Empowering Caregivers To Customizing The Assistive Computing Support of Older Adults - An End-User Domain-Specific Approach

Stéphanie Giraud, Nic Volanschi, Charles Consel

▶ To cite this version:

Stéphanie Giraud, Nic Volanschi, Charles Consel. Empowering Caregivers To Customizing The Assistive Computing Support of Older Adults - An End-User Domain-Specific Approach. International Journal of Human-Computer Interaction, Taylor & Francis, 2020, 10.1080/10447318.2020.1752479 . hal-02551047

HAL Id: hal-02551047

https://hal.archives-ouvertes.fr/hal-02551047

Submitted on 22 Apr 2020

Empowering Caregivers To Customizing The Assistive Computing Support of Older Adults - An End-User Domain-Specific Approach

Stéphanie Giraud[a] and Nic Volanschi[a] and Charles Consel[a,b]

[a]Inria Bordeaux – Sud-Ouest; [b]Bordeaux INP

ARTICLE HISTORY
Received: September 2019
Accepted: April 2020

ABSTRACT
Smart homes are a promising infrastructure to support assistive services that can prolong independent living of older adults. However, smart homes are mostly developed using a technology-centered approach, raising challenges for users to comprehend the purpose of smart home services and anticipate their behavior. These challenges are more acute when services are to be deployed in homes of older adults, preventing the expert knowledge of their caregivers to be leveraged.

This paper presents a user study that assesses the comprehensibility of an end-user language, named Maloya, dedicated to developing assistive services. Participants (9 professional caregivers) are presented with services written in Maloya and must determine whether they detect scenarios of daily living.

Our results show that Maloya is well-understood by participants; they successfully determine whether a service detects a scenario (success rate of 94%) and substantiate their answers with consistent explanations. As such, Maloya should be effective in empowering caregivers to select appropriate services for their care-receivers and to accurately anticipate the service behavior.

KEYWORDS
Smart home, Ambient assisted living, Domain-specific language, User study, Readability.

1. Introduction

The rapidly growing population of older adults calls for the emergence of a research field, dedicated to developing assistive technologies (ATs), which support independent living. ATs must meet the needs of both older adults and their caregivers, and prevent the transition of older adults to costly care-giving facilities (Tang & Venables, 2000). Concretely, the goal of ATs is to assist older adults in all aspects of their daily life, including daily activities (*e.g.,* meal preparation monitoring, appointment or medication reminder), user safety (*e.g.,* stove or entrance door monitoring), health management (*e.g.,* medication management, exercise coaching, meal tracker), and social participation (*e.g.,* collaborative games, social networks, video chat apps) (Baecker, Moffatt, & Massimi, 2012). Increasingly, the design of these ATs takes into account the capabilities, limitations and motivations of older adults (Czaja, Rogers, Fisk, Charness, & Sharit, 2009). When properly designed, developed and deployed, ATs have the potential (1) to inform an older adult or their clinician about their status, (2) to detect significant changes thereof, and (3) to suggest an intervention for addressing a problem detected by such monitoring systems (Schulz et al., 2014).

However, in practice, older adults have heterogeneous capacities, needs and preferences. Age-related changes vary across individuals, depending on a number of factors, including their level of education, their socio-economic category, their past professional activities, their lifestyle, and their diet. As a result, each older adult needs specific ATs, offering specific functionalities and specific configuration capababilities. Addressing the specificities of older adults, at the scale of a home, requires a highly customizable approach, as shown by Consel *et al.* with their assisted living platform for aging in place, named *HomeAssist* (Consel, Dupuy, & Sauzéon, 2017). Specifically, HomeAssist achieves customization via an open-ended catalog

CONTACT Charles Consel. Email: charles.consel@bordeaux-inp.fr

of assistive services, providing individual older adults and their caregivers with a mechanism to selectively address the user's needs and preferences. Their approach was validated with 129 deployments in the home of single, community-dwelling, older adults (Consel et al., 2017). In practice, each service of the catalog was presented to the participants to determine whether it was (1) useful, provided some parameterization, (2) useful, but required some adaptation, or (3) inappropriate. A service in the first category is, for example, one that requires the time interval of an activity (*e.g.,* breakfast preparation) to be defined. The second category of services is one that requires an adaptation that, for example, involves calling a caregiver if a meal is not prepared after a given time interval. The last category of services often corresponds to a mismatch with the user's lifestyle. For example, a user eating most of their meals out defeats the purpose of a service monitoring meal preparation. As illustrated by these examples, the more services are interwoven into the user's daily life, the more they need to be customized to match their daily activities.

To produce a large and highly customizable catalog of services, existing approaches are mostly programmatic. To address the many services that must be produced, software engineering research has proposed software development methodologies (*e.g.,* Alegre, Augusto, & Clark, 2016; Cassou, Bruneau, Consel, & Balland, 2012), raised the level of abstraction of mainstream programming languages with design languages (*e.g.,* Cassou, Balland, Consel, & Lawall, 2011; Chauhan, Patel, Sureka, Delicato, & Chaudhary, 2016), and introduced software layers and tools dedicated to the domain of pervasive computing (*e.g.,* Bertran et al., 2014; Salber, Dey, & Abowd, 1999). HomeAssist's catalog is a prime outcome of this line of research; it proposes a large set of services, addressing a range of areas of assistance for the daily life activities of older adults (Consel et al., 2017): night-time wandering, stove and oven hazards, suspected falls, deviations from meal routines, *etc.* In fact, each area of assistance often requires different services, depending on the needs and preferences of users. For example, monitoring the meal routines of users may involve (1) a simple weekly report on how well they followed their routines, (2) a notification reminding them to initiate their routines, (3) an application to be launched to suggest and guide them through a recipe, or even (4) the support of a caregiver that is called when a user does not prepare their meal. Not only does a range of services need to be developed to address an area of assistance, but each service needs to be configurable to account for fine-grained specificities of users, as mentioned earlier (*e.g.,* time interval of an activity).

