technologies

Article

ParlAmI: A Multimodal Approach for Programming Intelligent Environments †

Evropi Stefanidi, Michalis Foukarakis, Dimitrios Arampatzis, Maria Korozi *, Asterios Leonidis * and Margherita Antona

Institute of Computer Science, Foundation for Research and Technology—Hellas (FORTH), Heraklion, 70013 Crete, Greece; evropi@ics.forth.gr (E.S.); foukas@ics.forth.gr (M.F.); arabatzis@ics.forth.gr (D.A.); antona@ics.forth.gr (M.A.)

* Correspondence: korozi@ics.forth.gr (M.K.); leonidis@ics.forth.gr (A.L.)

† This paper is an extended version of our paper published in Proceedings of the 11th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2018), Corfu, Greece, 26–29 June 2018; pp. 50–57.

Received: 10 December 2018; Accepted: 7 January 2019; Published: 11 January 2019

Abstract: The proliferation of Internet of Things (IoT) devices and services and their integration in intelligent environments creates the need for a simple yet effective way of controlling and communicating with them. Towards such a direction, this work presents ParlAmI, a conversational framework featuring a multimodal chatbot that permits users to create simple "if-then" rules to define the behavior of an intelligent environment. ParlAmI delivers a disembodied conversational agent in the form of a messaging application named MAI, and an embodied conversational agent named nAoMI employing the programmable humanoid robot NAO. This paper describes the requirements and architecture of ParlAmI, the infrastructure of the "Intelligent Home" in which ParlAmI is deployed, the characteristics and functionality of both MAI and nAoMI, and finally presents the findings of a user experience evaluation that was conducted with the participation of sixteen users.

Keywords: conversational interface; chatbot; intelligent environment; smart home; NAO; embodied conversational agents; disembodied conversational agents; end-user programming; natural language processing

1. Introduction

The emergence of the Ambient Intelligence (AmI) [1] paradigm and the proliferation of Internet of Things (IoT) [2] have raised the need for appropriate tools that enable users to connect and manage the devices and services that are integrated inside "Intelligent Environments". In the case of domestic life, the advancement of IoT in combination with cloud computing [3] has led to an abundance of web-enabled devices and services for smart homes [4]. In such environments, individual users in their everyday lives interact frequently with various smart artifacts (e.g., smart lights, smart TV) in conjunction. For this reason, smart homes have become the most popular testbed for creating automated tasks based on the preferences, needs, and daily routines of their inhabitants. Developers already have means to manipulate the intelligent facility of a smart home and create useful integrations (e.g., "if nobody is at home, turn off all unnecessary electronic devices" or "save to Dropbox all attachments from incoming emails").

However, it is necessary to introduce a new type of programming environment that allows non-technical users to specify their own logic and create their own configurations in a simple yet effective manner. Providing end-users with tools suitable for programming their environment is not new [5–9]. Particularly, according to [10], one of the most straightforward approaches is the trigger-action paradigm. Trigger-action programming (TAP) has been under the spotlight from both the academic community and the industry [10–16]. Through such programming environments, users can define triggers (e.g., "if temperature falls below 18 °C") that initiate specific actions when their conditions are met (e.g., "turn on the heating"), thus automating common tasks. The majority of the tools created for enabling end-users to program an AmI environment feature graphical user interfaces (GUIs) that permit—mainly inexperienced—users to set their preferences visually rather than typing them in code since the latter is by far a more tedious task (especially for non-programmers).

Nonetheless, thanks to the recent advances in spoken language technology and artificial intelligence, it is possible to use natural language to control smart devices. This approach is supported by [17], which outlines the benefits of using natural language programming when it comes to novices, as it alleviates challenges to conduct software development, scripting, and verification. As an example of spoken language technologies enabling users to control smart devices, through Amazon's Alexa [18], the users can add a calendar event, set an alarm, or check the weather. Apart from giving simple one-off commands to intelligent artifacts, the ability to use natural language for creating the rules that dictate the environment's behavior also needs to be investigated. Towards that direction, conversational interfaces (CIs) [19] and, more specifically, chatbots, appear as promising tools. Chatbots are artificial intelligence (AI) algorithms that process natural language, analyze it and provide an intelligent response [20]. That way, they permit users to interact with intelligent artifacts using spoken language in a natural way, like engaging in a conversation with a person. In the recent past, chatbots have been utilized as the intelligent mechanism behind disembodied conversational agents (DCAs) and embodied conversational agents (ECAs). ECAs have a (virtual) body or face, usually human-like [21], per the general assumption that people attribute more intelligence and trust to human-like ECAs [22], while DCAs are usually found in the form of messaging applications [23].

This work presents ParlAmI [24], a conversational framework featuring a multimodal chatbot that permits users to create "if-then" rules so as to dictate the behavior of an intelligent environment. It delivers a disembodied conversational agent in the form of a messaging application that can be launched on the inhabitants' personal devices and employs the programmable humanoid robot NAO [25] in order to realize an embodied conversational agent. ParlAmI aims to assist inexperienced users in programming their environments in a simple yet effective manner while providing the necessary flexibility to define complex behaviors. Towards this end, it interoperates with the LECTOR framework [26] so as to specify which conditions signify a behavior that requires some action (i.e., triggers), and determine the appropriate interventions to accommodate/help/support the users (i.e., actions).

The following sections present (i) a discussion of related work, (ii) the infrastructure of the Intelligent Home where ParlAmI is deployed, (iii) a description of the ParlAmI system, (iv) its two faces, i.e., the two conversation agents MAI and nAoMI, (v) the results of a user-based evaluation that was conducted, and (vi) a discussion regarding its findings and future plans.

2. Related Work

2.1. End-User Programming

Over the next few years, the goal of interactive systems and services will evolve from just making systems easy to use to making systems that are easy to be extended by end-users [27]. More precisely, a system should offer a range of different modification levels with increasing complexity and power of expression; this trade-off, called "gentle slope", dictates that users will be able to make small changes simply, while more complicated ones will involve a proportional increase in complexity [28–30]. Modular approaches generally provide a gentle slope for a range of complexity by allowing successive decomposition and reconfiguration of software entities consisting of smaller components [31]. Moreover, systems aiming to support end-user development (EUD), apart from being considerably more flexible, must satisfy various non-technical aspects that influence the demanding task of EUD, i.e., they must be easy to understand, to learn, to use, and to teach [27].

EUD has been successfully applied in various domains, including commercial software where users are permitted to incorporate their own functionality (e.g., recording macros in word processors, setting up spreadsheets for calculations, and defining e-mail filters). As aforementioned, the trigger-action paradigm is a straightforward approach that enables users to configure the behavior of a system by specifying triggers and their consequent actions (e.g., "if a new e-mail with an attachment has been received, then save the attached on Dropbox") [11]. The wide acceptance of these systems by users stems from their simple wizard-style interface, with IFTTT (if-this-then-that) [15] and Zapier [16] being the most popular TAP ecosystems that allow users to create their "own" programs. Furthermore, their support for wearables, smartphone sensors, and other commercial devices tightly couple such approaches with ubiquitous computing. Apart from IFTTT-style tools that offer GUIs to construct user-defined programs, semantic parsers [32] are also employed to allow users to describe recipes in natural language and automatically map them to executable code. Their approach utilizes a probabilistic log-linear text classifier (i.e., machine learning) to map natural language to a formal program representation.

EUD tools are not only used to integrate distinct online services, but also towards the development of web applications. In [14], a set of tools is presented that allow end-users to create and configure context-aware web applications through an intuitive interface by creating trigger-actions rules. Bubble [33] is a tool and service that allows end users to create rich web applications by drag-and-drop style manipulation of components and setting the value of different parameters, claiming that "you don't need to be a coder to build software". Another popular approach to EUD is visual programming, which allows end-users to program without writing actual code. Visual programming has been explored through programming languages such as Scratch [34], Alice [35], Snap! [36], and an extension of it called NetsBlox [37], which all aim to facilitate programming by non-experts or even children, mainly in the domain of education.

Programming by demonstration (PBD) is another known technique whereby programming is conducted in the form of performing a sequence of steps. Some of the early efforts in this area were focused on user interface creation, such as Peridot [38] and Marquise [39], or KidSim, which allows children to create simulations [40]. As more recent examples, Koala [41] and CoScripter [42] allow end-users to create scripts in order to automate web-based processes (e.g., repeating online orders), while [43] presents an environment where end-users can program context-aware behaviors according to the PBD programming principle (e.g., log medicine intake to a medical journal).

Finally, EUD is not limited to web or desktop deployment and usage. The need to automate mobile phone processes led to an increase of mobile applications that allow users to build rules that fire when mobile related events are triggered. Tasker [44] and Automate [45] are two of the most popular mobile applications that offer such functionality.

2.2. End-User Programming Using Natural Language

Although the state-of-the-art greatly differs between the seventies and now, the subject of natural language programming was already being discussed back then. Arguments about whether it is possible to program in natural language date at least as far back as 1966 [46]. For example, in 1976, [47] presented four research projects regarding systems that could carry on natural language dialogues with non-programmers about their requirements and then produce appropriate programs, such as a natural language automatic programming system for business applications by MIT, and a system allowing users to create business application programs by holding an informal, interactive dialogue with the computer by IBM. Shortly after, [48] presented an attempt to obtain detailed empirical information about the nature of natural language programming, which led them to the identification of three major obstacles: style, semantics, and world knowledge. As another example, NLC is a natural language programming system [49–51] that allows users to display and manipulate tables or matrices while sitting at a display terminal.

Almost four decades later, the subject has increasingly become one of interest, especially due to the technological advancements in spoken language technology and artificial intelligence, which have now made it possible to program an AmI environment using natural language. Natural language processing (NLP) is still not perfect, but it is continuously improving so that various NLP algorithms can be used to recognize and process the voice command given by user [52].

A straightforward example of using natural language to control and program is the use of Amazon's Alexa [18], through which it is possible to control smart devices by voice, such as setting a notification for an event. In the area of robotics, [53] introduces a method that uses natural language to program robotized assembly tasks by using a generic semantic parser to produce predicate-argument structures from the input sentences, while in [54], the design of a practical system that uses natural language to teach a vision-based robot how to navigate in a miniature town is proposed, enabling unconstrained speech by providing the robot with a set of primitive procedures derived from a corpus of route instructions. The idea of empowering end-users to control their surroundings also extends to other dimensions, such as Ambient Assisted Living, as is presented in [55], which introduces natural-language enabled wheelchairs and their capabilities, such as wheelchairs limited to metric commands (can only follow verbal commands to go forward, turn, or stop), wheelchairs capable of following local features (can navigate relative to local environmental features such as walls and elevators), and wheelchairs capable of navigating to specified locations.

Regarding natural programming systems, [56] separates them into two categories: those automatically converting natural language input into code written in an existing programming language, such as Metafor [57,58] and MOOIDE [59], and those which are themselves executable, of which Inform 7 is probably the most well-known example [60,61]. The work in [56], whose main contribution is Flip, a bi-modal programming language designed for use by young people in the context of game creation, uses the latter as the basis for a study into the feasibility of natural language for coding. They discuss problems they encountered due to the lack of syntactic variations of a semantic meaning in programming languages, where, for example, a synonym of a keyword means absolutely nothing, e.g., if a programming language utilizes the keyword "activate" and the word "turn on" is used in natural language for the same purpose, the programming language is normally not equipped to make the mapping between the two synonyms—it will simply not understand what the user is trying to say.

In the domain of conversational agents (CA), recent technological advances have made it possible to use the voice to perform various tasks on a device in a quicker and more efficient manner with respect to typing or tapping [62]. As an example of a CA, IntructableCrowd is a system that utilizes crowdsourcing and is implemented as a CA for its end-users, which allows the user to instruct the crowd—the so-called crowd workers—via conversation and ask them to create trigger-action/if-then rules that meet their needs [63].

2.3. Physical Conversational Agents and Intelligent Environments

There are a number of studies that highlight the positive effects of having a physical presence compared to a virtual one. One early use of an embodied agent was Gandalf [64], a humanoid software agent with some ability to perceive multimodal acts such as speech, prosody, body language, and gaze. Later studies [65,66] concluded that users believe the interaction with an agent capable of natural conversational behaviors (e.g., gesture, gaze, turn-taking) is smoother than an interaction with these features disabled. The advantage of embodied agents is further strengthened by several works, one of which is [67]. The authors claim that utilizing an embodied agent with locomotion and gestures can have positive effects on the users' sense of engagement, social richness, and social presence with the agent. In particular, research in virtual reality (VR) and robotics has shown that agents that are aware of the user elicit more social presence and better performance than those that are not. This is also supported by the work in [68,69], which demonstrated that embodied agents who maintained a mutual gaze with a user caused the user to respect the agent's personal space and also obtained higher self-report measures of influence. Moreover, with respect to engagement, the participants in [70] could better recall the story of a robotic agent when the agent looked at them more during a storytelling scenario.

Additionally, several studies support the additional benefits of physical presence in conversational agents. In [71], the difference between human operated co-present, tele-present, and simulated robots solving the Towers of Hanoi puzzle was investigated. The users favored the physical presence of the robot in this case. In another experiment where people actually interacted with the agent using dialogue rather than operating it [72], it was found that users were more expressive in their interaction with a physical entity rather than a virtual agent. An interesting observation in this research is that the robot used was iCat, an interactive cat-like robot, while the screen agent was humanoid, meaning that even though humanoid agents are considered to be more accepted, the physical presence of the conversational agent seemed to be more important. To support these results, the iCat robot was also used in [73] and was compared to a virtual agent of iCat, a 3D representation of the robot on a computer screen, to measure the enjoyment of chess players versus physically or virtually embodied agents. The performance of the physical version of a chess opponent scored significantly higher with respect to feedback, immersion, and social interaction. In an attempt to interpret the impact of physical presence of robots versus tele-presence and virtual agents, [74] includes an extensive survey of experimental works including the above. The results support the hypothesis that physical presence plays a greater role in psychological responses to an agent than physical embodiment and virtual presence. In summary, co-present, physical robots received more attention, were better accepted, and generally performed better in most experiments.

