favor the rapid prototying of the Social Robotics solutions. Moreover, to favor the interoperability among the architectures, we introduced the *RASR*

Ontology which defines a common vocabulary of terms that enables robot designers to formally specify their software architectures. The architecture was evaluated with the ATAM methodology.

The ultimate goal of this work is to establish a common open-ended software architecture so as to encourage the development of standardized software components for social robots. As shown by the instantiation presented in Section 6, these components can be provided by off-the-shelf software tools or are available as research prototypes. In the context of the MARIO project these components were (forcibly) integrated in order to provide a proof-of-concept for the architec-

- ture. But a standardization effort is needed for improving the reusability of the components. On the basis of the experience in the presented case study, we are turning proof-of-concept components into standardized and easily deployable software artifacts. Particularly, we are generalizing the protocols used in the MARIO architecture and we are developing an affordance-inspired C:Behavior
- Scheduler. Moreover, in order to endow robots with sophisticated behavior and capabilities, software artifacts can be adapted (or wrapped) in order to be integrated in the architecture (to this end the Adapter Pattern [97] can be considered). In addition to the development of new components, it is also necessary to define a set of standard protocols that let components communicate.
- ¹¹³⁰ We also plan to explore the possibility of enhancing the autonomy of robot by design. To this end, an autonomic computing solution [98] is being considered in order to endow the robot with self-configuration, self-optimization, self-healing and self-protection capabilities. Furthermore, the design focused was lead by the main social drivers (i.e. acceptability, trust and personalized interaction)
- that favor the establishment of a human-robot relation. However, design impact of additional drivers (such as safety [99]) have to be carefully assessed in the future. Finally, we assessed the general utility of the architecture on social robotics scenarios but we plan to evaluate the capability of the architecture to be adaptable to different scenarios.

1140 Acknowledgements

1145

The research leading to these results has received funding from the European Union Horizons 2020 the Framework Programme for Research and Innovation (2014-2020) under grant agreement 643808 Project MARIO "Managing active and healthy aging with use of caring service robot". P.Ciancarini also thanks CINI for partial support.

References

1160

- T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, F. Tanaka, Social robots for education: A review, Science Robotics 3 (21) (2018) 1–9. doi:10.1126/scirobotics.aat5954.
- [2] J. Broekens, M. Heerink, H. Rosendal, Assistive social robots in elderly care: a review, Gerontechnology 8 (2) (2009) 94–103.
 - [3] K. Dautenhahn, Getting to know each other artificial social intelligence for autonomous robots, Robotics and Autonomous Systems 16 (2-4) (1995) 333-356. doi:10.1016/0921-8890(95)00054-2.
- [4] A. Ahmad, M. A. Babar, Software architectures for robotic systems: A systematic mapping study, Journal of Systems and Software 122 (2016) 16–39. doi:10.1016/j.jss.2016.08.039.
 - [5] K. Dautenhahn, B. Ogden, T. Quick, From embodied to socially embedded agents implications for interaction-aware robots, Cognitive Systems Research 3 (3) (2002) 397-428. doi:10.1016/S1389-0417(02)00050-5. URL https://doi.org/10.1016/S1389-0417(02)00050-5
 - [6] M. A. Goodrich, A. C. Schultz, Human-robot interaction: a survey, Foundations and Trends in Human-Computer Interaction 1 (3) (2008) 203-275. doi:10.1561/1100000005.
- III65 [7] D. 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
 - [8] A. Mannion, S. Summerville, E. Barrett, M. Burke, A. Santorelli, C. Kruschke, H. Felzmann, T. Kovacic, K. Murphy, D. Casey, et al., Introducing the social robot MARIO to people living with dementia in long term residential care: Reflections, International Journal of Social Robotics (2019) 1–13.