Although the programmatic approach allows to address a range of user specificities, it is not realistic in practice at full scale. Indeed, even if the development of services could be sustained to cover the entire domain of aging in place, this approach is out of reach for users because it requires them to first understand the behavior of available services for each relevant area of assistance, then to try to match their behavior and configuration capabilities against the specificities of a user, and to eventually deploy and validate the selected services. These tasks need to be performed by the users and/or their caregivers because they are the most knowledgeable about the user's specificities, as showed by Dupuy, Sauzéon, and Consel (2015), and thus whether a service is likely to be successfully interwoven into the users' daily life. The resulting burden on older adults and their caregivers can be significant because they need to dive into a potentially heavy documentation, assuming the assistive service is properly documented, to ensure that it is appropriate for a user, and if so, train them to anticipate its behavior. If inappropriate and/or misunderstood, the assistance delivered by a service eventually becomes ignored, often leading the older user to discouragement, loss of confidence in the service, and the desire to discard other services (Brush et al., 2011).

So the question arises of how to bridge the gap between users and service developers, so that the former accurately understand assistive services without requiring extensive documentation from the latter. Our proposal is to provide service developers with a form of programming language that is understandable by the users (*i.e.,* caregivers). To achieve this goal on the developer side, this language would need to be expressive enough to cover the scope of assistive services without incurring an overhead in the development process (Volanschi, Carteron, & Consel, 2018; Volanschi, Serpette, Carteron, & Consel, 2018). On the user side, the understandability of the language would need to be evaluated by end-users.

Our approach. We present the evaluation of a domain-specific language (DSL) dedicated to assistive services by professional caregivers and demonstrate its comprehensibility by these users. This language, named *Maloya*, allows developers to formulate assistive services with everyday, domain-specific terms (Volanschi, Carteron, & Consel, 2018). At the heart of Maloya is a layer devoted to extensively describing the contexts in which assistance must be delivered, leveraging existing works on the design and development of context-aware assistive systems (*e.g.,* Consel et al., 2017; Mihailidis & Fernie, 2002; Nehmer, Becker, Karshmer, & Lamm, 2006; Salber et al., 1999) and their potential benefits for aging in place (*e.g.,* Dawadi, Cook, & Schmitter-Edgecombe, 2013; Kaye et al., 2011; Rashidi & Mihailidis, 2013). Another layer of the language is dedicated to defining the actions to be triggered when a context requiring assistance has been detected. As such, Maloya can be viewed as a user-understandable, domain-specific language, dedicated to developing context-aware assistive services. To illustrate the language, consider a service aiming to monitor the entrance door at night for older adults with night wandering disorders. The context in which an alert should be triggered can be written in Maloya as follows.

Door **is** open **occurs for** 5 minutes **while** Night time

If such a context is detected, a typical action consists of triggering a notification to the user and/or the caregiver.

Our approach to assessing the comprehensibility of Maloya consists of presenting professional caregivers with assistive services written in Maloya and ask them to determine, for each service, whether it detects a set of scenarios of daily living. This approach allows us to assess whether participants comprehend and anticipate the behavior of services written in Maloya.

Contributions. The contributions of this paper can be summarized as follows.

- We introduce professional caregivers to an end-user, domain-specific language, named Maloya and aimed to be a vehicle for them to understand the behavior of assistive services written in this language, and to best match their care-receivers' needs with appropriate services.
- We conduct a user study involving professional caregivers to assess the comprehensibility of Maloya in the context of realistic assistive services, applied to scenarios of daily living.
- This study reveals that participants successfully determine whether a service detects a scenario (success rate of 94%) and substantiate their answers with consistent explanations.

An outcome of this paper is that an end-user language such as Maloya can be an effective vehicle to empower caregivers to select appropriate services for their care-receivers, accurately anticipating the service behavior. This outcome is a step toward facilitating technology dissemination and adoption.

Outline. This paper is organized as follows. Section 2 discusses the related work, addressing assistive technologies, end-user development, and smart homes. Then, Section 3 gives a short introduction to Maloya, making this paper self-contained. Next, Section 4 presents our user study, its design and its findings. Section 5 discusses our findings and their limitations. Finally, Section 6 envisions the next step of our work, outlining how Maloya could enable caregivers to choose assistive services from a catalog. Section 7 concludes and outlines the future work.

2. Related Work

We first briefly review assistive technologies for aging. Then, we explore research works in end-user development. Finally, we review approaches to application development for smart homes, targeting either end-users or professional programmers.

2.1. Assistive technologies.

Exploring existing ATs for aging reveals that they are typically special purpose, addressing a specific need (*e.g.,* agenda reminder, pill dispenser, light path, fall detector) and offering limited customization (Gillespie, Best, & O'Neill, 2012). However, because of their narrow target, they may become ineffective or unusable as the user needs evolve with age-related changes. When ATs are multi-purpose, they are usually hosted by a general-purpose computing infrastructure, such as smartphones, tablets, computers, smart speakers, and smart homes. Assistive services take the form of applications that leverage the capabilities of the underlying infrastructure, such as sensing, actuating, user interaction, and networking (Consel, 2018). Examples of such services include agenda reminders, collaborative games, sleep monitor, and pedometer.

2.2. End-user development

End-User Development (EUD), sometimes called end-user programming, aims at empowering users to develop or modify applications (Paternò, 2013). To do so, EUD provides users with different programming approaches and models that make this activity amenable to non-programmers. For example, languages can be textual or visual, programming can be done by demonstration or example. Recent surveys examine a wide range of topics on EUD: touring EUD research topics (Bolmsten & Dittrich, 2011), reviewing the EUD research methods and user studies (Tetteroo & Markopoulos, 2015), examining EUD application domains (*e.g.,* web and mobile applications) (Paternò, 2013), and presenting tools supporting EUD (*e.g.,* programming environments, frameworks, spreadsheet) (Maceli, 2017).