2.4. Programming Intelligent Environments

Intelligent environments are physical spaces in which information and communication technologies and other ambient facilities are interwoven and used to achieve specific goals for the user, the environment, or both [75]. Their proliferation has resulted in the need for appropriate tools that allow non-technical users to program their behavior by specifying their own logic and creating their own automations in a simple yet effective manner. The authors in [76] present a rule-based agent mechanism of a ubiquitous end-user programming system that allows users to program smart spaces in a uniform way. Moreover, research has shown that inexperienced users can quickly learn to create programs containing multiple triggers or actions when a trigger-action programming paradigm is applied [11]. Finally, CAMP [9] is another example of an end-user programming environment for creating context-aware applications that automatically capture details of a live experience and provide future access to that experience for the smart home based on a magnetic poetry metaphor, which allows users to create applications in a way that takes advantage of the flexibility of natural language.

Regarding the programming of the behavior of objects, [77] introduces the Reality Editor, which allows for the programming and operation of smarter objects by authoring basic connections between smart objects using augmented reality (AR) on handheld devices. The idea of modifying the interface and behavior of physical objects—as well as their interactions with other smart objects—is also presented in [78]. Built on these ideas is Ivy [79], a spatially situated visual programming environment for authoring logical connections between smart objects. Another approach for authoring object behaviors is to incorporate tools into virtual environments, allowing users to more readily understand the behavior of their scene and the element [80–82]. In [83], a model that includes new operators for defining rules combining multiple events and conditions exposed by smart objects is presented. In the domain of domestic life particularly, the proliferation of web-enabled devices and services for smart homes [4] has led to the emergence of various programming environments that enable non-technical users to create complex home automation scenarios with heterogeneous devices [8,84–88]. Finally, in [89], the authors stress the importance of the user being in control, i.e., having the option to define or modify a system's behavior, and they investigated how end-user development can empower users to define or modify the behavior on an intelligent system or environment beyond simple parameter specification. Among others, their findings highlight the need for different programming metaphors that will enable end users to modify the behavioral rules of context-aware devices.

3. The Infrastructure of the "Intelligent Home"

3.1. AmI-Solertis

The AmI-Solertis framework [90] empowers users to create behavior scenarios in AmI by delivering a scripting mechanism that can dynamically adapt the execution flow and govern the "high-level business logic" (i.e., behavior) of the intelligent environment, as well as a complete suite of tools allowing management, programming, testing, and monitoring of all the individual artifacts (i.e., services, hardware modules, software components). AmI-Solertis enables the creation of programs called AmI scripts, which constitute software agents who dictate how the technologically-enhanced environment will react in contextual stimuli (e.g., physical aspects, users, current activities) [91].

Reflecting on the benefits of asynchronous and event-based communication and embracing the software-as-a-service cloud computing design pattern, AmI-Solertis introduces a unified hybrid communication protocol that combines the widely-used Representation State Transfer (REST) protocol with asynchronous and event-based communication facilities to integrate heterogeneous services in a standardized yet agnostic manner (Figure 1). To that end, any AmI artifact (an AmI artifact refers to an intelligent service or application that exposes part or all of its functionality in the AmI environment for further use) or AmI script is expected to expose a REST interface to receive incoming calls and communicate its intention to the AmI ecosystem by emitting appropriate events. Since REST does not accommodate a standardized event mechanism by design, AmI-Solertis, following the widely used and well-established practice of intermediary message brokers [92,93], introduces a custom universal event federator server to enable asynchronous communication among the various artifacts. To further enhance interoperability and discoverability, the OpenAPI Specification (OAS) [94] is used to formally describe every Application Programming Interface (API), whereas based on that formal service definition, AmI-Solertis automatically generates service-specific wrappers (i.e., proxies) that expose the underlying remote API as local functions by encapsulating the communication-related boilerplate code in order to streamline remote service usage.

Upon successful integration, an AmI artifact or script is stored in the universal AmI artifacts and scripts repository, which copes with the overall service management and enforces the common intercommunication scheme. Among others, this module incorporates the storage mechanisms that hold the metadata and executable files, the service discovery tools, and the installation wizard that installs new services. This component also offers a subset of its functionality through a public REST API that enables third party tools (e.g., LECTOR, ParlAmI) to query the registry for extended descriptions of the integrated service and facilitate authenticated users or systems to administer the registry by executing CRUD (Create, Read, Update and Delete) operations (i.e., insert a new or delete an existing service).

Moreover, since Ambient Intelligence (AmI) environments incorporate a variety of heterogeneous devices with different capabilities and fluctuating availability, and AmI-Solertis aims to be deployed into and manage such intelligent environments with many different stationary and mobile devices, it consolidates well-established practices and approaches that automate the packaging process, simplify distribution and deployment, and facilitate real-time management and operation monitoring of compliant artifacts or AmI scripts. In particular, AmI-Solertis generates and packs together all the necessary resources (e.g., executable files, configuration files, OS directives) that should be transferred to the host (via the deployment process) in order to be present and ready to be used when a relevant execution command is given. Finally, since AmI-Solertis targets highly mutable and dynamic environments that change quite rapidly, any references to its components (e.g., AmI artifacts and scripts) should minimize (or even eliminate) the amount of static information they require to discover and communicate with each other; on the contrary, the ecosystem should offer a dynamic and flexible naming system that promotes ubiquitous and distributed access to any requested resource or service [95]. To that end, AmI-Solertis employs an enhanced Domain Name System (DNS) tool named Components Directory Service that associates various information with the domain names assigned to each of the participating entities and serves as an advanced "phone book" for the ecosystem.

AmI-Solertis has been already used by the LECTOR framework (Section 3.2) so as to integrate AmI artifacts and AmI scripts in its reasoning process and publish new executable components that detect a variety of behaviors at runtime and apply context-aware interventions when necessary.

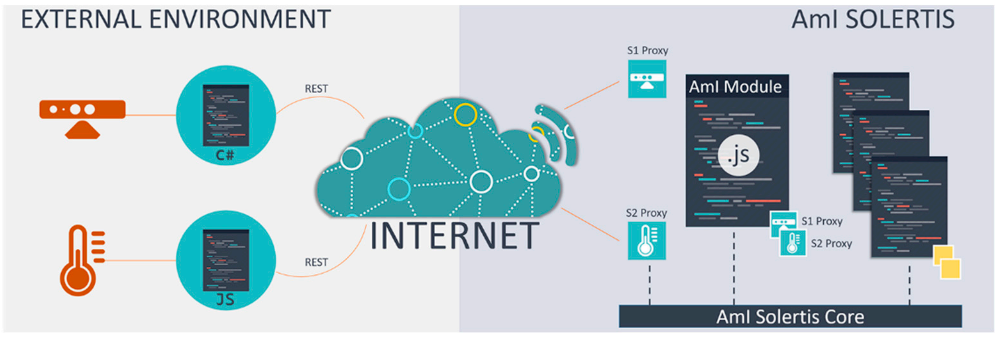


Figure 1. The AmI-Solertis Hybrid Communication Protocol.

3.2. LECTOR

LECTOR [26] exploits the potential of AmI technologies to observe either human- or artifact-oriented behaviors (SENSE), identify whether they require actions (THINK), and intervene (ACT) accordingly—when deemed necessary—in order to fulfill the user needs. According to cognitive psychology, the sense-think-act cycle stems from the processing nature of human beings, who receive input from the environment (i.e., perception), process that information (i.e., thinking), and act upon the decision reached (i.e., behavior) [96]. This identified pattern constitutes the basis for many design principles regarding autonomous agents and traditional AI [97].

Fundamentally, LECTOR's mechanisms entail two basic steps: (i) identifying a user need and, (ii) acting accordingly. This simplification reveals similarities with the trigger-action model [10,11], which in recent years has been in the spotlight as a form of programming intelligent environments using simple "if trigger(s), then action(s)" rules. Indeed, providing the right type of help or support (i.e., action) as soon as the user needs it (i.e., trigger) is imperative in many application domains such as Ambient Assisted Living (AAL), eHealth, domestic life, learning and education, etc.

Particularly with respect to domestic environments, many solutions have been introduced towards programming the behavior of smart homes, enabling people without programming experience to create their own configurations. In these approaches, however, triggers are simple sensor readings, and actions are nothing more than mere automations. On the contrary, LECTOR embraces the concept of behavior, which seems more appropriate for describing triggers that originate from or revolve around human activities. Additionally, taking into consideration the user-oriented nature of AmI, the term intervention was introduced to describe the actions that the environment undertakes to support its users.

Since the decision-making mechanisms of LECTOR rely on rule-based conditions, it supports the creation of three (3) types of rules:

- Rules that "model" a behavior[5] based on physical context.[3]
- Rules that "model" the triggers[6] based on the behavior[5] of actors[1] under domain specific context.[4]
- Rules that describe the conditions (triggers[6] and domain specific context[4]) under which an intervention[7] is initiated on a specific intervention host.[2]

This decomposition increases the number of steps that a user must complete in order to connect a trigger to an intervention compared to simply generating an if-then clause. Nevertheless, it improves scalability, promotes reusability of triggers and actions, and enhances overall rule management.

In particular, the three ingredients (i.e., behavior, trigger, intervention) are defined in isolation and are only connected in terms of public "connection points". Therefore, any ingredient can be modified independently of the others and as long as its connection points remain the same, no more adjustments will be required for the system to continue to operate as prior to the change. This approach not only minimizes unwanted ramifications, but also facilitates collaboration, as new rules can be easily created by different users. This is inspired by how an application programming interface (API) simplifies programming and enables computer programs to evolve independently by abstracting the underlying implementation and only exposing objects the developer needs. The core concepts of this rule-based approach are explained below:

1. Actors are the (groups of) users or artifacts of the intelligent environment whose behavior needs to be monitored in order to decide whether an intervention is required.
2. Intervention hosts can either launch an application with specific content or control the physical environment. They are: (i) common computing devices such as smartphones, tablets, and laptops, (ii) technologically augmented everyday physical objects (e.g., augmented coffee table), or (iii) custom made items (e.g., smart sofa).
3. The physical context encapsulates information regarding physically observable phenomena via sensors (e.g., luminance, heart rate, sound levels, etc.).
4. The domain specific context refers to any static or dynamic information that is provided through software components (e.g., user profile or agenda).
5. Behavior is a model that represents the actions of an actor (e.g., a user is watching TV, cooking, sleeping).
6. Trigger is the model of a high-level behavior that can initiate an intervention.
7. Interventions are the system-guided actions that aim to help or support users during their daily activities.

3.3. The Intelligent Home

ParlAmI is currently deployed inside the "Intelligent Home" simulation space located at the AmI Facility Building (http://ami.ics.forth.gr/) within the FORTH-ICS campus. Inside this environment, everyday user activities are enhanced with the use of innovative interaction techniques, artificial intelligence, ambient applications, sophisticated middleware, monitoring and decision-making mechanisms, and distributed micro-services. Regarding the programmable hardware facilities of the "Intelligent Home", they currently include both commercial equipment and technologically augmented custom-made artifacts:

- Common domestic equipment such as the wide variety of commercial devices (e.g., Philips Hue Lights [98], blinds [99] Alexa [18], oil diffusers [100], and appliances (fridge, oven, coffee machine, air-conditioner) that can be controlled either via their own API or by using dedicated solutions to that matter (e.g., KNX bridge [101]).
- Technologically augmented objects of the living room (Figure 2):
    - AmITV constitutes the main display of the "Intelligent Living Room". It hosts a number of interactive applications that mostly aim to entertain and inform the user (e.g., ambient movie player, news reader, online radio, music player)
    - AugmenTable is a stylish commercial coffee table made of wood with a smooth, non-reflective white finish that, in addition to its intended use for placing objects (e.g., cups, plates, books) on top of it, acts as a large projection area where secondary information can be presented. Through a Kinect sensor [102], AugmenTable becomes a touch-enabled surface that can recognize the physical objects placed on it as well.
    - SurroundWall transforms the wall around the TV into a secondary non-interactive display that provides an enhanced viewing experience by augmenting—in a context-sensitive manner—the content presented on the AmITV artifact.

- SmartSofa is a commercial sofa equipped with various sensors aiming to detect user presence inside the room and provide information regarding the user's posture while seated.

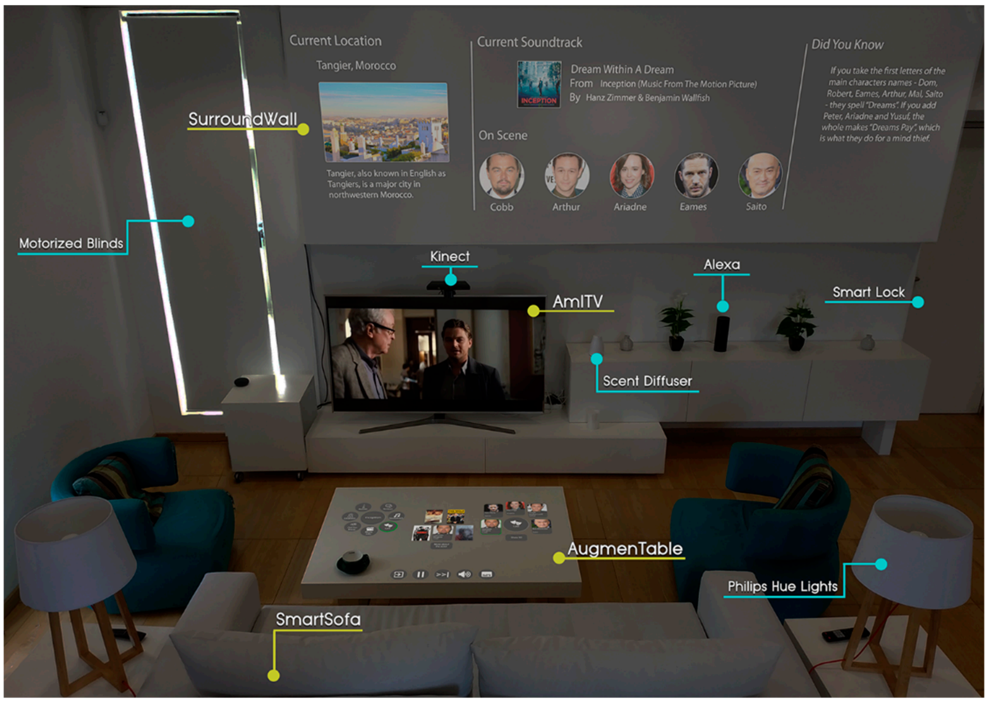


Figure 2. The "Intelligent Living Room" setup.