- [9] C. Bartneck, J. Forlizzi, A design-centred framework for social humanrobot interaction, in: Proceedings of the 13th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN 2004), IEEE, 2004, pp. 591–594.
- [10] C. Breazeal, Toward sociable robots, Robotics and Autonomous Systems
 42 (3) (2003) 167 175. doi:10.1016/S0921-8890(02)00373-1.
- [11] K. Dautenhahn, The art of designing socially intelligent agents: Science, fiction, and the human in the loop, Applied artificial intelligence 12 (7-8) (1998) 573-617.
- [12] B. R. Duffy, C. Rooney, G. M. O'Hare, R. O'Donoghue, What is a social robot?, in: Proceedings of the 10th Irish Conference on Artificial Intelligence & Cognitive Science, 1999, pp. 1–7.
- ¹¹⁸⁵ [13] B. R. Duffy, The social robot, Ph.D. thesis, University College Dublin (2000).
 - [14] T. Fong, I. R. Nourbakhsh, K. Dautenhahn, A survey of socially interactive robots, Robotics and Autonomous Systems 42 (3-4) (2003) 143–166. doi:10.1016/S0921-8890(02)00372-X.
- [15] J. F. Kelley, An iterative design methodology for user-friendly natural language office information applications, ACM Transactions on Information Systems 2 (1) (1984) 26–41. doi:10.1145/357417.357420.
 - [16] L. D. Riek, Wizard of oz studies in hri: A systematic review and new reporting guidelines, Journal of Human-Robot Interaction 1 (1) (2012) 119–136. doi:10.5898/JHRI.1.1.Riek.
 - [17] A. Dillon, User acceptance of information technology, in: Encyclopedia of human factors and ergonomics, London: Taylor and Francis, 2001, pp. 1–10.

1175

- [18] A. R. Wagner, The role of trust and relationships in human-robot social interaction, Ph.D. thesis, Georgia Institute of Technology (2009).
- [19] B. Kuipers, How can we trust a robot?, Communications of the ACM 61 (3) (2018) 86–95. doi:10.1145/3173087.
- [20] M. K. Lee, J. Forlizzi, S. Kiesler, P. Rybski, J. Antanitis, S. Savetsila, Personalization in hri: A longitudinal field experiment, in: Proceedings of the 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2012, pp. 319–326.
- [21] G. Gordon, S. Spaulding, J. K. Westlund, J. J. Lee, L. Plummer, M. Martinez, M. Das, C. Breazeal, Affective personalization of a social robot tutor for children's second language skills, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016), 2016, pp. 3951–3957.
- [22] P. Alves-Oliveira, S. Gomes, A. Chandak, P. Arriaga, G. Hoffman, A. Paiva, Software architecture for yolo, a creativity-stimulating robot, SoftwareX 11 (2020) 100461. doi:10.1016/j.softx.2020.100461.
- [23] M. Bonaccorsi, L. Fiorini, F. Cavallo, A. Saffiotti, P. Dario, A cloud robotics solution to improve social assistive robots for active and healthy aging, International Journal of Social Robotics 8 (3) (2016) 393–408. doi:10.1007/s12369-016-0351-1.
- [24] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, D. Mulanda, Humanoid robots as cooperative partners for people, International Journal of Humanoid Robots 1 (2) (2004) 1–34.
- [25] H.-L. Cao, P. G. Esteban, A. D. Beir, R. Simut, G. V. D. Perre, B. 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), 2016, pp. 1–5.
- [26] H. Cao, G. V. de Perre, J. Kennedy, E. Senft, P. G. Esteban, A. D. Beir,
 R. Simut, T. Belpaeme, D. Lefeber, B. Vanderborght, A personalized

1210

1205

1215

1220

and platform-independent behavior control system for social robots in therapy: Development and applications, IEEE Transactions on Cognitive and Developmental Systems 11 (3) (2019) 334–346. doi:10.1109/TCDS. 2018.2795343.

- [27] J. Casas, N. C. Gomez, E. Senft, B. Irfan, L. F. Gutiérrez, M. Rincón, M. Múnera, T. Belpaeme, C. A. Cifuentes, Architecture for a social assistive robot in cardiac rehabilitation, in: Proceedings of the IEEE 2nd Colombian Conference on Robotics and Automation (CCRA 2018), 2018, pp. 1–6.
- [28] S. Coşar, M. Fernandez-Carmona, R. Agrigoroaie, J. Pages, F. Ferland, F. Zhao, S. Yue, N. Bellotto, A. Tapus, Enrichme: Perception and interaction of an assistive robot for the elderly at home, International Journal of Social Robotics (2020) 1–27doi:10.1007/s12369-019-00614-y.
- [29] E. Coronado, D. Deuff, P. Carreno-Medrano, L. Tian, D. Kulic, S. Sumartojo, F. Mastrogiovanni, G. Venture, Towards a modular and distributed end-user development framework for human-robot interaction, IEEE Access 9 (2021) 12675–12692. doi:10.1109/ACCESS.2021.3051605.
 - [30] P. S. Dehkordi, H. Moradi, M. Mahmoudi, H. R. Pouretemad, The design, development, and deployment of roboparrot for screening autistic children, Interantional Journal of Social Robotics 7 (4) (2015) 513–522. doi:10.