One of the most widely known end-user programming language is Scratch (Resnick et al., 2009). This language provides visual notations, mixing text and icons, for programming. Because it is general purpose, Scratch does not provide users with domain-specific abstractions that could ease the task of designing and building programs in our target domain, namely assistive computing.

The need for domain-specific approaches has been gradually recognized, as illustrated by the domain of mashups. Mashups are tools for building web pages or web information portals, that aggregate information from different web sites and web services. An early EUD system for mashups called Marmite introduced a visual dataflow language, which was evaluated in an informal user study (Wong & Hong, 2007). Results were mixed in that only half of the users succeeded in building the required applications. A subsequent EUD system called Cocoa Buzz revealed that there exists a spectrum between end-user programming and end-user software configuration; Eagan and Stasko introduced an approach that addresses this spectrum, enabling complex configuration to be done by instantiating predefined domain abstractions using menus (Eagan & Stasko, 2008). A more recent proposal of EUD for mashups proposed to simplify EUD for mashups by narrowing the target even more. Specifically, it proposed EUD dedicated to mashups for the domain of scientific conferences for researchers (Soi, Daniel, & Casati, 2014).

Compared to the above approaches, ours is positioned between user programming and user configuration, within a specific domain. It explores a new path for end-user development in which users are destined to comprehend, select and configure, not only predefined domain abstractions, but also an open-ended list of services. To achieve this goal, we need the logic of services to be exposed to the users, without requiring them to be programmers. This approach can be achieved by a DSL, whose comprehensibility must be validated with a user study.

2.3. EUD for smart homes

The domain of Smart Home (SH) applications has been a major domain of study in the context of EUD for more than a decade. Early work on EUD for smart homes included such systems as iCAP, CAMP, and MAPS. Dey *et al.* produced a seminal result that showed by means of a user study that 95% of the smart-home services envisioned by users can be expressed as simple if-then rules (Dey, Sohn, Streng, & Kodama, 2006). Based on this result, they implemented a system called iCAP, which included a user interface for writing SH-related if-then rules, mixing icons and text. Rules only allowed for rudimentary temporal relations, *e.g.,* sequences and durations. Another approach, pursued by Truong, Huang, and Abowd (2004), allowed end-users to specify smart-home services by freely combining a small set of English words, relevant for a very narrow sub-domain, namely sound/video recording/playback applications. In this approach, services were specified using a system named CAMP, which automatically translated them to executable form. Truong *et al.* recognized that their approach did not scale for wider domains of services. Carmien and Fischer developed a system named MAPS that allowed caregivers to define interactive prompting services for users with cognitive disabilities, using a film scripting interface (Carmien & Fischer, 2008). The services were not context aware and were limited to prompting (*e.g.,* no activity monitoring). Recent research on EUD for smart homes (*e.g.,* AppsGate, Coutaz, Demeure, Caffiau, & Crowley, 2014; CCBL, Terrier, Demeure, & Caffiau, 2017a) suggests that if-then rules, as pioneered by iCap, are still the dominating paradigm in EUD for smart homes. This paradigm consists of two variants: trigger-action programming (TAP) and event-condition-action (ECA). TAP rules consist in actions triggered by a single event of the form "ON event THEN action" (*e.g.,* "when someone enters the house, turn on the heating"). Sometimes the trigger is a boolean condition similar to a state (e.g. "when I am talking on the phone, set the radio volume to low"). In this case, the action is triggered in fact by the event, when the condition becomes true. Some users also expect the action to be undone when the condition becomes false. Not surprisingly, this situation is a source of ambiguity, as noted by researchers working on TAP rules. ECA rules slightly extend TAP rules with a guarding condition; they are of the form "ON event IF condition THEN action".

The typical example of TAP rules is the widely known IFTTT system (Ovadia, 2014). IFTTT has been much used and studied, and has been shown to be accessible to general users, even when rules are extended to allow combining several triggers (Ur, McManus, Pak Yong Ho, & Littman, 2014). However, two user studies conducted by Huang and Cakmak revealed that end-users encounter difficulties when programming IFTTT rules that involve states and events because they are confusing these concepts (Huang & Cakmak, 2015a). AppsGate (Coutaz et al., 2014) proposed to address these difficulties by clearly distinguishing events from states in the rules. However, their approach limits expressiveness since they forbid a condition to combine several events and/or states. This limitation severely restricts the kind of assistive services that can be expressed where rich temporal relations between states and events are common.

Recently, Terrier *et al.* extended the ECA model with their EUD language called CCBL (Terrier et al., 2017a); they introduced a notion of state context, distinct from an event context. State and event contexts can be embedded in other state contexts via a cascading mechanism. Such an embedding entails that a state temporally includes its sub-states and/or sub-events. A preliminary user study shows that end-users write CCBL rules more easily than TAP rules, even when more complex rules are considered. However, no temporal relations other than the ones expressed as temporal inclusion can be defined between states. This limitation precludes sequences of states that are commonly expressed to detect user activities. For instance, one cannot express a go-to-bed routine as the user being in the bedroom (short) *after* being in the bathroom.

This limitation is partially relaxed in an extension of CCBL (Terrier, Demeure, & Caffiau, 2017b), which includes a subset of Allen temporal relations. Importantly, Allen's temporal logic (Allen, 1983) provides a comprehensive, semantically well-founded model, which covers all the 13 possible, mutually exclusive, temporal relations between two time intervals, such as precedence, inclusion, and overlapping. However, only a small subset of the Allen relations close to temporal inclusion are fitted into CCBL, and not in conformance with the semantics defined by Allen. Furthermore, no user study has been conducted on these extensions to determine whether the usability of CCBL carries over when adding temporal relations.