- Technologically augmented objects of the bedroom:
    - The Hypnos framework aims to improve quality of sleep by providing sleep hygiene recommendations. To do so, it relies on the SleepCompanion (i.e., a bed that integrates various sensors like Withings Sleep Tracking Mat [103], Emfit QS Sleep Tracker [104]), and the inhabitants' wearables (e.g., Nokia Steel HR [105], Fitbit Charge 3 [106]) to monitor their physical activity (e.g., movement, time in bed), physiological signals (e.g., respiration rate, heart rate, snoring) and various sleep-related parameters (e.g., time to fall asleep, time asleep, time awake, sleep cycles) while resting.
    - SmartWardrobe is a custom-made closet equipped with various sensors (e.g., Radio Frequency Identification (RFID) readers, Passive Infrared (PIR) motion sensors, magnetic contacts) and a tablet that aims to provide outfit recommendations based on real-time weather prediction and the user's daily schedule.
    - SmartDresser is a commercial dresser equipped with various sensors (e.g., RFID readers, load sensors) that is able to identify the location of the stored clothing items in order to help users find what there are looking for faster.
    - A-mIrror is an intelligent mirror comprised of a vertical screen and a Kinect sensor that identifies the inhabitants via facial recognition and provides clothing advice and make up tips.
- Technologically augmented objects of the kitchen:
    - AmICounterTop is a regular work surface that has been augmented with a (hidden) projector and a Kinect sensor that transforms it into an interactive display. By adding load sensors, it can double as a digital kitchen scale (which is useful while cooking).

    - AmIBacksplash is a system adjacent to the countertop. The walls around the kitchen act as projection surfaces. Their position makes them ideal as secondary (non-interactive) displays to AmICounterTop, but they can also be used as primary displays if the user chooses so.

- Technologically augmented objects of the main entrance:
    - SmartDoorBell is a system installed outside the main entrance of the house. The door is equipped with a smart lock [107], and a custom-made wooden case embedded in the wall houses a tablet device, a web-cam, an RFID reader, and a fingerprint reader. Through appropriate software, the SmartDoorBell delivers a sophisticated access control system for the residents and visitors of the home.

4. The ParlAmI System

Based on the concept of allowing end-users to define the behavior of an intelligent environment via natural language, we present ParlAmI. It introduces a hybrid approach that combines natural language understanding (NLU) with semantic reasoning and service-oriented engineering so as to deliver a multimodal conversational interface that assists its users in determining the behavior of AmI environments. Particularly, it offers an alternative easy-to-use approach (especially for novice users with little to no programming experience) towards generating such rules through conversing in natural language with a context-aware intelligent agent (i.e., chatbot). This chatbot comes in two forms, i.e., a disembodied conversational agent in the form of a messaging application (MAI), and an embodied conversational agent (nAoMI) by employing the programmable humanoid robot NAO.

4.1. Requirements

ParlAmI is a conversational framework featuring a multimodal chatbot that interoperates with LECTOR (Section 3.2) in order to empower non-technical users to define the rules that guide LECTOR's decision-making mechanisms. This section presents the high-level functional requirements that ParlAmI satisfies, which have been collected through an extensive literature review and an iterative requirements elicitation process based on multiple collection methods such as brainstorming, focus groups, observation, and scenario building.

FR-1. Be aware of contextual information. LECTOR's decision-making mechanisms heavily depend on contextual information to identify the actual conditions that prevail in the intelligent environment at any given time and act accordingly. Contextual information refers to domain specific context (e.g., inhabitants' schedule, family dietary plan) and physical context (e.g., luminance, heart rate, sound levels), which are both invaluable to ParlAmI. That way, the system is able to put the user sentences in context and interpret them correctly in order to create the desired rule(s), while at the same time being able to disambiguate possible misunderstandings through conversation.

FR-2. Be aware of the potential actors. According to LECTOR, the term actor is used to describe the subjects of the intelligent environment (e.g., users, artifacts) whose behavior needs to be monitored in order to decide whether an intervention (ACTION) is required. In the context of the "Intelligent Home", the potential actors are the inhabitants, their guests, and the intelligent artifacts of the house. Generally, different actors have diverse characteristics (e.g., children have different needs and preferences than adults), while they are also expected to display different behaviors (e.g., different routine for waking up in the morning). To this end, ParlAmI must be aware of such information when building a rule. Binding behaviors to individual actor(s) not only simplifies the rule creation process but mainly enables different courses of action to be taken for different actors. For example, when baking is still ongoing, the adult inhabitants that are present in the house can be notified to check the food as soon as they enter the kitchen (i.e., behavior), whereas such a process is not valid for children.

FR-3. Be aware of the available intervention types and intervention hosts. Interventions are, in fact, applications running on personal or public devices (intervention hosts) instantiated at a key point in time with appropriate content. ParlAmI needs to be aware of the different intervention types that are available at the "Intelligent Home" so as to be able to choose the most appropriate one according to the context of use. As an example of an intervention type, AmITV can utilize its built-in slideshow application to present a compilation initialized with specific pictures. Additionally, ParlAmI needs to be aware of the available intervention hosts, namely the artifacts that can either be used as application launchers (e.g., launch a video with specific content) or controllers for the physical environment (e.g., adjust the house temperature during the night). That way, the system will be able to choose the best one so as to deploy an intervention at any given point according to various criteria (e.g., user preferences, presence of different users, user location).

FR-4. Enable the definition of behaviors. The term behavior is used to describe the acts performed by users or artifacts (e.g., someone talks, a device switches on). In some cases, multiple cues (from diverse sources) are required in order to identify a behavior. To this end, it is important to allow the definition of a behavior by combining multiple information from the physical context (i.e., if physical context 1 and . . . physical context N, then behavior X).

FR-5. Enable the definition of triggers. As already mentioned, the term trigger is used to describe a high-level behavior that requires the initiation of an intervention. A behavior can potentially become a trigger under specific context (e.g., the behavior COOKING might initiate the trigger COOKING FINISHED if the cooking time has passed, or the trigger COOKING STOPPED if the oven is suddenly turned off). When defining the conditions under which a trigger is initiated, LECTOR supports the combination of multiple contextual information. Furthermore, LECTOR differentiates from other trigger-action programming systems that evolve around device- or software-initiated triggers. In more detail, the rule logic is de-coupled from the artifacts and is human-oriented instead. For example, the condition "if motion is detected in the hallway" becomes clearer to the simple user when expressed as, for example, "if I pass through the hallway". Additionally, the definition of a trigger does not depend merely on the behavior of a single actor; on the contrary, the combination of more than one actor behaviors is required so as to support the realization of more complex scenarios (e.g., if the father is sleeping and the mother is in the bathroom then CHILD TEMPORARILY UNSUPERVISED). To this end, ParlAmI should be able to model a trigger following the requirements set by LECTOR.

FR-6. Enable the definition of intervention rules. LECTOR permits the definition of rules that describe the conditions (triggers and virtual context) under which one or more intervention types are initiated on one or more presentation hosts (e.g., if the LIGHTS ARE ON and I am WATCHING A MOVIE (trigger) and IT IS DAYTIME (context), then I want THE BLINDS TO SHUT (intervention 1) and the LIGHTS TO DIM (intervention 2)). The combination of multiple intervention types when defining an intervention rule results in the creation of richer interventions. Furthermore, in order to support the realization of complex scenarios, LECTOR permits the creation of rules that combine multiple triggers with multiple interventions types. Hence, the above functionality should also be supported by ParlAmI.

4.2. Architecture Overview

ParlAmI consists of multiple services (Figure 3) that expose their functionality through a REST API:

- Messaging web app (called MAI) is a service that hosts and controls a web-based chatbot user interface intended for end-users. MAI forwards all user written messages to the ParlAmI bridge and then waits for a response. The response directs MAI on how to reply to the user, especially in situations where the user has to provide additional input. To that end, it chooses between custom rich UI elements that can be used for different types of input, including controls that allow users to pick an action from a list, select a specific location, determine the smart artifacts to be controlled, set specific parameters (e.g., brightness), or answer polar questions (e.g., YES/NO). For example, if the user writes "I would like to turn on the lights", the system will prompt her to choose which of the controllable lights/lamps currently available in the intelligent environment will be turned on.

- NAO manager (nAoMI) service is responsible for communicating with the bridge and for handling all robotic actions related to the current state of the conversation. NAO can be programmed using a Python software development kit (SDK) that gives high-level access to all robotic functions. In addition, there is a graphical editor for high-level robotic behaviors named Choregraphe [108] that allows the creation of simple scripts using functional boxes and connections between them. The speech recognition part of the conversation is implemented as a Choregraphe behavior while everything else is handled by a Python script called NAO manager. Whenever the robot is in listening mode, the speech recognition behavior is running. Since the system should be able to handle arbitrary input, it is not possible to use the built-in speech recognition that NAO offers, which can only listen to specific words or patterns. Instead, the robot records a voice clip that includes the user's utterance, as it can understand when the user starts and stops a sentence. This clip is sent to an online speech-to-text service that returns either the text form of the user utterance or an error if nothing was recognized. In the latter case, the robot will kindly ask the user to repeat the last sentence. Finally, the recognized text is sent back to the NAO manager. The NAO manager forwards the texts it receives from the speech recognition behavior to the ParlAmI bridge for further processing. When a response is available from the bridge, the manager receives an encoded message with instructions on what to say and additional robotic actions that may be required according to the nature of the response. For example, when the response contains an object or concept that the robot can recognize or understand, it can point to the object or perform a relevant gesture to improve the quality of the conversation and make it seem more natural (e.g., when saying that the ceiling lights will turn on, the robot will point up towards their direction, or when greeting the user, the robot will make appropriate waving gestures).
- ParlAmI bridge provides an API that allows different conversational agent interfaces to be powered by the ParlAmI framework. It acts as a gateway to the ParlAmI framework where different embodied (e.g., humanoids) or disembodied (e.g., messaging web apps) agents can authenticate with and start conversations that result in the creation of new rules in the intelligent environment. After a connection is established with the ParlAmI bridge, any message received is forwarded to spaCy to initiate the analysis pipeline of ParlAmI. Upon receiving a response from ParlAmI, it forwards the message back to the corresponding conversational agent alongside the metadata regarding the meaning of the message; that way, the interface (visual or not) can be customized accordingly (e.g., location data are interpreted by the NAO manager in order to point to the specific object that the answer is referring to).
- spaCy is a library for advanced natural language processing in Python and Cython [109]. It includes various pre-trained statistical models and word vectors and currently supports tokenization for 30+ languages. It features a syntactic parser, convolutional neural network models for tagging, parsing, and named entity recognition, and easy deep learning integration. ParlAmI supports complex messages that can be composed of multiple triggers and/or actions. To that end, spaCy is used in order to tokenize a sentence to a syntactic tree. After the decomposition of the complex sentence, the tree is traversed in order to artificially generate the verbose equivalents of the separated sentences. The new utterances are then forwarded to Rasa for further processing to extract their meaning.
- Rasa is responsible for extracting structured data from a given sentence (such as user intents, i.e., what task the user aims to accomplish via the current dialogue, like "turn on the light") and entities that have been successfully recognized and will be later translated into messages that will be forwarded to the services of the AmI-Solertis framework to execute the actual task (e.g., turn on the ceiling light and adjust its brightness to 65%). Rasa is being continuously trained with sentences provided by AmI-Solertis, which map directly to available services and their respective functions. To that end, service integrators are asked to supply indicative utterances while importing a new or updating an existing service via AmI-Solertis. The generated output is forwarded to the NLP core service for further processing.

- NLP core handles multiple key parts that enable ParlAmI to generate natural dialogues and to create rules affecting an actual intelligent environment. The service receives the data provided by Rasa and artificially aggregates them into a single sentence that can be interpreted by ChatScript. Alongside, as the dialog progresses, it collects the incoming data (e.g., intents, entities, parameters) that will compose the complete object to be forwarded to LECTOR as soon as all the pieces are recorded. For that to be achieved, it communicates with AmI-Solertis to retrieve, in a context-sensitive manner (i.e., based on the recognized intents/entities), information regarding the available actions (i.e., functions) and their parameters so as to provide appropriate hints to ChatScript on how to proceed with the conversation, store the data in a form that is recognizable by the Rule Builder Service, and invoke the appropriate services for building and deploying the aforementioned rule when ready. Moreover, the NLP core also handles user sessions, allowing concurrent conversations between multiple users and the ParlAmI.
- ChatScript is the endpoint for the conversation between the user and ParlAmI from the framework's stand point. ChatScript uses powerful pattern matching techniques that allow natural dialogues with end-users. Upon receiving a sentence from the NLP core, based on the context and the previous state of the conversation, it generates and reports back the appropriate answer.
- Rule Builder acts as a proxy from the ParlAmI environment to LECTOR. It transforms the extracted data from the conversation to a structured rule that can be interpreted by LECTOR. It then sends the data to LECTOR in order to create the appropriate actions, interventions, and triggers and finally generate the code that will then be deployed to the intelligent environment via AmI-Solertis.
- AmI-Solertis has two roles in ParlAmI. On the one hand, it feeds ParlAmI with rich information about the available programmable artifacts (i.e., service names, functions and their descriptions, parameters, expected values, example utterances) that can be used as training data. On the other hand, it is the backbone of LECTOR, providing an infrastructure that allows the deployment and execution of the created rule to the actual environment.
- LECTOR receives the rule that ParlAmI generates from a conversation with a user and transforms the appropriate triggers, interventions, and/or actions into AmI-Scripts. These scripts are finally forwarded to AmI-Solertis for deployment.