1007/s12369-015-0309-8.

- [31] J. Fan, D. Bian, Z. Zheng, L. Beuscher, P. A. Newhouse, L. C. Mion, N. Sarkar, A robotic coach architecture for elder care (rocare) based on multi-user engagement models, IEEE Transactions on Neural Systems and Rehabilitation Engineering 25 (8) (2017) 1153–1163. doi: 10.1109/TNSRE.2016.2608791.
- [32] J. Fasola, M. J. Matarić, A socially assistive robot exercise coach for the elderly, Journal of Human-Robot Interaction 2 (2) (2013) 3–32. doi: 10.5898/JHRI.2.2.Fasola.

1230

1235

1245

1250

- [33] V. González-Pacheco, A. A. Ramey, F. Alonso-Martín, Á. Castro González, M. A. Salichs, Maggie: A social robot as a gaming platform, Interantional Journal of Social Robotics 3 (4) (2011) 371–381. doi:10.1007/s12369-011-0109-8.
- [34] M. Á. González-Santamarta, F. J. Rodríguez-Lera, C. Álvarez-Aparicio, 1260 Á. M. Guerrero-Higueras, C. Fernández-Llamas, Merlin a cognitive architecture for service robots, Applied Sciences 10 (17) (2020) 1-17. doi: 10.3390/app10175989.
 - [35] H. Gross, C. Schroeter, S. Mueller, M. Volkhardt, E. Einhorn, A. Bley, T. Langner, C. Martin, M. 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, pp. 2481-2488. doi:10.1109/ICSMC.2011.6084050.
 - [36] H. Gross, C. Schroeter, S. Müller, M. Volkhardt, E. Einhorn, A. Bley,
- T. Langner, M. Merten, C. A. G. J. Huijnen, H. van den Heuvel, A. van 1270 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), 2012, pp. 637-644. doi:10.1109/ICSMC.2012.6377798.
- [37] J. Hirth, N. Schmitz, K. Berns, Towards social robots: Designing an 1275 emotion-based architecture, International Journal of Social Robotics 3 (3) (2011) 273-290. doi:10.1007/s12369-010-0087-2.
 - [38] C. Jayawardena, I.-H. Kuo, E. Broadbent, B. A. MacDonald, Socially assistive robot healthbot: Design, implementation, and field trials, IEEE Systems Journal 10 (3) (2016) 1056-1067. doi:10.1109/JSYST.2014. 2337882.
 - [39] J. Kim, I. Jeong, I. Park, K. Lee, Multi-layer architecture of ubiquitous robot system for integrated services, Interantional Journal of Social Robotics 1 (1) (2009) 19-28. doi:10.1007/s12369-008-0005-z.

- [40] W. G. Louie, T. S. Vaquero, G. Nejat, J. C. 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), 2014, pp. 5292–5298. doi:10.1109/ICRA.2014.
 6907637.
- [41] W.-Y. G. Louie, G. Nejat, A social robot learning to facilitate an assistive group-based activity from non-expert caregivers, International Journal of Social Robotics (2020) 1–18doi:10.1007/s12369-020-00621-4.
 - [42] R. Mead, E. Wade, P. Johnson, A. S. Clair, S. Chen, M. J. Mataric, An architecture for rehabilitation task practice in socially assistive humanrobot interaction, in: C. A. Avizzano, E. Ruffaldi (Eds.), Proceedings of 19th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2010), IEEE, 2010, pp. 404–409. doi:10. 1109/ROMAN.2010.5598666.