In contrast, by leveraging Maloya (Volanschi, Carteron, & Consel, 2018), our work includes all Allen relations relevant for the assistive computing domain. This DSL targets the particular domain of assistive computing with domain-specific concepts, and allows to capture realistic assistive services. The Maloya language has been validated in terms of expressiveness and its implementation in terms of performance. Specifically, its expressiveness allowed to encode a complete set of 53 assistive services, deployed in over 100 homes of seniors living alone, while requiring moderate resource consumption. However, no user study was conducted to assess the language with respect to caregivers (nor with developers). An issue yet to be studied is whether caregivers understand assistive services written in Maloya to the extent of building a mental model of these services and matching it against the needs of care-receivers. This understanding is a key to enable caregivers to accurately select and configure assistive services from a catalog. The goal of our work is to conduct a user study to assess whether caregivers understand services written Maloya.

2.4. Other programming models

Other programming technologies, either for ubiquitous computing in general, or for smart homes in particular, include general programming languages (GPL), complemented with specific middleware, complex event processing (CEP), finite state automata, and timed automata. More recently, a few more specific DSLs for professional developers have been used for recognizing human activities within smart homes.

Specifically, a DSL based on a temporal logic called MCL has been recently used for recognizing activities of daily living (ADLs) (Lago et al., 2017). MCL was initially designed for verifying program properties using model checking, but the ContextAct@A4H experiment showed that it could be successfully used for describing ADLs on logs of sensor events, using a combination of temporal logic formulas extended with regular expressions and recursion. In ContextAct@A4H, MCL has been exercised for monitoring ADLs on a 4-week log from a smart home equipped with a dense, but non-intrusive, sensor infrastructure. The results are convincing in terms of precision and recall of the recognized activities, but the formulation of ADLs in MCL is highly technical, as it requires extensive knowledge of temporal logic, and it is low level, as ADLs are expressed as sequences of events. Indeed, there is no notion of states in this programming model.

Allen (Volanschi, Serpette, et al., 2018) is a recent DSL specifically designed for developing applications based on binary sensors, in such domains as assistive services in a smart home. Allen addresses the same domain as Maloya, but at an increased level of generality. Thus, it includes the 13 Allen relations and other temporal operators, but is also extensible with new user-defined operators. Allen has been validated by expressing a complete set of assistive services for activity monitoring, and exercised on 10 sensor logs from real homes, each spanning 1 year. Unlike Maloya, Allen exclusively targets software developers of assistive service and thus offers a textual syntax similar to functional programming languages, that is not adequate for non-programmers. No user study was conducted to assess the usability of the language.

None of these programming languages or models are adequate for non-technical end users. Indeed, non technical users are not necessarily interested in learning a GPL; expertise in CEP and temporal logics is rare even among professional software developers; and coding SH applications using finite-state automata has been shown to be difficult for non-technical end-users (Terrier et al., 2017a).

Recently, Stefanidi, Korozi, Leonidis, and Antona (2018) introduced a new approach to programming smart home environments, based on a conversational interface. In their approach, end users engage in a textual conversation with a monitoring system called

ParlAmI to program smart home behaviors. Based on this conversation in natural language (English), ParlAmI automatically defines If-Then rules (belonging to the TAP and ECA programming models) for an underlying home monitoring system called LECTOR. The system further questions the user, if some details are missing, or if some user requirements are ambiguous. In an alternative implementation of the system, the conversational agent is embodied by a humanoid robot (Stefanidi et al., 2019). This approach is very promising for empowering end users to program smart-home automation rules with no explicit programming involved. However, it has only been applied and tested to define comfort-related automation rules. In comparison, our approach is less general and more focused on assistive applications. Instead of using general-purpose natural language and prompting the user to resolve inherent ambiguities, we use a restricted subset of natural language, which can be assigned an unambiguous meaning. Furthermore, in our work, we conduct a user study to check whether the intended target users – the professional caregivers – accurately understand the meaning of the rules with no preliminary training.

Summary and aim of our research

Caregivers are the most knowledgeable to configure ATs for detecting the activity of older adults, as they observe daily their habits, needs and preferences. In fact, it has been shown that the declarations given by older adults about their own needs differ from those given by caregivers, and are less linked to their sensorimotor and cognitive inabilities (Dupuy et al., 2015). Thus, the observations of caregivers about older adults are essential.

However, caregivers often do not have computer skills. This is why they need a simple end-user language to understand these ATs, without getting lost by a rich catalog of assistive services. Indeed, a catalog of customized services may provide a large choice of services with different versions, covering different needs of older adults and various home configurations. To address this major obstacle, we propose to introduce caregivers to an end-user language, namely Maloya, and use it as a vehicle for them to comprehend the logic of assistive services. More specifically, our goal is to conduct a study involving professional caregivers and evaluating their comprehension of assistive services written in Maloya with respect to practical scenarios. As such, this approach can pave the way to empowering caregivers to customize the assistive computing support of their care-receivers.