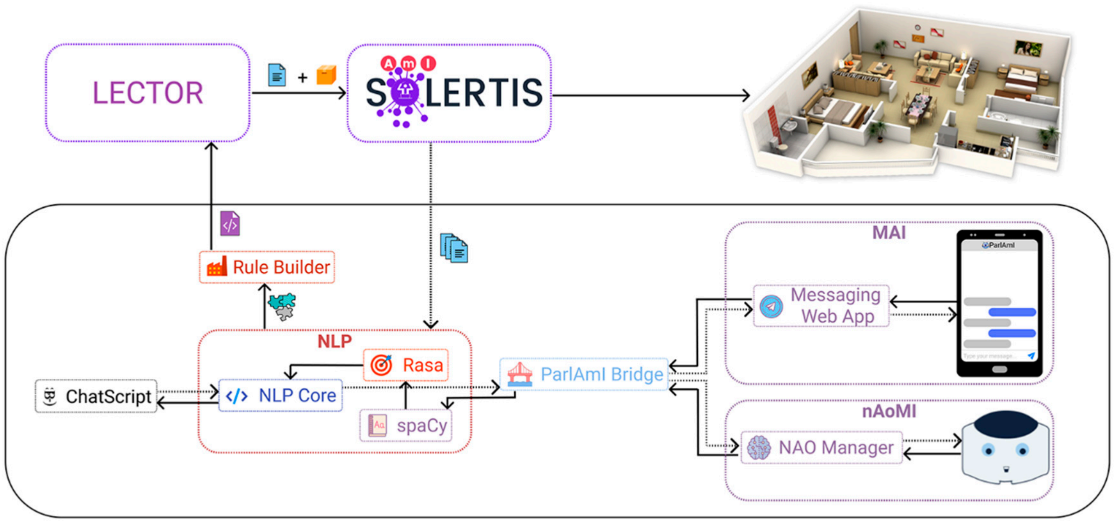


Figure 3. ParlAmI's architecture.

4.3. The Analysis Pipeline

ParlAmI adopts a multi-stage process so as to facilitate the programming of intelligent environments in natural language. As depicted in Figure 4, ParlAmI enables the creation of new rules that control the intelligent facilities of the intelligent environment. In particular, according to the LECTOR three-step process, a user via ParlAmI can model a new behavior, specify the conditions under which a combination of behaviors constitutes a trigger that signifies the need for system actions, devise a new intervention rule that specifies how certain triggers are handled, define a new behavior and its respective trigger in a single sentence, or create a trigger and associate it with the intervention that handles it in a single sentence.

Even though the conversation flow varies in each of these cases due to the specific requirements of each intent that guide the user in providing the necessary input to ParlAmI (as is described below), the analysis pipeline is similar. Specifically, the user's input is processed in four discrete stages in order to collect the required data via natural dialogues that "intelligently" distill and propagate them to LECTOR. The chatting engine is built using the ChatScript engine [110], which is a pattern-based heuristics engine that uses a set of rules with pattern-based conditions and is able to keep track of dialogues and support long scripts. When ParlAmI receives a message, it goes through all the patterns until it finds one that matches; if a match is found, then ParlAmI uses the corresponding template to generate a response. A sample pattern (defined in ChatScript's rule-based language) that aims to identify a user intention to create a new behavior model is presented in Figure 5a.

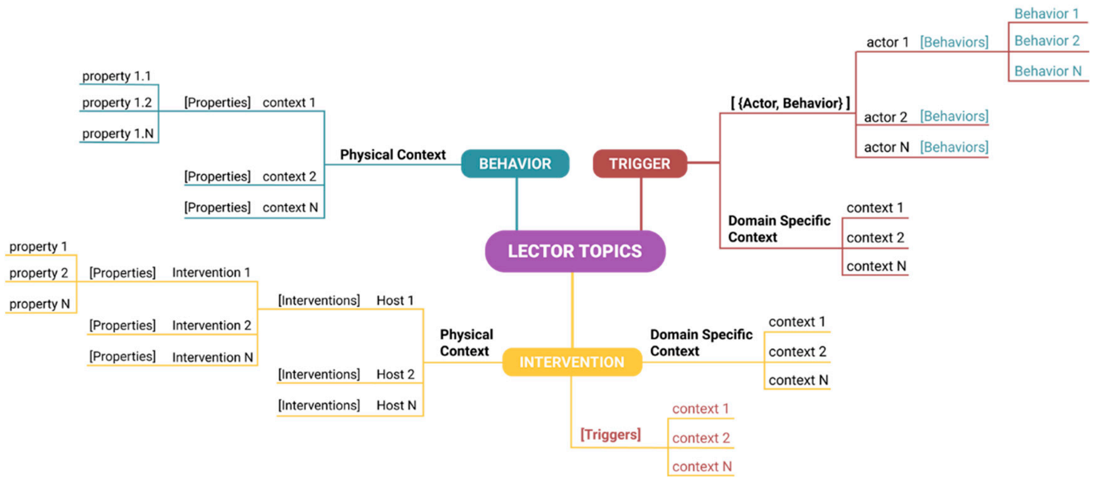
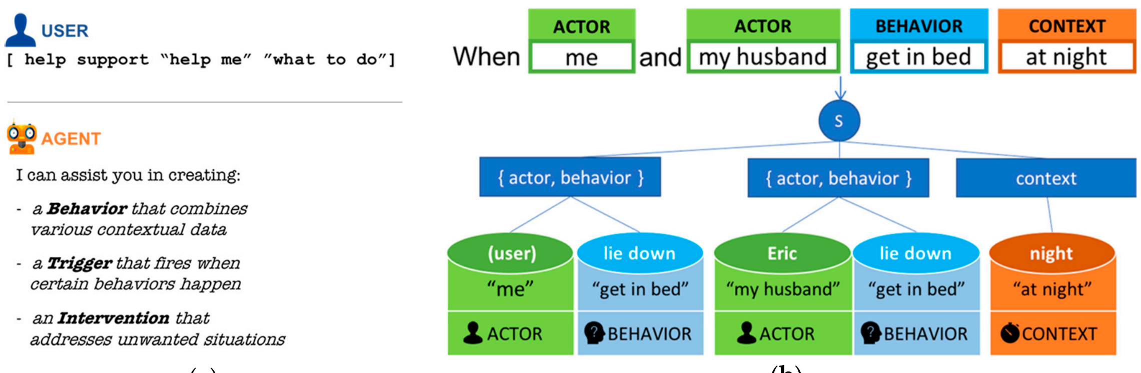


Figure 4. ParlAmI supports a variety of topics that correspond to the different rule types that can be created. Contrary to a sequential user interface (UI), conversations about different topics can be interwoven without affecting the overall functionality.


USER

```
[ help support "help me" "what to do"]
```

AGENT

I can assist you in creating:

- a ***Behavior*** *that combines various contextual data*
- a ***Trigger*** *that fires when certain behaviors happen*
- an ***Intervention*** *that addresses unwanted situations*

(**a**) (**b**)

Figure 5. (**a**) Pattern that identifies user intention to ask for help and get informed about the bot's facilities; (**b**) a sample of the ParlAmI's enhanced syntax tree.

To ensure scalability and maintainability, ParlAmI employs multiple algorithms of varying complexity for pattern matching purposes in each stage of the analysis pipeline (e.g., regular expressions, algorithms utilizing neural networks), which formulate its hierarchical decision-making mechanism (Figure 6b). Nevertheless, since ParlAmI aims to interpret commands given using natural language, finding a match is not guaranteed. To that end, when needed, a generic fallback mechanism asks the user to further clarify its intention (e.g., rephrase a request).

Every dialogue is characterized by a high-level objective (i.e., intent) that reflects a user's intention (e.g., "behavior definition"). Every intent specifies a collection of its relevant arguments (i.e., entities) that ParlAmI's core engine aims to extract from the user input. To that end, it guides the conversation flow accordingly to ensure that at the end of a single session, all of the necessary parameters are fully defined, and LECTOR will be able to generate a valid rule. Figure 4 presents the available high-level intents (behavior, trigger, intervention) with their respective entities being rendered as their direct descendants.

The "behavior" intent guides the user to set the physical context data (FR-3) that LECTOR has to monitor within the environment (FR-4) so as to identify a behavior. The "trigger" intent requires the connection of actors (FR-1) with behaviors (FR-4) and contextual information (FR-2) in a meaningful way, thus forming the conditions of a rule that, when met, will signal the need for system actions (FR-5). Finally, the "intervention" intent expects the user to determine the actions (i.e., intervention types) that the environment has to initiate and present the available intervention hosts (FR-3) when certain triggers are detected (FR-5), customized appropriately to the current context of use (FR-2) in order to address the undesired situation (FR-6).

ParlAmI's conversational control follows a hybrid approach that relies on conditions that guide the dialogue based on what the user has actually said (e.g., a user spoke the exact phrase "help me", as seen in Figure 5a) or meant (Figure 6a), and which entities remain to be defined for the current intent. Figure 6a presents the analysis pipeline that processes users' input from a syntactic and semantic perspective.

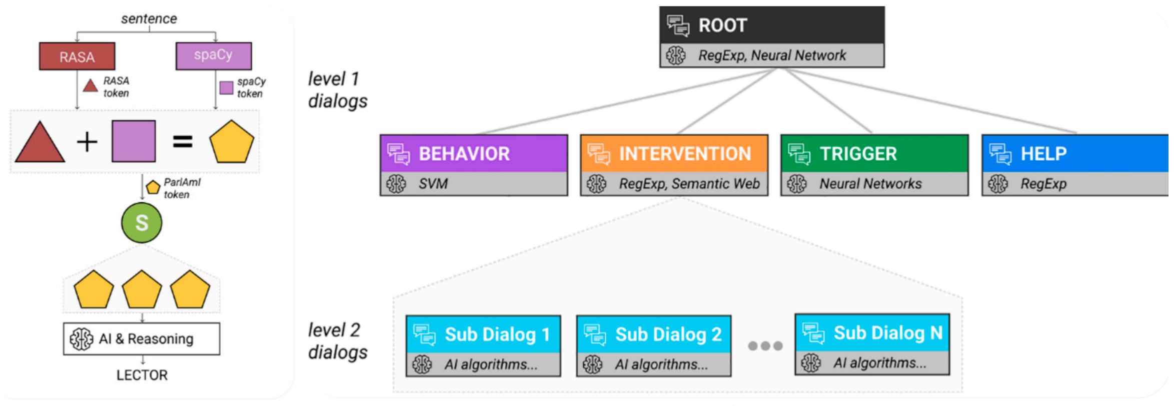


Figure 6. (**a**) The analysis pipeline: Rasa & spaCy process a sentence and their outcomes are combined with context-sensitive data to generate a valid rule; (**b**) ParlAmI's hierarchical decision-making mechanism.

Rasa NLU [111] is an open source Python library for building conversational software that is used to detect the entities in the input sentence as well as the intention behind it. This is done by its intent classification and entity extraction tools. Rasa NLU is basically a set of high level APIs for building a language parser using existing NLP and machine learning libraries [112]. On the other hand, the spaCy [113] library is used to tokenize and annotate words as a part of speech (POS) as well as provide all necessary syntactic information about each word-token in order to construct the syntax tree of the input sentence.

Then, the information stemming from both Rasa NLU and spaCy is combined to create the ParlAmI's representation of the sentence, which is a sophisticated tree aggregating the syntactic information deriving from spaCy and the semantic information from Rasa that artificially creates the necessary additional entries to formulate a complete tree (e.g., multiple entities such as actors). A sample of the ParlAmI's enhanced syntax tree is depicted in Figure 5b, which illustrates how the sentence "when me and my husband get in bed at night" is annotated. Finally, the enhanced tree is further processed by the LECTOR's reasoning modules so as to map the annotated user input with the available intelligent facilities and services of the environment, thus creating a complete rule.

5. The Two Faces of ParlAmI

5.1. HCI Aspects

The multimodal chatbot featured by ParlAmI is available both as a disembodied conversational agent via MAI, which can be launched on the inhabitants' personal devices (e.g., smartphones, tablets), and an embodied conversational agent via nAoMI, which can verbally speak the chatbot's response. Both MAI and nAoMI utilize the following mechanisms offered by the chatbot in order to assist the users to effectively program the behavior of their home while being engaged in a meaningful yet relaxed conversation with the system.

A. User intent identification: when speaking in natural language, different people might use different words/sentences when talking about the same thing (e.g., "YES" versus "SURE"). ParlAmI supports this diversity and permits users to express their directives in any way they desire without compromising the functionality of the system.

B. Message decomposition: different users, depending on their preferences or experience, might build their rules step by step or define a rule in a single sentence (i.e., utterance). ParlAmI's chatbot supports both options, and in case the user provides a complex message, it is able to decompose it into smaller sentences.

- ○ Example B: consider a user creating the following rule: "when I come home, I would like the lights to turn on, the coffee machine to start, and the heater to turn on".

C. Confirmation mechanism: trying to cope with the fact that the chatbot might misunderstand the user's intention, the need for a confirmation mechanism emerged. However, it would be tedious to continuously ask the user to (dis)approve every system action. To this end, the chatbot, before asking a new question, repeats the last acquired message, inviting the user to interrupt the conversation in case he/she identifies a wrong assumption.

- ○ Example C: consider the vague user message "when we leave the house"; in that case, the chatbot assumes that "WE" refers to all family members but subtly prompts the user to confirm or reject that assumption.

D. Error recovery: a user disapproving a system statement means that the chatbot misinterpreted their intention. To this end, it initiates an exploratory process to identify what the source of the problem was by explicitly asking the user what the mistake was (i.e., mimicking the reaction of a human in a similar real-life situation).

- ○ Example D: consider the same vague user message "when we leave the house", where instead of referring to all family members, the user implies only the adults of the family; in that case, the chatbot is mistaken if it assumes that "WE" refers to both adults and children and must be corrected.