1300

- [43] D. Portugal, P. Alvito, E. Christodoulou, G. Samaras, J. Dias, A study on the deployment of a service robot in an elderly care center, International Journal of Social Robotics 11 (2) (2019) 317–341. doi:10.1007/ s12369-018-0492-5.
 - [44] M. A. Salichs, Á. C. González, E. Salichs, E. Fernández-Rodicio, M. Maroto-Gómez, J. J. Gamboa-Montero, S. Marques-Villarroya, J. C.
- Castillo, F. Alonso-Martín, M. Malfaz, Mini: A new social robot for the elderly, International Journal of Social Robotics 12 (6) (2020) 1231–1249.
 doi:10.1007/s12369-020-00687-0.
 - [45] M. Sarabia, R. Ros, Y. Demiris, Towards an open-source social middleware for humanoid robots, in: Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2011), 2011, pp. 670–675. doi:10.1109/Humanoids.2011.6100883.
 - [46] C. Shi, S. Satake, T. Kanda, H. Ishiguro, A robot that distributes flyers to

pedestrians in a shopping mall, International Journal of Social Robotics 10 (4) (2018) 421–437. doi:10.1007/s12369-017-0442-7.

- [47] E. Torta, F. Werner, D. O. Johnson, J. F. Juola, R. H. Cuijpers, M. Bazzani, J. Oberzaucher, J. Lemberger, H. Lewy, J. Bregman, Evaluation of a small socially-assistive humanoid robot in intelligent homes for the care of the elderly, Journal of Intelligent and Robotic Systems 76 (1) (2014) 57-71. doi:10.1007/s10846-013-0019-0.
- [48] P. Uluer, N. Akalin, H. Köse, A new robotic platform for sign language tutoring - humanoid robots as assistive game companions for teaching sign language, Interantional Journal of Social Robotics 7 (5) (2015) 571–585. doi:10.1007/s12369-015-0307-x.
 - [49] L. J. Wood, A. Zaraki, B. Robins, K. Dautenhahn, Developing kaspar: a humanoid robot for children with autism, International Journal of Social Robotics (2019) 1–18doi:10.1007/s12369-019-00563-6.
 - [50] A. Zibafar, E. Saffari, M. Alemi, A. Meghdari, L. Faryan, A. G. Pour, A. RezaSoltani, A. Taheri, State-of-the-art visual merchandising using a fashionable social robot: Roma, International Journal of Social Robotics (2019) 1–15doi:10.1007/s12369-019-00566-3.
 - [51] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: G. Visaggio, M. T. Baldassarre, S. G. Linkman, M. Turner (Eds.), Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE 2008), 2008, pp. 1–10.

1335

1325

- [52] L. Asprino, Engineering background knowledge for social robots, Ph.D. thesis, University of Bologna (2019). URL http://amsdottorato.unibo.it/9020/
- [53] R. Barber, M. A. Salichs, A new human based architecture for intelligent

- autonomous robots, in: Proceedings of the Fourth IFAC Symposium on Intelligent Autonomous Vehicles, 2001, pp. 85–90.
- [54] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, in: Proc. Workshop on Open Source Software in Robotics (co-located with ICRA), Vol. 3.2, 2009, p. 5.
- [55] H. Yan, M. H. A. Jr., A. N. Poo, A survey on perception methods for human-robot interaction in social robots, International Journal of Social Robotics 6 (1) (2014) 85–119. doi:10.1007/s12369-013-0199-6.
- [56] B. R. Duffy, Anthropomorphism and the social robot, Robotics and Autonomous Systems 42 (3) (2003) 177 - 190. doi:10.1016/ S0921-8890(02)00374-3.
- [57] C. L. Breazeal, Sociable machines: Expressive social exchange between humans and robots, Ph.D. thesis, Massachusetts Institute of Technology (2000).
- [58] B. R. Duffy, G. Joue, J. Bourke, Issues in assessing performance of social robots, in: Proceedings of the 2nd WSEAS International Conference (RODLICS 2002), 2002, pp. 1–8.
 - [59] K. Dautenhahn, T. Christaller, Remembering, rehearsal and empathytowards a social and embodied cognitive psychology for artifacts, in: S. O'Nuallain, P. Mc Kevitt, J. Benjamins (Eds.), Two sciences of the mind. Readings in cognitive science and consciousness, North America Inc., 1995, pp. 257–282.
 - [60] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, M. Barreto, Ubiquitous robotics: Recent challenges and future trends, Robotics and Autonomous Systems 61 (11) (2013) 1162 – 1172. doi:10.1016/j.robot. 2013.04.003.