3. A Domain-Specific, End-User Language: Maloya

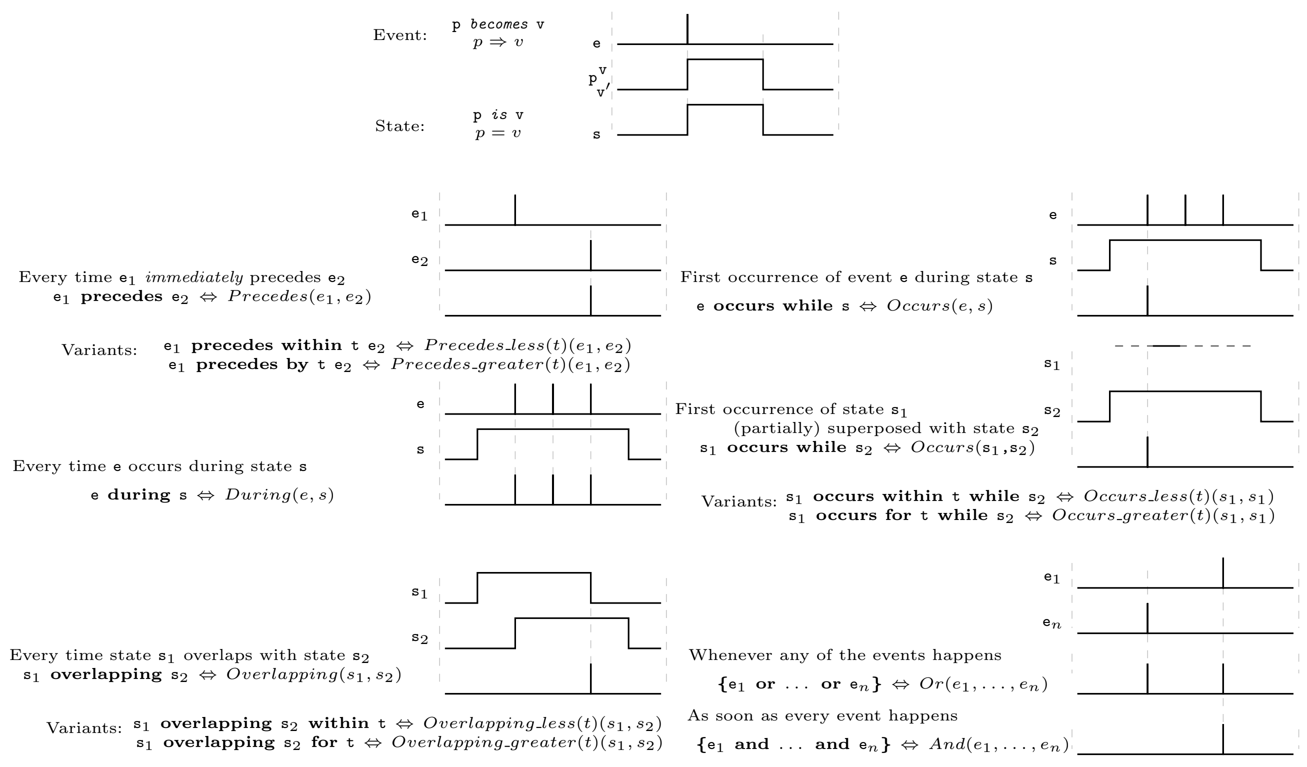
Because our user study revolves around the Maloya language, we first give an introduction to this language and present its main concepts (Volanschi, Carteron, & Consel, 2018). We then present a grammar for Maloya, which formalizes its syntax and outlines its scope. Finally, we present several scripts, illustrating the language. These same scripts will then be used in our user study, presented in Section 4.

3.1. Concepts of the Maloya language

Maloya is a DSL for defining contextual conditions for assistive services in a smart home. It also aims to foster collaboration between the various stakeholders of assistive services, such as software developers, aging experts, and caregivers. To do so, Maloya uses a restricted set of natural-language style syntax; it is text-based and features concepts dedicated to the home assistance domain. Building upon several user studies on end-user languages in the home-automation domain (Coutaz et al., 2014; Huang & Cakmak, 2015b), Maloya provides states and events as distinct concepts, which can only be combined in disciplined and meaningful ways, using typed operators. Operators include all the operators in the Allen logic (Allen, 1983) that are relevant for programming assistive services. These operators have been generalized in two respects: (1) to accept events and/or states, where applicable, and (2) to handle multiple occurrences of events and/or states. In constrast, the original relations in the Allen logic only hold between two states, lasting over two time intervals, and do not consider timepoint events.

The syntax and semantics of the Maloya language are defined in Figure 1, extracted from Carteron *et al.*'s original article. The graphical representation visualizes events and states, as well as how they are combined by operators. Both events and states are represented as boolean signals: boolean functions of time. In practice, time is discrete, and is given by standard Unix timestamps (number of milliseconds elapsed since some fixed date). Thus, states correspond to rectangle-shaped signals, and events to spike signals.

Syntactically, *events* are expressed as "p becomes v" where p is a sensor and v is a binary value, which can map to 'open' or 'closed' for a contact sensor, 'on' or 'off' for a smart plug associated to an electric appliance, and 'true' or 'false' for other sensors, such as presence



Event: p becomes v $p \Rightarrow v$
e
p v v′
State: p is v $p = v$
s
e1
e2
Every time e1 immediately precedes e2
e1 precedes e2 ⇔ Precedes(e1, e2)
Variants: e1 precedes within t e2 ⇔ Precedes_less(t)(e1, e2)
e1 precedes by t e2 ⇔ Precedes_greater(t)(e1, e2)
e
s
First occurrence of event e during state s
e occurs while s ⇔ Occurs(e, s)
e
s
Every time e occurs during state s
e during s ⇔ During(e, s)
s1
s2
First occurrence of state s1 (partially) superposed with state s2
s1 occurs while s2 ⇔ Occurs(s1,s2)
Variants: s1 occurs within t while s2 ⇔ Occurs_less(t)(s1, s1)
s1 occurs for t while s2 ⇔ Occurs_greater(t)(s1, s1)
s1
s2
Every time state s1 overlaps with state s2
s1 overlapping s2 ⇔ Overlapping(s1, s2)
Variants: s1 overlapping s2 within t ⇔ Overlapping_less(t)(s1, s2)
s1 overlapping s2 for t ⇔ Overlapping_greater(t)(s1, s2)
e1
en
Whenever any of the events happens
{e1 or ... or en} ⇔ Or(e1, ..., en)
As soon as every event happens
{e1 and ... and en} ⇔ And(e1, ..., en)

Figure 1. Informal syntax and semantics of Maloya (Volanschi, Carteron, & Consel, 2018)

detectors. *States* are expressed as "p is v", and cover the time interval between "p becomes v" and "p becomes v'", where v' is the negation of v.