E. Exploitation of contextual information: ParlAmI's chatbot relies heavily on contextual information so as to make the conversation appear more natural and alleviate the user from stating the obvious. Additionally, as the authors in [114] suggest, in case of failure to recognize the context, the chatbot initiates a dialogue with the user regarding the context of the statement. Consider the following example where the user provides an ambiguous message: "I want the lights to turn off automatically". The system understands that the user wants to TURN OFF the lights; however, no information is given regarding the room of interest and the desired group of lights to control (e.g., ceiling lights, bedside lamps). In that case, the response of the system depends on the broader context:

 - ◦ Example E1: if this action is part of a rule referring to "leaving the house" (i.e., a "house-oriented" rule), the chatbot assumes that the user wants to control ALL the lights of the HOUSE.
 - ◦ Example E2: if this action is part of a rule referring to "when I watch a movie in the living room" (i.e., a "room-oriented" rule), the system assumes that the user wants to control ONLY the lights of the LIVING ROOM.

F. Rule naming: regarding rule naming, when the user wants to create a rule, the system asks them how they want to name it. If, however, the user skips this step by immediately providing the condition and action of the rule in a single utterance, the system asks for a rule name at the end of the creation process.

G. Hiding complexity: if the user provides a message that touches more than one of the available topics (i.e., behavior, trigger, intervention), the chatbot has the ability to implicitly complete any intermediate steps, hence avoiding burdening the user. For example, the user message "lock the doors when I get in bed at night" requires the creation of a trigger rule and an intervention rule. In that case, the following trigger rule is created implicitly: "during night, user getting into bed means TRIGGER_X". As a next step, the intervention rule is formed as follows: "if TRIGGER_X, then lock the doors". As shown, the outcome of the trigger rule gets an auto-generated name; however, if that specific trigger appears in multiple rules, after completing the conversation, the user is prompted to change the name so as to better reflect its meaning and thus promote its reusability in the future.

5.2. MAI: Messaging Application Interface

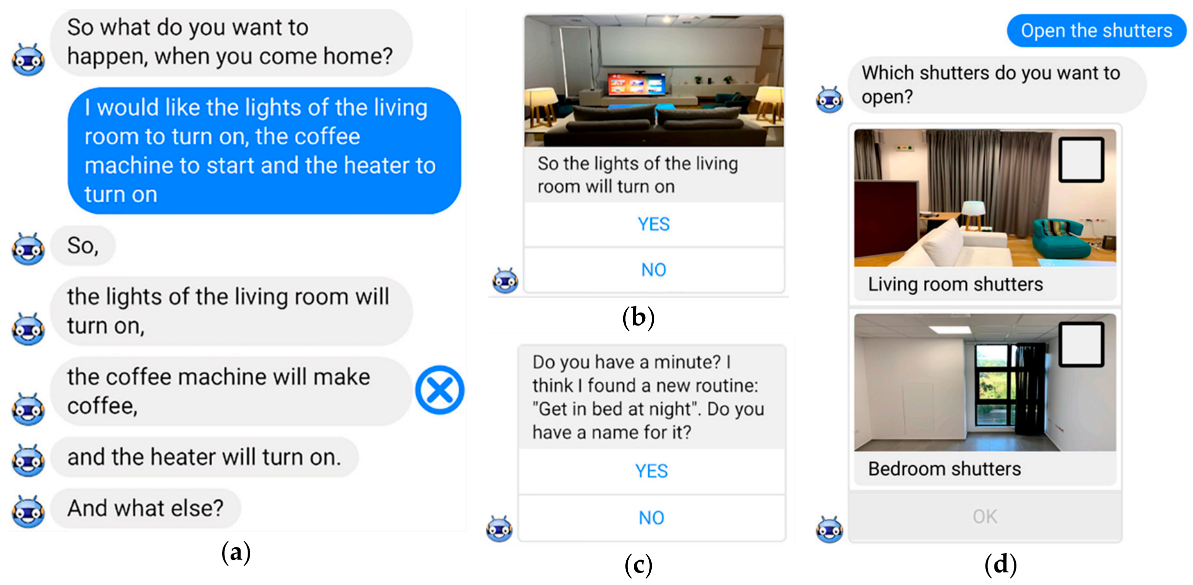
The interface of a messaging application gives to users the impression that they are communicating in real time with a friend despite the fact that in our case, ParlAmI's chatbot is the communicating party. However, MAI does not rely merely on text messages to receive user input; on the contrary, it employs graphical user interface components so as facilitate the communication process. To this end, a core set of custom conversation components were designed based on visual components that users are already familiar with: (i) text, (ii) group of buttons, (iii) text and buttons, (iv) images with text and buttons, and (v) lists of selectable images with text and buttons. These UI elements simplify the user's effort by limiting the required input to just a few options, as reducing the number of attention-grabbing elements simplifies the interface while strengthening the focus on what is actually important [115]. However, in order not to limit user freedom, MAI also permits the users to provide their own input in case they are not satisfied with the suggested pre-defined answers.

Upon launching the application, MAI greets the users with the following message: "hello! What can I help you with?" in order to invite them to engage in a conversation through which they will be able to create, edit, and delete the rules that dictate the behavior of the "Intelligent Home". At this point forward, as the conversation evolves, MAI utilizes the mechanisms mentioned in Section 5.1 so as to ensure usability and improve user experience (UX):

- User intent identification: as soon as a user types a message, before the pipeline processes it, typos and spelling mistakes are automatically corrected by a spellchecker, which ensures that every sentence is correct (at a lexical level).
- Message decomposition: in case the user provides a complex message, the chatbot decomposes it into smaller sentences and then MAI repeats them to the user one by one, permitting them to interrupt the conversation (i.e., selecting a response activates an appropriate control to mark it as incorrect) in case a misconception is identified (Figure 7a).
- Confirmation mechanism: in the case described in Example C, MAI, before asking the user about the desired action that this behavior should trigger, displays the following message: "so what do you want to happen when you, Eric, and the kids leave the house?". That way, the user is immediately provided with feedback regarding what the system understood and can either

intervene to correct the system or proceed with the remaining information that is required. Not interrupting the conversation signifies a confirmation, whereas if the user disagrees with the system, typing "no" or selecting the respective button from the GUI (Figure 7b) initiates the error recovery process.

- Error recovery: in the case described in Example D, the user can interrupt the system (e.g., "no, you did not understand correctly") so as to correct the wrong assumption. Then, MAI displays a "text and buttons" type of message asking the user what the mistake was and then reactivates the respective dialogue so as to retrieve the correct information.
- Exploitation of contextual information: in the case described in Example E1 (where the user wants to control ALL the lights of the HOUSE), MAI displays an "image with text and buttons" message stating the parameters of the rule to be created (i.e., that all the lights of the house will turn off) and invites the user to agree or disagree (by selecting from the available "yes"/"no" buttons). If the user selects "no", MAI initiates the error recovery process and asks the user "what do you want to do instead?". Considering a different context (Example E2, where the user wants to control ONLY the lights of the LIVING ROOM), MAI offers the choices of the available groups of living room lights (e.g., ceiling lights, decorative lights, floor lamps, table lamps) in the form of "list of selectable images with text and buttons" so that the user can select which of the lights they wish to turn off (Figure 7d).
- Rule naming: regarding rule naming, MAI expects the users to provide a name for the new rule at the beginning of the creation process. However, if the user skips this step, MAI repeats the question in the end of the main conversation.
- Hiding complexity: ParlAmI is able to handle the implicit creation of behaviors, triggers, and interventions internally. However, if a recurrent use of the same behavior/trigger/intervention is identified, MAI asks the user to type in a friendly name after completing the conversation (Figure 7c).


(a)

So what do you want to happen, when you come home?

I would like the lights of the living room to turn on, the coffee machine to start and the heater to turn on

So,

the lights of the living room will turn on,

the coffee machine will make coffee,

and the heater will turn on.

And what else?

(b)

So the lights of the living room will turn on

YES

NO

(c)

Do you have a minute? I think I found a new routine: "Get in bed at night". Do you have a name for it?

YES

NO

(d)

Open the shutters

Which shutters do you want to open?

Living room shutters

Bedroom shutters

OK

Figure 7. Mechanisms of ParlAmI as employed by MAI: (**a**) message decomposition; (**b**) confirmation mechanism; (**c**) hiding complexity; (**d**) exploitation of contextual information.

5.3. nAoMI: Robot Interface

The author in [116] recently identified a number of requirements (D1–D10, with D1, D3, D4, D7, and D8 being significantly hard) concerning human-robot interactive communication with both verbal and non-verbal aspects, which ParlAmI attempts to fulfill using nAoMI as an embodied conversational agent.

D1. Breaking the "simple commands only" barrier involves being flexible when interpreting human speech so as not to be constrained by predefined designer responses while being able to initiate dialogue from both the human and the robot perspective. ParlAmI's mechanisms, user intent identification, and message decomposition offer flexibility while interacting in natural language, since complex commands can be analyzed and understood by the system. In addition, nAoMI is able to interoperate with the intelligent environment and initiate dialogue in response to events (e.g., "I have noticed that you always lock the door when you return home, do you want me to create a rule for that?").

D2. Multiple-speech-acts concerns handling more than one types of command with the appropriate robot response, which could be interpreted as a motor action (directive/request) or even a change in the robot's "knowledge" (assertive/declaration). Careful construction of rules and types of input accepted using natural language can improve the support for multiple speech acts in ParlAmI. The rule naming mechanism of ParlAmI, for example, leads to a change in the "mental model" or "situation model" (e.g., "let us name this rule 'Morning Routine'").

D3. Mixed initiative dialogue combines human-initiative dialogue with robot-initiative dialogue where appropriate. This is a feature natively supported by most chatbots where they can either respond to user commands or initiate their own topic of dialogue, which is the case for nAoMI as well.

D4. Situated language and the symbol grounding problem refers to language that is concrete and concerns the here-and-now, which is something that conversational robots can learn to interpret. However, the issue is how they can evolve to understand more abstract entities. Moreover, given that different conversational partners might have different models of meaning for the same entity (e.g., for the lexical semantics of a color term such as "pink"), the confirmation and error recovery mechanisms along with exploitation of contextual information permit nAoMI to have a better understanding of the context of use and drive situated discussions with a conversational partner.

D5. Affective interaction has a significant impact on the quality of conversation. The NAO robot, on the one hand, does not support facial expressions or many options for voice prosody, but on the other hand, it is well equipped to perform gestures and change its pose to convey emotions, which nAoMI uses to express her "emotions" (e.g., sadness when unable to understand the user).

D6. Motor correlates and non-verbal communication is closely related to D5. Additional features—such as face tracking and nodding—on recognized speech support a more natural interaction. For example, as soon as the user gives a command to nAoMI, the robot nods its head to indicate that she heard what he/she said. Furthermore, thanks to ParlAmI's exploitation of contextual information mechanism, while in the middle of creating a rule, nAoMI has the ability to point towards various room artifacts in case the user's command was ambiguous and further input is needed. In the case described in Example E2 (where the user wants to control ONLY the lights of the LIVING ROOM), nAoMI points to the available living room lights (e.g., ceiling lights, decorative lights, floor lamps, table lamps) so that the user can select which of the lights they wish to turn off (Figure 8).

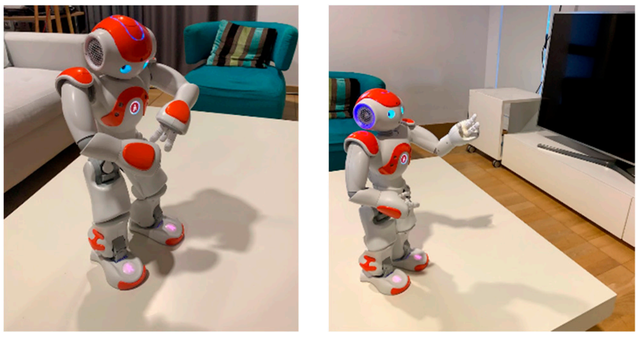

(**a**) (**b**)

Figure 8. (**a**) NAO points to an imaginary watch when referring to time; (**b**) NAO points towards the ceiling lights.

D7. Purposeful speech and planning describes a robot's ability to plan its actions using algorithms or artificial intelligence techniques. This is quite a difficult task to perform correctly due to uncertainty of sensory data and imperfections of speech and language processing systems. In the context of ParlAmI, this desideratum is not directly applicable since this work focuses more on dialogue planning rather than motor planning. To that end, a chat engine that can handle many different dialogue subcases [110] was used so as to plan the response accordingly and dynamically instruct nAoMI how to react.

D8. Multi-level learning is, in a way, already integrated in ParlAmI, as the system's goal is to acquire knowledge on the user's preferences for the intelligent environment. While the knowledge is not stored on the robot itself in this case, it can be used for enhancing future conversations.

D9. Utilization of online resources and services is a core requirement for this system and the robot makes use of various online tools in order to enhance its conversational interface (see Section 4.3 for extensive descriptions of these cloud-based services).

D10. Miscellaneous abilities refer to requirements such as having multiple conversational partners (not currently supported by ParlAmI), multimodal natural language support, and multilingual capabilities. Multimodal natural language support is already considered by offering two different modalities: (i) a spoken dialog with the robot and (ii) a text messaging application through a web app. Finally, since NAO can offer basic multilingual capabilities, a future objective will be to take into consideration the language aspect in ParlAmI's core so as to support interactions in multiple languages (e.g., translate input before importing data to the analysis pipeline, translate output before communicating the result to the user).

6. Results

A user-based evaluation experiment was organized in order to assess the user experience and draw insights by observing the users interacting with the system and noting their comments and general opinion. All subjects gave their informed consent for inclusion before they participated in the study, while the study and its protocol were approved by the Ethics Committee of FORTH-ICS (Reference Number: 22/01-12-2018). In more detail, 16 users of ages 19–45 years (Figure 9) participated in the experiment (seven male in their early-thirties and nine female in their mid-twenties. All participants were familiar with the creation of simple if-then statements (as reported in Table 1), since the purpose of this experiment was not to evaluate their ability to compose such rules. As a matter of fact, previous research [11] has already proven that rule-based programming is intuitive, even for non-programmers.