1345

1350

1360

- [61] R. Bailey, K. Wise, P. Bolls, How avatar customizability affects children's arousal and subjective presence during junk food–sponsored online video games, Cyber Psychology & Behavior 12 (3) (2009) 277–283.
- [62] D. Fischinger, P. Einramhof, K. E. Papoutsakis, W. Wohlkinger, P. Mayer,
 P. Panek, S. Hofmann, T. Körtner, A. Weiss, A. A. Argyros, M. Vincze,
 Hobbit, a care robot supporting independent living at home: First prototype and lessons learned, Robotics and Autonomous Systems 75 (2016)
 60–78. doi:10.1016/j.robot.2014.09.029.
- [63] F. Alaieri, A. Vellino, Ethical decision making in robots: Autonomy, trust and responsibility - autonomy trust and responsibility, in: Proceedings of the 8th International Conference on Social Robotics (ICSR 2016), 2016, pp. 159–168. doi:10.1007/978-3-319-47437-3_16.

[64] P. Wegner, Interoperability, ACM Computing Surveys (CSUR) 28 (1) (1996) 285–287.

1380

1385

- [65] V. Mayoral, A. Hernández, R. Kojcev, I. Muguruza, I. Zamalloa, A. Bilbao, L. Usategi, The shift in the robotics paradigm the hardware robot operating system (h-ros); an infrastructure to create interoperable robot components, in: Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2017), 2017, pp. 229–236.
- [66] C. Bartneck, J. Hu, Rapid prototyping for interactive robots, in: Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8), 2004, pp. 136–145.
- [67] J. Won, K. J. D. Laurentis, C. Mavroidis, Rapid prototyping of robotic systems, in: Proceedings of th IEEE International Conference on Robotics and Automation, (ICRA), 2000, pp. 3077–3082. doi:10.1109/ROBOT. 2000.845136.
 - [68] R. C. Arkin, Behavior-based robotics, MIT press, 1998.

- [69] M. P. Papazoglou, Service-oriented computing: concepts, characteristics
- 1395

and directions, in: Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003), 2003, pp. 3–12. doi: 10.1109/WISE.2003.1254461.

[70] T. Friese, M. Smith, B. Freisleben, Hot service deployment in an ad hoc grid environment, in: Proceedings of the 2nd International Conference of Service-Oriented Computing (ICSOC 2004), 2004, pp. 75–83. doi:

1400

1410

- of Service-Oriented Computing (ICSOC 2004), 2004, pp. 75–83. (10.1145/1035167.1035179.
- [71] D. D. Corkill, Blackboard systems, AI Expert 6 (9) (1991) 40–47.
- [72] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Scientific american 284 (5) (2001) 34–43.
- [73] M. J. Mataric, Behaviour-based control: examples from navigation, learning, and group behaviour, Journal of Experimental & Theoretical Artificial Intelligence 9 (2-3) (1997) 323–336. doi:10.1080/095281397147149.
 - [74] J. Gibson, The theory of affordances, in: Perceiving, acting, and knowing: Toward an ecological psychology, Lawrence Erlbaum Associates, 1997, pp. 67–82.
 - [75] L. Asprino, A. G. Nuzzolese, A. Russo, A. Gangemi, V. Presutti, S. Nolfi, An ontology design pattern for supporting behaviour arbitration in cognitive agents, in: K. Hammar, P. Hitzler, A. Krisnadhi, A. Lawrynowicz, A. G. Nuzzolese, M. Solanki (Eds.), Advances in Ontology Design and Patterns, IOS Press, 2017, pp. 85–95.
 - [76] M. Bratman, Intention, plans, and practical reason, Harvard University Press Cambridge, MA, 1987.
 - [77] C. A. Welty, D. A. Ferrucci, A formal ontology for re-use of software architecture documents, in: Proceedings of the 14th IEEE International
- ¹⁴²⁰ Conference on Automated Software Engineering (ASE 1999), IEEE Computer Society, 1999, pp. 259–262. doi:10.1109/ASE.1999.802304.