Operators are used to express complex conditions, involving events and states. Although operators are applied to events and/or states, they always return events. Operators are typed, which means that every operator accepts only specific combinations of events or states as arguments. For instance, 'precedes' only applies to two events, while 'during' only applies to an event and a state. Among the 13 relations in the Allen logic, which cover all the possible orderings between two time intervals, Maloya excludes the Allen relations that require simultaneous events, such as 'starts' (the two intervals starting simultaneously) and 'meets' (one interval ending simultaneously with the start of the other). These relations have been excluded because events coming from independent sensors rarely occur simultaneously, given that in most systems, events are timestamped with a precision of milliseconds (or less). Thus, expressing such conditions is not useful for assistive applications. Moreover, in practice, simultaneous events can be serialized before being handled. As a result, Maloya defines operators corresponding to only 3 Allen relations ('precedes', 'during', and 'overlapping' (see Figure 1), knowing that the inverse relations ('follows', 'contains', 'overlapped by') are also covered by these 3 operators, by simply reversing the arguments (*e.g.,* "e_2 follows e_1" can be written as "e_1 precedes e_2"). To express time delays, some of these operators optionally take a duration: "precedes by/within t", which puts a lower, respectively upper, bound on the delay between two events, and "overlapping for/within t", which puts a lower/upper bound on the overlapping time between two states. There also is an 'occurs' operator, which has been added later on to the Allen logic (Ghallab & Alaoui, 1989) for existentially quantifying over a time interval: "e occurs while s" means that there exists at least one occurrence of event e during state s. When used between two states, "s_1 occurs while s_2", this operator also admits a lower/upper bound on the duration of the superposition between two states. Finally, complementing the operators originating in the Allen logic, there is the conjunction and disjunction between events, which produce an event when all of, respectively any of, the given events have occurred.

When building complex conditions, operators can be nested, provided that the typing constraints are satisfied for each operator. Thus, as operators always produce events, they can be nested in any operator position accepting an event. For instance, an operator can be nested as the first argument of an 'occurs' operator, but not as its second argument.

```
rule -> s | s "for" t | simple_event | complex_event

s -> ID "Time" | p "is" ("true" | "open" | "on" | "false" | "closed" | "off")

e -> simple_event | "(" complex_event ")"

simple_event ->
  p "becomes" ("true" | "open" | "on" | "false" | "closed" | "off")
| "BatteryLevel(" (ID | "Any") ")" "becomes" ("more" | "less") "than" INT

complex_event ->
  e "precedes" (("within" | "by") t)? e
| e "during" s
| s "overlapping" s (("within" | "for") t)?
| s "occurs" (("within" | "for") t)? "while" s
| e ("and" e)* "occurs" "while" s
| e ("or" e)* "occurs" "while" s

p -> ID | "Presence(" ID ")"

t -> INT ("seconds" | "minutes" | "hours")
```

Figure 2. Syntax of the Maloya language

3.2. The syntax of the Maloya language

The syntax of Maloya, informally expressed by the examples in Figure 1, can be formally defined by the grammar given in Figure 2. The notation of the grammar is standard: alternation is denoted by a vertical bar, keywords are quoted, terminals are in all capitals (ID, INT), a question mark suffixes a construct appearing 0 or 1 time, a Kleene star suffixes a construct appearing 0 or more times, and parentheses are used for grouping.

A complete Maloya rule is defined by non-terminal ‘rule’. States are defined by non-terminal ‘s’, and are either the name of a calendar slot (*e.g.,* “Night Time”), or a sensor name having a specific value. Events are defined by non-terminal ‘e’, and are either simple or complex, when produced by an operator. To avoid any ambiguity, while minimizing the number of parentheses, complex events must be parenthesized only when embedded in other complex events.

As can be seen, the grammar allows a term such as “Fridge is open for 5 minutes” to occur at the top level via the non-terminal ‘rule’. In turn, the ‘and’ and ‘or’ operators are only allowed to appear within an ‘occurs’ operator on a state, which explicits the fact that their meaning is subordinated to a time slot. Note also that the grammar enforces the typing discipline of each operator, so that, for example, a state is admitted as an argument only in suitable positions of some operators. For instance, operator ‘during’ requires an event as its first argument and a state as its second argument.

3.3. Example scripts

To illustrate the language with representative examples, Table 1 shows a subset of the assistive services on which Maloya had been validated in (Volanschi, Carteron, & Consel, 2018). Out of the 13 services detailed in the original article, we extracted all the services that could be of interest to caregivers. There were 10 such services, covering personal and home security, monitoring of daily activities, and monitoring technical aspects that could be fixed by caregivers. Three services were left out; they were exclusively intended for technicians: they concerned deep technical monitoring of the sensor infrastructures, such as detecting malfunctioning sensors or communication problems with the wireless network. For each service, the table gives the script in Maloya implementing the context logic of the service, that is, recognizing the situation monitored by the service. The action part of the services is left out, as it involves no complexity: actions are as simple as displaying an alert on a tablet installed in the home at a fixed position, or sending an SMS to some number.

In sum, Maloya offers a number of promising features for a user study in comprehensibility of assistive services because of its natural-language style syntax, its distinction between events and states, and its selection of the relevant Allen-temporal relations for the domain of home assistance.

Table 1. Ten different assistive services and their encoding as Maloya scripts.

#	Name	Description	Script in Maloya
1	Long inactivity	Detect if no movement in Bedroom since 24 hours	Presence(Bedroom) is false **for** 24 hours
2	Fridge opened	Detect if fridge remains open at least 5 minutes	Fridge is open **for** 5 minutes
3	Battery alert	Detect battery level of any sensor that become less than 5%	BatteryLevel(Any) becomes less than 5
4	Departure alert	Detect if entrance door is opened at least for 5 minutes during calendar night time	Door is open **occurs for** 5 minutes **while** Night time
5	Door alert	Detect if entrance door is opened at least for 5 minutes during which there is no presence in entrance	Door is open **occurs for** 5 minutes **while** Presence(Entrance) is false
6	Breakfast	Detect cupboard and coffeemaker opening (any order) during breakfast period	Cupboard becomes open **and** CoffeeMaker becomes on **occurs while** Breakfast Time
7	Dinner	Detect fridge opening and microwave starting (any order) during dinner period	Fridge becomes open **and** Microwave becomes on **occurs while** Dinner Time
8	Wake-up	Detect end of presence in bedroom, then begin of presence in kitchen in the following 10 minutes, during go-to-bed period	(Presence(Bedroom) becomes false **precedes within** 10 minutes Presence(Kitchen) becomes true) **occurs while** Wakeup Time
9	Go to bed	Detect end of presence in bathroom, then begin of presence in bedroom in the following 10 minutes, during go-to -bed period	(Presence(Bathroom) becomes false **precedes within** 10 minutes Presence(Bedroom) becomes true) **occurs while** Gotobed Time
10	Lunch reheat	Detect freezer opening, then stove use in the following 10 minutes, or freezer opening during stove use; all during lunch period	(Freezer becomes open **precedes within** 10 minutes Stove becomes on) **or** (Freezer becomes open **occurs while** Stove is on) **occurs while** Lunch Time