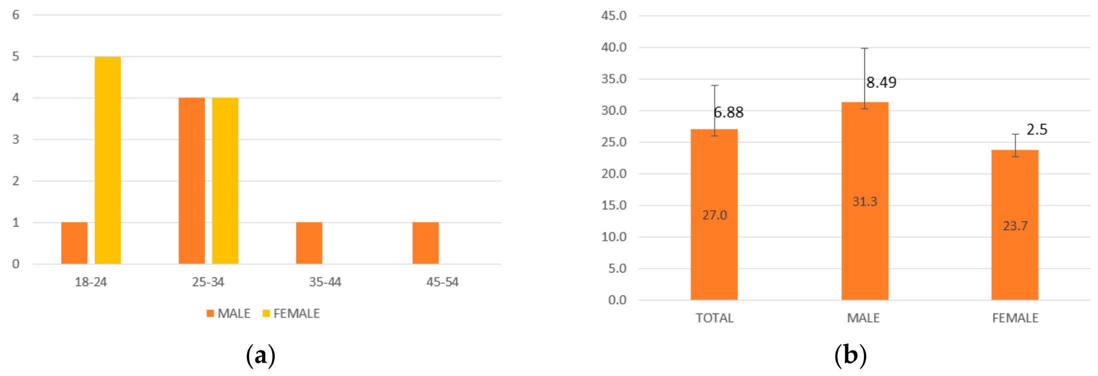


Figure 9. (**a**) Ages of the 16 users; (**b**) mean value and standard deviation of age and gender of the 16 users.

The experiment was divided into two parts; in both parts, each user had to create the same two rules while interacting with either MAI or nAoMI. However, the interaction technique that is first introduced to a user has a disadvantage compared to the second one because the user—during the first part of the experiment—might get accustomed to the rule creation process and complete the second part easily and without any particular problems. Moreover, the order of the two different interactions might bias the opinion of the users positively or negatively towards one interaction paradigm or the other. To this end, half of the participants began the experiment interacting with MAI, while the other half engaged with nAoMI first.

The experiment included four phases, (i) introduction, (ii) the experiment itself, (iii) debriefing and (iv) final interview. All participants attended a single introductory phase while phases two and three were repeated after the completion of each part (interaction with MAI versus interaction with nAoMI). At the end of both parts, a final short interview was held to acquire their general opinions on ParlAmI. The experiment was conducted in the "Intelligent Home" described in Section 3.3.

1. Introduction: during the introduction, the users were given a consent form for their voluntary participation in the experiment and were informed of the overall process. It was made clear that they were not the ones being evaluated, but rather the system, and that they could stop the experiment if they felt uncomfortable at any point. Subsequently, the concept of ParlAmI was explained to them, namely that they could interact with either a messaging application (through a smart phone) or a robot in order to create the rules that dictate the behavior of the "Intelligent Home" using natural language.
2. The experiment itself: in this phase, the users were requested to follow a scenario (Appendix A) including the creation of two rules. The first one was relatively "easy", including a simple trigger and a simple action (intervention) so that the user would grasp the "basics" of the rule making while also having their confidence boosted. The second one was more complicated since it was comprised of multiple actions (interventions). During the experiment, the users were encouraged to express their opinions and thoughts openly following the thinking-aloud protocol [117]. In order to make participants feel comfortable and ensure that the experiment progressed as planned, a facilitator was responsible for orchestrating the entire process, assisting the users when required, and managing any technical difficulties. Furthermore, two note-takers were present in order to record qualitative data (i.e., impressions, comments, and suggestions) along with a number of quantitative data (i.e., number of help requests and errors made).
3. Debriefing: after completing the scenario, the users were handed a user experience questionnaire (UEQ) [118] to measure both classical usability aspects (efficiency, perspicuity, dependability) as well as user experience aspects (originality, stimulation). Afterwards, a debriefing session was held during which the participants were asked some questions regarding their opinion on the system (i.e., MAI or nAoMI), what they liked or disliked the most, and whether they had suggestions about its enhancement.
4. Final interview: users were asked a small set of questions so as to be able to compare the two interaction paradigms—not necessarily to prove if one was better than the other but rather to understand the benefits that each provided, and the situations or conditions under which the users would prefer one interaction paradigm over the other (i.e., MAI versus nAoMI).

During the evaluation experiment, it was revealed that many of the participants, especially non-native English speakers, had a hard time communicating with NAO due to speech recognition issues. In more detail, the system either recognized different commands than the ones the users intended to give or did not recognize any of the users' sayings at all, which happened approximately 80% of the time. This is because NAO (after alerting the users that it is ready to listen with an indicative BEEP sound) records the sounds of its surroundings and sends quite a big data file (probably with a lot of noise) to the Google speech recognition API [119] in order to achieve speech recognition. Additionally, NAO's voice recording mechanism has a threshold for the level of decibels under which it stops recording; the latter led to false terminations while the users were still speaking.

Due to these issues, the first seven participants expressed negative feelings towards nAoMI and were biased in favor of MAI. To this end, for the next nine participants, an external microphone was used in order to better record the users' sayings. That way, speech recognition improved significantly since the recorded data were easier to interpret by the Google speech recognition API. This had an immediate effect on the performance of the system and the opinion of the users, as is observable from the results (Figure 10b,c). The users that faced difficulties while interacting with the nAoMI (i.e., NAO robot) graded the UX poorer in that modality when compared to the text-based alternative.

After analyzing the results of the evaluation, it was revealed that the majority of the participants found MAI easy to use, very responsive, and intelligent, while they also admitted that they would use it in their daily life. Half of them expressed a positive opinion towards the custom UI elements (Section 5.2) that allowed them to select an option instead of typing, while they also found the use of pictures particularly helpful. Additionally, 80% of the participants stated that the interaction was natural and that they enjoyed the natural language conversation that made them feel like they were exchanging messages with a friend. While interacting with MAI, the participants expressed several comments that will be taken into consideration for improving the next version of the system. In more detail, 40% of them pointed out that MAI's confirmation mechanism was too tedious since it was repeating previously acquired information way too often. However, participants who were not used to messaging applications and who admitted to "forgetting easily" were glad to have this feature. This issue can be easily resolved in a following version through a settings mechanism that allows users to set the frequency of confirmation (i.e., feedback) provided. Another valuable insight that was gained through the experiment was that 65% of the participants would prefer to be able to define the complete rule in one sentence instead of building it up step by step. Additionally, 40% of the users revealed that they would like more interactive UI elements and a custom auto-complete mechanism to reduce the amount of typing.

As far as the interaction with nAoMI is concerned, due to the voice recognition issues, participants one through seven expressed only a few positive comments that were mainly regarding the robot's appearance (i.e., cute, pleasant) and the idea of interacting with a humanoid robot. After the addition of the microphone, the attitude of the participants towards the system totally shifted. In more detail, the majority of the participants reacted with joyful comments (e.g., "wow", "super", "perfect") as soon as the interaction began, while during the experiment they expressed various sentiments of joy (e.g., laughed, wanted to shake hands with NAO). Eight out of nine participants who engaged with nAoMI after the microphone was used for voice input stated that they would use the system in their daily life since they found the interaction natural—like talking to a friendly human being.

The first seven users who engaged with nAoMI found it extremely frustrating that they had to wait for the BEEP sound in order to talk, as it goes against the natural instincts we have as humans to start speaking immediately when our speech counterpart finishes speaking. However, this comment was not mentioned by any of the remaining nine participants. Nevertheless, 70% of the 16 participants complained that waiting for the robot to BEEP was time consuming. Interestingly, 70% of the participants did not realize that nAoMI was pointing to different artifacts while talking. This finding should be further investigated in a subsequent experiment that will take place inside the participants' homes instead of a simulation area where they are unaware of the location of available artifacts/equipment.

In general, users were more certain that the system understood them correctly while using MAI, which also relates to the slow response time of NAO (due to the network delay resulting from the process of sending the audio files for recognition to the Google speech recognition API). A lot of users found the interaction with nAoMI "fun" and thought the interaction was more personal and expressive. Participants one through seven revealed that they would prefer MAI, while from the remaining nine, five expressed their fondness towards nAoMI and four towards MAI.

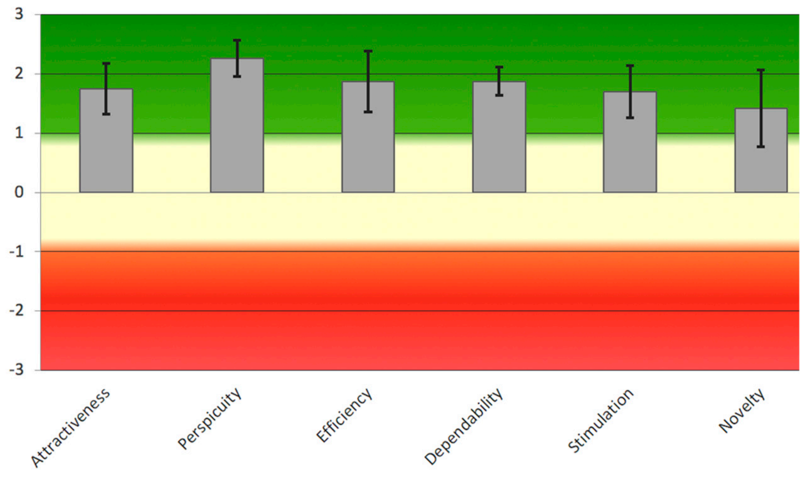
Table 1 presents execution statistics categorized based on the users' previous experience with intelligent systems. Technical expertise is ranked from one to four, with one being novice and four being expert in programming intelligent systems. There was no significant positive relationship between the level of expertise and either the errors made or help asked for both modalities (i.e., messaging application and robot interface). In more detail, the r and p values for the messaging application with respect to errors and number of help were: {r = 0.17, p-value (errors) = 0.52 and {r = 0.13, p-value (help)

= 0.67}, while for the robot, the respective values were: {r = 0.015, *p*-value (errors) = 0.59} and {r = 0.04, *p*-value (help) = 0.89}. Nevertheless, there was a statistically significant positive relationship between the expertise level and the amount of time spent with the system, as confirmed by the following values of r and *p*: {r = 0.59, *p*-value (time with robot) = 0.008} and {r = 0.45, *p*-value (time chatting) = 0.0014}. This is explained by the fact that expert users were willing to go beyond the predefined scenario and create more complex rules so as to explore the facilities of ParlAmI in depth.

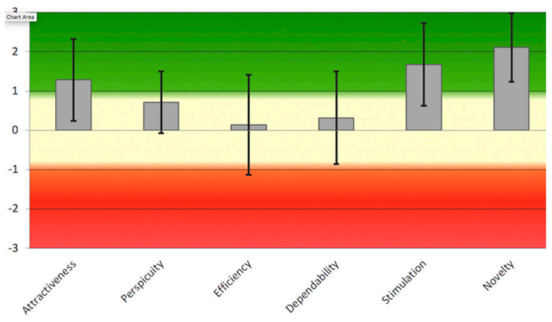
Table 1. Execution statistics categorized based on the users' previous experience with intelligent systems.

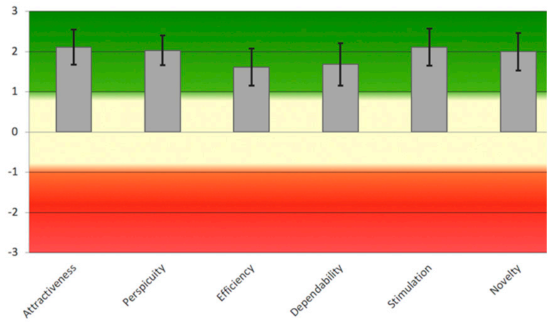
Expertise Level	**Users**	**MAI**			**nAoMI**		
		Avg. Time	**Errors**	**Help**	**Avg. Time**	**Errors**	**Help**
1	4	6:25	2	1	6.31	5	1
2	7	7:29	7	1	7:46	10	8
3	4	9:44	6	3	7:22	1	2
4	1	14:19	0	0	12:30	2	0

After analyzing the results of the UEQ questionnaires, which were handed to the users after completing the scenarios with either MAI or nAoMI, it was revealed that both systems fulfilled the general expectations concerning user experience. The standard interpretation of the scale means is that values between −0.8 and 0.8 represent a neutral evaluation of the corresponding scale, values > 0.8 represent a positive evaluation, and values < −0.8 represent a negative evaluation.



(**a**)





(**b**) (**c**)

Figure 10. Data analysis results of (**a**) UEQ for MAI; (**b**) UEQ for nAoMI for users 1–7; (**c**) UEQ for nAoMI for users 8–16.

In the case of MAI (Figure 10a), all scales show an extremely positive evaluation. The lowest score (i.e., 1.4) regarded novelty, since the users often observed that "using a messaging application is something we already do every day in our lives". On the other hand, nAoMI scored high on novelty, both with users one through seven (Figure 10b) and users eight through sixteen (Figure 10c). As already mentioned, users one through seven, who interacted with nAoMI before the external microphone was used for voice input, found the interaction frustrating and the system not dependable and not at all efficient. The latter is also depicted in Figure 10b, where nAoMI scores low in efficiency, dependability, and perspicuity. However, this impression completely changed for users eight through sixteen since, as depicted in Figure 10c, all scales reveal an extremely positive evaluation. The low score in efficiency can be explained by the fact that even after the using the microphone, there was still some delay for the voice recognition (due to the network delay resulting from the process of sending the audio files for recognition to the Google speech recognition API).

7. Discussion

This paper presented ParlAmI, a conversational framework that delivers a disembodied conversational agent in the form of a messaging application named MAI and an embodied conversational agent named nAoMI by employing the programmable humanoid robot NAO. ParlAmI is deployed in the "Intelligent Home" of FORTH-ICS and aims to enable users to use natural language for creating simple "if-then" rules that dictate the behavior of the environment. After presenting the "Intelligent Home", the characteristics and architecture of ParlAmI were described and the two alternative conversational agents (i.e., MAI and nAoMI) were introduced.