- [78] K. A. de Graaf, A. Tang, P. Liang, H. van Vliet, Ontology-based software architecture documentation, in: Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2012), 2012, pp. 121–130. doi:10.1109/WICSA-ECSA.212.20.
- [79] L. Asprino, A. Gangemi, A. G. Nuzzolese, V. Presutti, D. Reforgiato Recupero, A. Russo, Autonomous comprehensive geriatric assessment, in: Bastianelli et al. [100], pp. 41–45.
- [80] L. Asprino, A. Gangemi, A. G. Nuzzolese, V. Presutti, A. Russo, Knowledge-driven support for reminiscence on companion robots, in: Bastianelli et al. [100], pp. 51–55.
 - [81] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, D. Mc-Closky, The stanford corenlp natural language processing toolkit, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), 2014, pp. 55–60. doi:10.3115/v1/p14-5010.
 - [82] A. Russo, G. D'Onofrio, A. Gangemi, F. Giuliani, M. Mongiovi, F. Ricciardi, F. Greco, F. Cavallo, P. Dario, D. Sancarlo, V. Presutti, A. Greco, Dialogue systems and conversational agents for patients with dementia: The human-robot interaction, Rejuvenation research 22 (2) (2019) 109–120. doi:10.1089/rej.2018.2075.
 - [83] L. Asprino, A. Gangemi, A. G. Nuzzolese, V. Presutti, A. Russo, A knowledge management system for assistive robotics, in: Bastianelli et al. [100], pp. 46–50.
- [84] L. Asprino, V. Basile, P. Ciancarini, V. Presutti, Empirical analysis of foundational distinctions in linked open data, in: J. Lang (Ed.), Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 18), International Joint Conferences on Artificial Intelligence, 2018, pp. 3962–3969. doi:10.24963/ijcai.2018/551.

1425

- [85] A. Gangemi, M. Alam, L. Asprino, V. Presutti, D. Reforgiato Recupero, Framester: A wide coverage linguistic linked data hub, in: E. Blomqvist, P. Ciancarini, F. Poggi, F. Vitali (Eds.), Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2016), Springer International Publishing, 2016, pp. 239–254. doi:10.1007/978-3-319-49004-5_16.
- [86] D. Casey, M. Burke, T. Kovacic, K. Cortis, K. Murphy, A. Dolan, G. D'Onofrio, D. Sancarlo, F. Ricciardi, A. Teare, M. Raciti, R. Alonso, D8.3 Evidence of Service Robots Benefits (2018).
- [87] Y.-C. Lin, Y.-T. Dai, S.-L. Hwang, The effect of reminiscence on the elderly population: A systematic review, Public Health Nursing 20 (4) (2003) 297–306. doi:10.1046/j.1525-1446.2003.20407.x.
 - [88] M. A. Babar, L. Zhu, D. R. Jeffery, A framework for classifying and comparing software architecture evaluation methods, in: Proceedings of the 15th Australian Software Engineering Conference (ASWEC 2004), 2004, pp. 309–319. doi:10.1109/ASWEC.2004.1290484.
 - [89] M. A. Babar, I. Gorton, Comparison of scenario-based software architecture evaluation methods, in: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC 2004), 2004, pp. 600–607. doi:10.1109/APSEC.2004.38.
 - [90] L. Dobrica, E. Niemelä, A survey on software architecture analysis methods, IEEE Transactions on Software Engineering 28 (7) (2002) 638–653. doi:10.1109/TSE.2002.1019479.
- [91] R. Kazman, M. H. Klein, M. Barbacci, T. A. Longstaff, H. F. Lipson,
 S. J. Carrière, The architecture tradeoff analysis method, in: Proceedings of the 4th International Conference on Engineering of Complex Computer Systems (ICECCS '98), 1998, pp. 68–78. doi:10.1109/ICECCS.1998. 706657.

1465

- [92] L. J. Bass, P. C. Clements, R. Kazman, Software architecture in practice, Addison-Wesley-Longman, 1999.
- [93] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, H. S. Koppula, RoboBrain: Large-Scale Knowledge Engine for Robots, CoRRs (2014). URL http://arxiv.org/abs/1412.0691
- [94] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, M. Beetz, Oro, a knowledge management platform for cognitive architectures in robotics, in: Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2010), IEEE, 2010, pp. 3548–3553. doi:10.1109/IROS. 2010.5649547.
- [95] M. F. Story, J. L. Mueller, R. L. Mace, The universal design file: Designing for people of all ages and abilities, ERIC, 1998.
- [96] D. Saplacan, J. Herstad, T. Schulz, Situated abilities within universal design-a theoretical exploration, International Journal on Advances in Intelligent Systems Volume 13, Number 3 & 4, 2020.
- [97] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
- [98] J. O. Kephart, D. M. Chess, The vision of autonomic computing, IEEE Computer 36 (1) (2003) 41–50. doi:10.1109/MC.2003.1160055.
- [99] A. Zacharaki, I. Kostavelis, A. Gasteratos, I. Dokas, Safety bounds in human robot interaction: A survey, Safety Science 127 (2020) 1–19. doi: 10.1016/j.ssci.2020.104667.