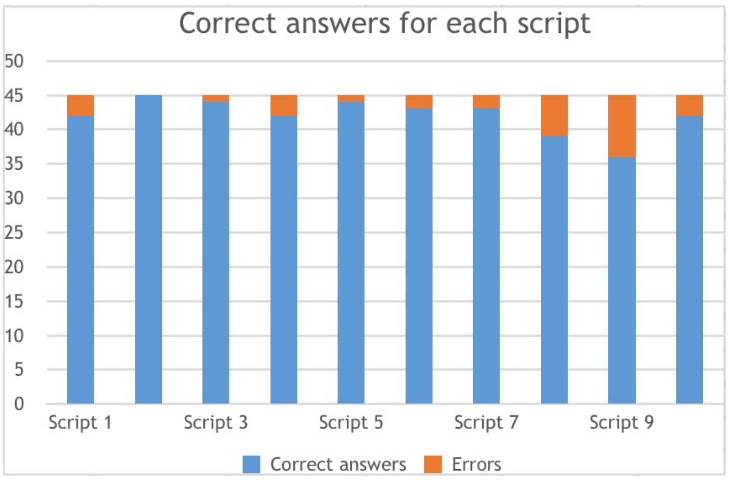


Figure 3. Number of correct answers for each script.

4. User Study

We conducted a study to assess the comprehensibility of the Maloya language by professional caregivers through scenarios of daily life activities of older adults.

4.1. Study design

Nine professional caregivers participated to this study. Participants consisted of 9 women, aged from 22 to 40 years (median = 26). Among the participants, there were five professionals specifically trained to assist caregivers, two occupational therapists, one nurse, and one house keeper. Their level of education ranged from vocational diploma to masters degree. Participants were individually tested in a quiet room.

4.1.1. Material of quantitative study

The 10 scripts of assistive services written in Maloya, listed in Table 1, have been assessed. Each participant had to determine whether a script allowed specific scenarios to be detected. The order of the scripts submitted to the participants was counterbalanced. Five scenarios were evaluated by participants for each script. Thus, a total of 50 scenarios were evaluated, consisting of positive and negative situations. For each scenario, participants had to determine whether the script detected it, and if so, explain why. In doing so, we were able to check that participants provided the correct conditions for a script to detect the scenario. Participants received no training prior to being tested; they had an unbiased view toward our language. The collected data were the number of correct answers and the response time for each scenario. Table 4.1.1 presents an example of scenario for each of the 10 scripts. Participants were allowed one hour for evaluating the 50 scenarios.

4.1.2. Material of qualitative study

At the end of the study, participants answered six questions about their perception of this end-user language for caregivers: two open questions and four questions using a 5-point Likert scale (1 meaning "not agree at all" and 5 meaning "totally agree"). The collected data were the participants' answers to the six questions.

4.2. Findings of the study

4.2.1. Quantitative study

The scores (total number of correct answers) for each participant are presented in Table 3. The scores for each script are displayed as a histogram in Figure 3. The results clearly show that the end-user language is comprehensible by caregivers, and perceived as such by them. Indeed, caregivers answered correctly to 94% of the scenarios they were submitted. For each

Table 2. Ten scripts of assistive services written in Maloya and examples of scenarios.

#	Name	Script in Maloya	Example of Scenario	Activity detected
1	Long inactivity	Presence(Bedroom) is false **for** 24 hours	Ms. Hill wakes up in the middle of the night to go to the bathroom. But when she comes back from the toilet, she falls into the living room, unable to get back on her feet. The next morning, her nurse sees her on the floor, picks her up, and takes her back to her bed.	No
2	Fridge opened	Fridge is open **for** 5 minutes	Mr. Smith comes back at home after shopping. He puts the cheese in his fridge. The phone rings, interrupting him in his task. He rushes to the phone without taking the time to close the fridge. The phone call takes less than a minute. He hangs up and then closes the fridge.	No
3	Battery alert	BatteryLevel(Any) becomes less than 5	The battery level of the tablet shows 4% but Ms. Jones puts it to charge immediately.	Yes
4	Departure alert	Door is open **occurs for** 5 minutes **while** Night time	Mr. William comes back home at 8:40 pm after shopping. He opens the door without closing it to be able to go back and forth between the car and his home. The storage takes more than half an hour.	Yes
5	Door alert	Door is open **occurs for** 5 minutes **while** Presence(Entrance) is false	Ms. Brown comes back home after going to the doctor. His dog welcomes her cheerfully, jumping on her knees; she plays with him in the entrance for ten minutes, leaving the door open.	No
6	Breakfast	Cupboard becomes open **and** CoffeeMaker becomes on **occurs while** Breakfast Time	Mr. Taylor wakes up. He gets dressed and takes out his dog. He comes back and enters the kitchen at 8:30 am. He opens the cupboard to give a treat to his dog. Then, He turns on the coffee machine to make a coffee.	Yes
7	Dinner	Fridge becomes open **and** Microwave becomes on **occurs while** Dinner Time	It is 6.30 pm. Ms. Evans is already hungry. She opens her fridge, takes a ready-made dish and turns on the microwave to warm it up.	No
8	Wake-up	(Presence(Bedroom) becomes false **precedes within** 10 minutes Presence(Kitchen) becomes true) **occurs while** Wakeup Time	It is 7:30 am. Mr. Wilson wakes slowly up, He stretches in his bed. He turns on the radio on his bedside table, listens to the news and the weather forecast. It is 7:50 am. He gets up and prepares his breakfast in the kitchen.	Yes
9	Go to bed	(Presence(Bathroom) becomes false **precedes within** 10 minutes Presence(Bedroom) becomes true) **occurs while** Gotobed Time	It is 21:40 pm. Ms. Roberts goes to her bedroom to get into her pyjamas, before going to the bathroom to wash up. Then, she goes into the living room to turn off the TV and closes the shutters. She takes her dog out for 10 minutes and goes to bed.	No
10	Lunch reheat	(Freezer becomes open **precedes within** 10 minutes Stove becomes on) **or** (Freezer becomes open **occurs while** Stove is on) **occurs while** Lunch Time	Mr. Thomas comes back home after his appointment with the doctor. He waited a very long time there and is now hungry. It is 12:55 pm. He opens his fridge, takes a ready-made dish and turns on the stove to preheat it. Ten minutes later, he puts the ready-made dish into the stove.	Yes