The results of the evaluation experiments, which were conducted with the participation of 16 users, revealed that MAI and nAoMI received an extremely positive evaluation regarding UX, and that users would use either one of them to program the behavior of their Intelligent Homes. While most users found MAI more reliable and effective, nAoMI was received with enthusiasm as something novel, "cute", and "friendly". Despite the fact that nAoMI received negative comments before utilizing an external microphone for voice input, the users' attitudes shifted after the fix to laughing and saying they would definitely nAoMI in their homes. Generally, the positive points of MAI regarded its speed, effectiveness, and reliability, while users thought nAoMI was more novel and "fun".

For the time being, ParlAmI constitutes a functional prototype that still has various limitations. The most obvious one is that it can be used effectively only by English speakers and voice recognition issues can occur when dealing with non-native speakers. Another limitation refers to the fact that ParlAmI currently can only handle a single dialogue with a specific user at a time, which means that a user cannot change topics (e.g., start a conversation to create a new rule and while in the middle, start a new conversation to define a behavior). Finally, even if the flow of the conversation is dynamic (i.e., conditions and actions can be defined in an arbitrary order), users cannot easily modify previously confirmed statements without forcing ParlAmI to re-check everything to maintain consistency.

Regarding future directions, our first goal is to address the issues that were uncovered by the evaluation experiments. Following the iterative design process, another full-scale evaluation is scheduled to take place. Appropriate components will be added to identify contradictions and loops while the possibility of the system learning from the users' feedback to improve dialogues and its internal reasoning mechanisms will be explored. To the last point, we would like to explore the idea of instruction-based-learning, as described in [54], where when the user refers to a route that is not known, the system will learn it by combining primitives as instructed by the user. Additionally, another future plan is to introduce collaboration features to allow different stakeholders (e.g., the different inhabitants of an "Intelligent Home") to cooperate for the definition of a smart space. As pointed out in [83], these features are of particular interest in domains that require the involvement of different stakeholders, such as Ambient Assisted Living and universal access [120]; especially with respect to elderly and people with disabilities, ParlAmI could constitute a powerful tool that has the potential to empower these sensitive user groups to harness the benefits of intelligent environments and feel independent at home.

Author Contributions: Conceptualization, M.K., A.L. and M.A.; Project administration, M.K. and A.L.; Software, E.S., M.F. and D.A.; Supervision, M.A.; Writing—original draft, E.S., M.F., D.A., M.K. and A.L.; Writing—review & editing, M.A.

Funding: This research received no external funding.

Acknowledgments: This work has been supported by the FORTH-ICS internal RTD Programme "Ambient Intelligence and Smart Environments".

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

IoT	Internet of Things
AmI	Ambient Intelligence
EUD	End User Development
TAP	Trigger Action Programming
GUI	Graphical User Interface
UI	User Interface
CI	Conversational Interface
DCA	Disembodied conversational agent
ECA	Embodied conversational agents
PBD	Programming by Demonstration
NLP	Natural Language Processing
CA	Conversational Agent
VR	Virtual Reality
AR	Augmented Reality
NLU	Natural Language Understanding
REST	Representation State Transfer
API	Application Programming Interface
DNS	Domain Name System
AI	Artificial Intelligence
AAL	Ambient Assisted Living
HCI	Human Computer Interaction
UX	User Experience
UEQ	User Experience Questionnaire

Appendix A. Scenarios

Appendix A.1. Task 1

You recently acquired a set of **SMART LIGHTS** for your house, as well as a **SMART DOOR** system that can sense a user entering/being present. However, the light switch is far from the entrance, so every time you come home, you stumble across the furniture trying to reach it and turn the lights on. Therefore, you decide to create a rule so that your home does that automatically for you.

Rule information:

- **Name:** coming home
- **Condition:** you come home
- **Actions:** turn on the lights

Appendix A.2. Task 2

You recently acquired a plethora of smart devices for your home: **presence sensors**, a **smart TV**, a **smart coffee maker**, **smart lights** etc. Therefore, you would like to use the system to create a new rule that automates your morning routine.

Rule information:

- **Name:** morning routine

- **Condition:** it is 8 in the morning
- **Actions:** alarm rings, lights turn on, coffee machine starts, turn on the TV on channel 5, your schedule is displayed on the wall

References

1. Aarts, E.; Wichert, R. Ambient intelligence. In *Technology Guide: Principles–Applications–Trends*; Bullinger, H.-J., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 244–249. ISBN 978-3-540-88546-7.
2. Xia, F.; Yang, L.T.; Wang, L.; Vinel, A. Internet of things. *Int. J. Commun. Syst.* **2012**, *25*, 1101. [CrossRef]
3. Mell, P.; Grance, T. The NIST Definition of Cloud Computing. *Commun. ACM* **2011**, *53*. [CrossRef]
4. Stojkoska, B.L.R.; Trivodaliev, K.V. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* **2017**, *140*, 1454–1464. [CrossRef]
5. Dahl, Y.; Svendsen, R.-M. End-user composition interfaces for smart environments: A preliminary study of usability factors. In Proceedings of the 2011 International Conference of Design, User Experience, and Usability, Orlando, FL, USA, 9–14 July 2011; pp. 118–127.
6. Davidoff, S.; Lee, M.K.; Yiu, C.; Zimmerman, J.; Dey, A.K. Principles of smart home control. In Proceedings of the 8th International Conference on Ubiquitous Computing, Orange County, CA, USA, 17–21 September 2006; pp. 19–34.
7. Dey, A.K.; Sohn, T.; Streng, S.; Kodama, J. iCAP: Interactive prototyping of context-aware applications. In Proceedings of the International Conference on Pervasive Computing, Dublin, Ireland, 7–10 May 2006; pp. 254–271.
8. Newman, M.W.; Elliott, A.; Smith, T.F. Providing an Integrated User Experience of Networked Media, Devices, and Services through End-User Composition. In Proceedings of the Pervasive Computing, Sydney, Australia, 19–22 May 2008; pp. 213–227.
9. Truong, K.N.; Huang, E.M.; Abowd, G.D. CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. In *Proceedings of the International Conference on Ubiquitous Computing*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 143–160.
10. Ur, B.; Pak Yong Ho, M.; Brawner, S.; Lee, J.; Mennicken, S.; Picard, N.; Schulze, D.; Littman, M.L. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, 7–12 May 2016; pp. 3227–3231.
11. Ur, B.; McManus, E.; Pak Yong Ho, M.; Littman, M.L. Practical Trigger-action Programming in the Smart Home. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Toronto, ON, Canada, 26 April–1 May 2014; pp. 803–812.
12. Nacci, A.A.; Balaji, B.; Spoletini, P.; Gupta, R.; Sciuto, D.; Agarwal, Y. Buildingrules: A trigger-action based system to manage complex commercial buildings. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan, 7–11 September 2015; pp. 381–384.
13. Ghiani, G.; Manca, M.; Paternò, F. Authoring context-dependent cross-device user interfaces based on trigger/action rules. In Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia, Linz, Austria, 30 November–2 December 2015; pp. 313–322.
14. Ghiani, G.; Manca, M.; Paternò, F.; Santoro, C. Personalization of context-dependent applications through trigger-action rules. *ACM Trans. Comput. Hum. Interact. (TOCHI)* **2017**, *24*, 14. [CrossRef]
15. IFTTT. Available online: https://ifttt.com/ (accessed on 5 October 2017).
16. Zapier. Available online: https://zapier.com/ (accessed on 7 October 2017).
17. Liu, X.; Wu, D. From Natural Language to Programming Language. In *Innovative Methods, User-Friendly Tools, Coding, and Design Approaches in People-Oriented Programming*; IGI Global: Hershey, PA, USA, 2018; pp. 110–130.
18. Amazon Alexa. Available online: https://developer.amazon.com/alexa (accessed on 5 November 2018).
19. McTear, M.; Callejas, Z.; Griol, D. *The Conversational Interface: Talking to Smart Devices*; Springer: Berlin, Germany, 2016; ISBN 3-319-32967-7.
20. Serrano, J.; Gonzalez, F.; Zalewski, J. CleverNAO: The intelligent conversational humanoid robot. In Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, 24–26 September 2015; Volume 2, pp. 887–892.

21. Ruttkay, Z.; Pelachaud, C. *From Brows to Trust: Evaluating Embodied Conversational Agents*; Springer Science & Business Media: Berlin, Germany, 2006; Volume 7, ISBN 1-4020-2730-3.
22. King, W.J.; Ohya, J. The representation of agents: Anthropomorphism, agency, and intelligence. In Proceedings of the Conference Companion on Human Factors in Computing Systems, Vancouver, BC, Canada, 13–18 April 1996; pp. 289–290.
23. Araujo, T. Living up to the chatbot hype: The influence of anthropomorphic design cues and communicative agency framing on conversational agent and company perceptions. *Comput. Hum. Behav.* **2018**, *85*, 183–189. [CrossRef]
24. Stefanidi, E.; Korozi, M.; Leonidis, A.; Antona, M. Programming Intelligent Environments in Natural Language: An Extensible Interactive Approach. In Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference, Corfu, Greece, 26–29 June 2018; pp. 50–57.
25. Nao (robot). Available online: https://en.wikipedia.org/wiki/Nao_(robot) (accessed on 10 December 2018).
26. Korozi, M.; Leonidis, A.; Antona, M.; Stephanidis, C. LECTOR: Towards Reengaging Students in the Educational Process Inside Smart Classrooms. In Proceedings of the International Conference on Intelligent Human Computer Interaction, Evry, France, 11–13 December 2017; pp. 137–149.
27. Lieberman, H.; Paternò, F.; Klann, M.; Wulf, V. End-User Development: An Emerging Paradigm. In *End User Development*; Lieberman, H., Paternò, F., Wulf, V., Eds.; Human-Computer Interaction Series; Springer Netherlands: Dordrecht, The Netherlands, 2006; pp. 1–8. ISBN 978-1-4020-5386-3.
28. Dertouzos, M.L.; Dertrouzos, M.L.; Foreword By-Gates, B. *What Will Be: How the New World of Information Will Change Our Lives*; HarperCollins Publishers: New York, NY, USA, 1998; ISBN 0-06-251540-3.
29. MacLean, A.; Carter, K.; Lövstrand, L.; Moran, T. User-tailorable systems: Pressing the issues with buttons. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Seattle, WA, USA, 1–5 April 1990; pp. 175–182.
30. Wulf, V.; Golombek, B. Direct activation: A concept to encourage tailoring activities. *Behav. Inf. Technol.* **2001**, *20*, 249–263. [CrossRef]
31. Won, M.; Stiemerling, O.; Wulf, V. Component-based approaches to tailorable systems. In *End User Development*; Springer: Berlin, Germany, 2006; pp. 115–141.
32. Quirk, C.; Mooney, R.J.; Galley, M. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15), Beijing, China, July 2015; pp. 878–888.
33. You Don't Need to Be a Coder. Available online: https://bubble.is/ (accessed on 6 December 2018).
34. Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A.; Rosenbaum, E.; Silver, J.; Silverman, B. Scratch: Programming for all. *Commun. ACM* **2009**, *52*, 60–67. [CrossRef]
35. Cooper, S.; Dann, W.; Pausch, R. Alice: A 3-D tool for introductory programming concepts. In Proceedings of the Fifth Annual CCSC Northeastern Conference on the Journal of Computing in Small Colleges, Mahwah, NJ, USA, 28–29 April 2000; Volume 15, pp. 107–116.
36. Snap! Build Your Own Blocks 4.2.2.9. Available online: https://snap.berkeley.edu/snapsource/snap.html (accessed on 6 December 2018).
37. Broll, B.; Lédeczi, Á.; Zare, H.; Do, D.N.; Sallai, J.; Völgyesi, P.; Maróti, M.; Brown, L.; Vanags, C. A visual programming environment for introducing distributed computing to secondary education. *J. Parallel Distrib. Comput.* **2018**, *118*, 189–200. [CrossRef]
38. Myers, B.A. *Creating User Interfaces by Demonstration*; Academic Press Professional: San Diego, CA, USA, 1987.
39. Myers, B.A.; McDaniel, R.G.; Kosbie, D.S. Marquise: Creating complete user interfaces by demonstration. In Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems, Amsterdam, The Netherlands, 24–29 April 1993; pp. 293–300.
40. Cypher, A.; Smith, D.C. KidSim: End user programming of simulations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 7–11 May 1995; pp. 27–34.
41. Little, G.; Lau, T.A.; Cypher, A.; Lin, J.; Haber, E.M.; Kandogan, E. Koala: Capture, share, automate, personalize business processes on the web. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, 28 April–3 May 2007; pp. 943–946.