[100] E. Bastianelli, M. d'Aquin, D. Nardi (Eds.), Proceedings of the 1st International Workshop on Application of Semantic Web technologies in Robotics co-located with 14th ESWC (ANSWER 2017), 2017.

1485

1490

1495

Appendix A.

Scenario	Perceiving	Interacting	Learning	Extensibility	Customizability	Explainability	Interoperability	Reusability
S.LinguisticData		1					1	1
S.POI	1	1	1					
S.Help	1	1						
S.TakeMeTo	1	1	1	1				
S.Recognize	1	1						
S.Questions	1	1	1	1				
S.Games		1		1	1		1	1
S.Music		1	1		1	1		
S.Entertain		1	1		1	1		
S.PersonalData		1			1			
S.Activities		1			1			
S.Reminiscence		1	1		1	1	1	1
S.Reminders		1			1			
S.Knowledge		1					1	1
S.Assessment	1	1			1			
S.Prompting		1				1	1	
Total	6	16	6	3	8	4	5	4

Table A.1: The association of the scenarios with the architectural drivers.

Scenario	C:Behavior	C:BehaviorScheduler	C:BehaviorManager	C:Text2Speech	C:Speech2Text	C:PerceptionAndMotionController	C:HCIManager	C:ConvolutedCapabilityModule	C:ConvolutedCapabilityManager	C:KnowledgeBase	C:ObjectOntologyMappingManager	C:ObjectOntologyMappingModule	C:SemanticBus	C:HotDeployManager
S.LinguisticData	1	1	1	1	1			1	1	1	1	1	1	1
S.POI	1	1		1	1	1	1	1	1	1		1	1	
S.Help	1	1		1	1	1							1	
S.TakeMeTo	1	1		1	1	1		1		1		1	1	
S.Recognize	1	1		1	1								1	
S.Questions	1	1		1	1			1		1		1	1	
S.Games	1	1	1	1	1		1			1	1	1	1	1
S.Music	1	1		1	1		1			1		1	1	
S.Entertain	1	1		1	1			1		1		1	1	
S.PersonalData	1	1		1	1					1		1	1	
S.Activities	1	1		1	1		1			1		1	1	
S.Reminders	1	1		1	1		1			1		1	1	
S.Reminiscence	1	1		1	1		1	1		1		1	1	
S.Knowledge	1	1		1	1					1		1	1	
S.Assessment	1	1		1	1	1	1	1		1		1	1	
S.Prompting	1	1		1	1			1		1		1	1	
Total	16	16	2	16	16	4	7	8	2	14	2	14	16	2

Table A.2: Analysis of the architectural components with respect to the scenarios.

Architecture	Perceiving	Interacting	Learning	Extensibility	Customizability	Explainability	Interoperability	Reusability
Alves-Oliveira2020	1	1		1	1			1
Bonaccorsi2016	1	1			1		1	
Breazeal2004	1	1	1	1				
Cao2016	1	1	1		1			1
Cao2019	1	1			1			1
Casas2018	1	1						
Cocsar2020	1	1						1
Coronado2021	1	1					1	
Fan2017	1	1			1			
Dehkordi2015	1	1						
Fasola2013	1	1			1			
Gonzalez-Pacheco2011	1	1		1				
Gonzalez-Santamarta2020	1	1						
Gross2011	1	1	1	1	1			1
Gross2012	1	1	1	1	1			1
Hirth2011	1	1						
Jayawardena2016	1	1		1	1			1
Kim2009	1	1	1					
Louie2014	1	1	1		1			
Louie2020	1	1	1	1	1			
Mead2010	1	1	1	1	1			
Portugal2019	1	1		1	1			1
Salichs2020	1	1			1			
Sarabia2011	1	1	1	1	1		1	1
Shi2018	1	1						
Torta2014	1	1						
Uluer2015	1	1	1					
Wood2019	1	1		1				
Zibafar2019	1	1						
Total	29	29	10	12	15	0	3	9

Table A.3: Analysis of the surveyed architecture with respect to the architectural drivers.