42. Leshed, G.; Haber, E.M.; Matthews, T.; Lau, T. CoScripter: Automating & sharing how-to knowledge in the enterprise. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy, 5–10 April 2008; pp. 1719–1728.
43. Dey, A.K.; Hamid, R.; Beckmann, C.; Li, I.; Hsu, D. A CAPpella: Programming by demonstration of context-aware applications. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vienna, Austria, 24–29 April 2004; pp. 33–40.
44. Tasker for Android. Available online: https://tasker.joaoapps.com/ (accessed on 6 December 2018).
45. Automate-Apps on Google Play. Available online: https://play.google.com/store/apps/details?id=com.llamalab.automate&hl=en (accessed on 6 December 2018).
46. Sammet, J.E. The use of English as a programming language. *Commun. ACM* **1966**, *9*, 228–230. [CrossRef]
47. Heidorn, G.E. Automatic programming through natural language dialogue: A survey. *IBM J. Res. Dev.* **1976**, *20*, 302–313. [CrossRef]
48. Miller, L.A. Natural language programming: Styles, strategies, and contrasts. *IBM Syst. J.* **1981**, *20*, 184–215. [CrossRef]
49. Ballard, B.W. Semantic and Procedural Processing for a Natural Language Programming System. Ph.D. Thesis, Duke University, Durham, NC, USA, 1980.
50. Ballard, B.W.; Biermann, A.W. Programming in natural language: "NLC" as a prototype. In Proceedings of the 1979 Annual Conference, New York, NY, USA, 16–17 August 1979; pp. 228–237.
51. Biermann, A.W.; Ballard, B.W. Toward natural language computation. *Comput. Linguist.* **1980**, *6*, 71–86.
52. Jain, A.; Kulkarni, G.; Shah, V. Natural language processing. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 161–167. [CrossRef]
53. Stenmark, M.; Nugues, P. Natural language programming of industrial robots. In Proceedings of the 2013 44th International Symposium on Robotics (ISR), Seoul, Korea, 24–26 October 2013; pp. 1–5.
54. Lauria, S.; Bugmann, G.; Kyriacou, T.; Klein, E. Mobile robot programming using natural language. *Robot. Auton. Syst.* **2002**, *38*, 171–181. [CrossRef]
55. Williams, T.; Scheutz, M. The state-of-the-art in autonomous wheelchairs controlled through natural language: A survey. *Robot. Auton. Syst.* **2017**, *96*, 171–183. [CrossRef]
56. Good, J.; Howland, K. Programming language, natural language? Supporting the diverse computational activities of novice programmers. *J. Vis. Lang. Comput.* **2017**, *39*, 78–92. [CrossRef]
57. Liu, H.; Lieberman, H. Metafor: Visualizing stories as code. In Proceedings of the 10th international conference on Intelligent user interfaces, San Diego, CA, USA, 10–13 January 2005; pp. 305–307.
58. Liu, H.; Lieberman, H. Programmatic semantics for natural language interfaces. In Proceedings of the CHI'05 Extended Abstracts on Human Factors in Computing Systems, Portland, OR, USA, 2–7 April 2005; pp. 1597–1600.
59. Lieberman, H.; Ahmad, M. Knowing what you're talking about: Natural language programming of a multi-player online game. In *No Code Required*; Elsevier: New York, NY, USA, 2010; pp. 331–343.
60. Nelson, G. Natural language, semantic analysis, and interactive fiction. *IF Theory Reader* **2006**, *141*, 99–104.
61. Nelson, G. Afterword: Five years later. In *IF Theory Reader*; Jackson-Mead, K., Wheeler, J.R., Eds.; Transcript On Press: Norman, OK, USA, 2011; pp. 189–202.
62. McTear, M.; Callejas, Z.; Griol, D. Introducing the Conversational Interface. In *The Conversational Interface*; Springer: Berlin, Germany, 2016; pp. 1–7.
63. Huang, T.-H.K.; Azaria, A.; Bigham, J.P. Instructablecrowd: Creating if-then rules via conversations with the crowd. In Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, San Jose, CA, USA, 7–12 May 2016; pp. 1555–1562.
64. Thórisson, K.R. Gandalf: An embodied humanoid capable of real-time multimodal dialogue with people. In Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA, USA, 5–8 February 1997; pp. 536–537.
65. Cassell, J.; Thorisson, K.R. The power of a nod and a glance: Envelope vs. emotional feedback in animated conversational agents. *Appl. Artif. Intell.* **1999**, *13*, 519–538. [CrossRef]
66. Cassell, J.; Bickmore, T.; Vilhjálmsson, H.; Yan, H. More than just a pretty face: Affordances of embodiment. In Proceedings of the 5th International Conference on Intelligent User Interfaces, New Orleans, LA, USA, 9–12 January 2000; pp. 52–59.

67. Kim, K.; Boelling, L.; Haesler, S.; Bailenson, J.N.; Bruder, G.; Welch, G. Does a Digital Assistant Need a Body? The Influence of Visual Embodiment and Social Behavior on the Perception of Intelligent Virtual Agents in AR. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality, Munich, Germany, 16–20 October 2018.
68. Bailenson, J.N.; Blascovich, J.; Beall, A.C.; Loomis, J.M. Equilibrium theory revisited: Mutual gaze and personal space in virtual environments. *Presence Teleoper. Virtual Environ.* **2001**, *10*, 583–598. [CrossRef]
69. Bailenson, J.N.; Blascovich, J.; Beall, A.C.; Loomis, J.M. Interpersonal distance in immersive virtual environments. *Personal. Soc. Psychol. Bull.* **2003**, *29*, 819–833. [CrossRef] [PubMed]
70. Mutlu, B.; Forlizzi, J.; Hodgins, J. A storytelling robot: Modeling and evaluation of human-like gaze behavior. In Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, 4–6 December 2006; pp. 518–523.
71. Wainer, J.; Feil-Seifer, D.J.; Shell, D.A.; Mataric, M.J. The role of physical embodiment in human-robot interaction. In Proceedings of the ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication, Hatfield, UK, 6–8 September 2006; pp. 117–122.
72. Heerink, M.; Krose, B.; Evers, V.; Wielinga, B. Observing conversational expressiveness of elderly users interacting with a robot and screen agent. In Proceedings of the 2007 IEEE 10th International Conference on Rehabilitation Robotics, Noordwijk, The Netherlands, 13–15 June 2007; pp. 751–756.
73. Pereira, A.; Martinho, C.; Leite, I.; Paiva, A. iCat, the chess player: The influence of embodiment in the enjoyment of a game. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3, Estoril, Portugal, 12–16 May 2008; pp. 1253–1256.
74. Li, J. The benefit of being physically present: A survey of experimental works comparing copresent robots, telepresent robots and virtual agents. *Int. J. Hum. Comput. Stud.* **2015**, *77*, 23–37. [CrossRef]
75. Amft, O.; Novais, P.; Tobe, Y.; Chatzigiannakis, I. *Intelligent Environments 2018: Workshop Proceedings of the 14th International Conference on Intelligent Environments*; IOS Press: Amsterdam, The Netherlands, 2018.
76. García-Herranz, M.; Haya, P.A.; Alamán, X. Towards a Ubiquitous End-User Programming System for Smart Spaces. *J. UCS* **2010**, *16*, 1633–1649.
77. Heun, V.; Hobin, J.; Maes, P. Reality editor: Programming smarter objects. In Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, Zurich, Switzerland, 8–12 September 2013; pp. 307–310.
78. Heun, V.; Kasahara, S.; Maes, P. Smarter objects: Using AR technology to program physical objects and their interactions. In Proceedings of the CHI'13 Extended Abstracts on Human Factors in Computing Systems, Paris, France, 27 April–2 May 2013; pp. 961–966.
79. Ens, B.; Anderson, F.; Grossman, T.; Annett, M.; Irani, P.; Fitzmaurice, G. Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In Proceedings of the 43rd Graphics Interface Conference, Edmonton, AB, Canada, 16–19 May 2017; pp. 156–162.
80. Lee, G.A.; Nelles, C.; Billinghurst, M.; Kim, G.J. Immersive authoring of tangible augmented reality applications. In Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, Arlington, VA, USA, 5 November 2004; pp. 172–181.
81. Sandor, C.; Olwal, A.; Bell, B.; Feiner, S. Immersive mixed-reality configuration of hybrid user interfaces. In Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05), Vienna, Austria, 5–8 October 2005; pp. 110–113.
82. Steed, A.; Slater, M. A dataflow representation for defining behaviours within virtual environments. In Proceedings of the Virtual Reality Annual International Symposium, Santa Clara, CA, USA, 30 March–3 April 1996; pp. 163–167.
83. Desolda, G.; Ardito, C.; Matera, M. Empowering end users to customize their smart environments: Model, composition paradigms, and domain-specific tools. *ACM Trans. Comput. Hum. Interact. (TOCHI)* **2017**, *24*, 12. [CrossRef]
84. Walch, M.; Rietzler, M.; Greim, J.; Schaub, F.; Wiedersheim, B.; Weber, M. homeBLOX: Making home automation usable. In Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication, Zurich, Switzerland, 8–12 September 2013; pp. 295–298.
85. Zipato. Available online: https://www.zipato.com/ (accessed on 7 October 2017).
86. Supermechanical: Twine. Listen to Your World. Talk to the Web. Available online: http://supermechanical.com/twine/technical.html (accessed on 6 October 2017).

87. WigWag. Available online: https://www.wigwag.com/ (accessed on 6 October 2017).
88. AI, R. Resonance | Predicting Human Behaviours. Available online: https://www.resonance-ai.com/ (accessed on 7 October 2017).
89. De Ruyter, B.; Van De Sluis, R. Challenges for End-User Development in Intelligent Environments. In *End User Development*; Lieberman, H., Paternò, F., Wulf, V., Eds.; Springer Netherlands: Dordrecht, The Netherlands, 2006; pp. 243–250. ISBN 978-1-4020-5386-3.
90. Leonidis, A.; Arampatzis, D.; Louloudakis, N.; Stephanidis, C. The AmI-Solertis System: Creating User Experiences in Smart Environments. In Proceedings of the 13th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Rome, Italy, 9–11 October 2017.
91. Preuveneers, D.; Van den Bergh, J.; Wagelaar, D.; Georges, A.; Rigole, P.; Clerckx, T.; Berbers, Y.; Coninx, K.; Jonckers, V.; De Bosschere, K. Towards an extensible context ontology for ambient intelligence. In Proceedings of the European Symposium on Ambient Intelligence, Eindhoven, The Netherlands, 8–11 November 2004; pp. 148–159.
92. Hohpe, G.; Woolf, B. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*; Addison-Wesley: Boston, MA, USA, 2004; ISBN 0-321-20068-3.
93. Menge, F. *Enterprise Service Bus*; O'Reilly Media, Inc.: Newton, MA, USA, 2007; Volume 2, pp. 1–6.
94. OpenAPI Specification | Swagger. Available online: https://swagger.io/specification/ (accessed on 9 December 2018).
95. Hoareau, D.; Mahéo, Y. Middleware support for the deployment of ubiquitous software components. *Pers. Ubiquitous Comput.* **2008**, *12*, 167–178. [CrossRef]
96. Dawson, M.R.W. *Minds and Machines: Connectionism and Psychological Modeling*; John Wiley & Sons: Hoboken, NJ, USA, 2008; ISBN 978-0-470-75298-2.
97. Siegel, M. The sense-think-act paradigm revisited. In Proceedings of the 1st International Workshop on Robotic Sensing, 2003. ROSE' 03, Orebo, Sweden, 5–6 June 2003; p. 5.
98. Wireless and Smart Lighting by Philips | Meet Hue. Available online: http://www2.meethue.com/en-us (accessed on 19 January 2018).
99. Motorized Blinds, Shades, Awnings and Curtains with Somfy. Available online: https://www.somfysystems.com (accessed on 27 November 2018).
100. RENPHO Essential Oil Diffuser WiFi Smart Humidifier Works with Alexa, Google Assistant and APP, 120ml Ultrasonic Aromatherapy Diffuser for Home Office, Adjustable Cool Mist, Waterless Auto Shut-off. Available online: https://renpho.com/Health/essential-oil-diffusers/product-950.html (accessed on 6 December 2018).
101. Merz, H.; Hansemann, T.; Hübner, C. *Building Automation: Communication Systems with EIB/KNX, LON and BACnet*; Springer Science & Business Media: Berlin, Germany, 2009.
102. Kinect-Windows App Development. Available online: https://developer.microsoft.com/en-us/windows/kinect (accessed on 9 December 2018).
103. Withings Sleep. Available online: /uk/en/sleep (accessed on 6 December 2018).
104. EMFIT Sleep Tracking & Monitoring with Heart-Rate-Variability. Available online: https://www.emfit.com (accessed on 9 December 2018).
105. Hybrid Smartwatch | Steel HR-Withings. Available online: /ca/en/steel-hr (accessed on 6 December 2018).
106. Fitbit Charge 3 | Advanced Health and Fitness Tracker. Available online: https://www.fitbit.com/eu/charge3 (accessed on 6 December 2018).
107. Smart Lock-Keyless Electronic Door Lock for Smart Access. Available online: https://nuki.io/en/ (accessed on 27 November 2018).
108. What is Choregraphe? | SoftBank Robotics Community. Available online: https://community.ald.softbankrobotics.com/en/resources/faq/developer/what-choregraphe (accessed on 6 December 2018).
109. Cython: C-Extensions for Python. Available online: https://cython.org/ (accessed on 10 December 2018).
110. Wilcox, B. Chatscript. Available online: http://meta-guide.com/bots/chatbots/chatscript (accessed on 10 December 2018).
111. Bocklisch, T.; Faulker, J.; Pawlowski, N.; Nichol, A. Rasa: Open Source Language Understanding and Dialogue Management. *arXiv*, 2017; arXiv:1712.05181.
112. Language Understanding with Rasa NLU—Rasa NLU 0.10.5 Documentation. Available online: https://rasa-nlu.readthedocs.io/en/latest/index.html# (accessed on 22 December 2017).

113. spaCy-Industrial-Strength Natural Language Processing in Python. Available online: https://spacy.io/index (accessed on 22 December 2017).
114. Lieberman, H.; Liu, H. Feasibility studies for programming in natural language. In *End User Development*; Springer: Berlin, Germany, 2006; pp. 459–473.
115. Brinck, T.; Gergle, D.; Wood, S.D. *Usability for the Web: Designing Web Sites that Work*; Morgan Kaufmann: Burlington, MA, USA, 2001; ISBN 0-08-052031-6.
116. Mavridis, N. A review of verbal and non-verbal human–robot interactive communication. *Robot. Auton. Syst.* **2015**, *63*, 22–35. [CrossRef]
117. Thinking Aloud: The #1 Usability Tool. Available online: https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/ (accessed on 31 October 2017).
118. User Experience Questionnaire (UEQ). Available online: https://www.ueq-online.org/ (accessed on 6 December 2018).
119. Cloud Speech-to-Text-Speech Recognition | Cloud Speech-to-Text API. Available online: https://cloud.google.com/speech-to-text/ (accessed on 9 December 2018).
120. Stephanidis, C. *The Universal Access Handbook*; CRC Press: Boca Raton, FL, USA, 2009; ISBN 1-4200-6499-1.