Table 3. Scores for each participant (maximal score = 50).

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
48	45	44	49	50	45	48	44	48

Table 4. Times (in minutes) for giving the answers for each script, including all its 5 scenarios.

Script #				**Participants**						**Average**
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	
Script 1	5	10	5	4	1	3	2	3	2	3,9
Script 2	4	2	3	2	2	3	5	2	2	2,8
Script 3	3	2	2	5	1	2	3	2	2	2,4
Script 4	3	2	8	3	2	2	5	2	2	3,2
Script 5	5	4	5	4	6	3	7	4	2	4,4
Script 6	3	2	9	3	2	4	4	4	2	3,7
Script 7	4	4	6	2	2	4	3	6	2	3,7
Script 8	5	4	5	3	2	6	5	5	2	4,1
Script 9	6	8	10	3	3	5	5	3	9	5,8
Script 10	6	9	5	3	4	9	5	5	4	5,6
Total	44	47	58	32	25	41	44	36	29	39,6
Average	4,4	4,7	5,8	3,2	2,5	4,1	4,4	3,6	2,9	4,0

correct answer, participants provided the correct conditions of a script to detect the target activity.

The response times for each script are given in Table 4. Each time is represented in minutes and includes the answers for all 5 scenarios associated to the script. Scripts are numbered 1 to 10 (first column), and participants, named P1 to P10, correspond to the subsequent columns. Average response times for each script are given in the last column. Finally, the last two lines in the table give the total response time and the average response time for each participant, when all 10 scripts are considered. For each participant, the shaded cell of the column represents the first script submitted to them; it changes across participants to counterbalance the script order.

The average response time across all participants and all scenarios was 4.0 minutes (see the last column of the bottom line). We notice that the majority of participants (8/9) answered more slowly for their first script (7.1 minutes on average, which is 3 minutes slower than the overall mean response time), regardless of the script, since their order changed for each participant. This result suggests the need of an introductory presentation of the language. We also note that Script 10 (Lunch Reheat) required longer response times (5.6 minutes on average, which is 1.5 minutes slower than the overall mean response time). This situation can be explained by the fact that this script has two alternatives to carry out the activity. As a result, it requires more cognitive resources, and thus more time for the participants to produce an answer. Despite these alternatives, participants correctly understood the script since it generated as many correct answers as the others. In fact, the scripts that caused the most errors are Scripts 8 (Wake-up) and 9 (Go to bed). These two scripts include the notion of duration (less than 10 minutes to change the room), which creates a level of confusion in some participants. Another potential source of confusion might be the room changes included in some scenarios associated to these scripts.

4.2.2. Qualitative study

While at the beginning of the test, most participants were questioning their comprehension of the scripts, at the end, they felt that they had well understood the ten scripts written in Maloya (mean = 4). They appreciated that the scripts were written in a natural-language style without complicated symbols. They also reported that Maloya adequately meets the needs of both the older adults and the caregivers (mean = 4). Indeed, the participants emphasized the importance of the flexibility of the language to account for the specificities of each older adult, and they noted that Maloya provided the required flexibility. They estimated that Maloya should allow to cover a large set of daily life activities, matching the needs of older adults. They estimated that all caregivers could easily understand this language (mean = 4), but they should initially receive a short training (mean = 4). This result seems consistent with the longer response times needed for the first script and argues for the introduction of some training to familiarize caregivers with the end-user language.

Some participants found that two expressions of Maloya were *unusual* ("Fridge becomes open" and "Presence(Bedroom) is false"), even though they could understand their meaning. Moreover, a script including a time slot as a condition to meet seemed inadequate for some participants because they reported that older adults can occasionally change their routine to eat earlier or later, for example. In this situation, the older adult will indeed receive a notification asking whether he/she ate. Such notifications, if repeated, should suggest that the corresponding changes in the user routines are not occasional and should require the caregiver's attention. When a user routine does not apply anymore, the caregiver can adapt the configuration of relevant assistive services. For example, an older adult may subscribe a service that delivers meals, requiring the microwave to be used to monitor meal preparation, instead of the stove. For another example, if an older adult changes his lunch habits, eating home or out, depending on social opportunities, it may be preferable to deactivate any service that monitors this activity.

5. Discussion

The results of our user study show that we met our aim to empower caregivers to customize the assistive support of older adults using an end-user language. This language allows caregivers to understand a catalog of services, select the ones that are appropriate for their care-receiver, and determine parameter values for the selected services. Specifically, Maloya is well understood by caregivers, who produce 94% of correct answers to our test. Moreover, we checked that each correct answer was not produced ramdomly because we asked the caregivers to explain the conditions under which a script detected a scenario. From a qualitative perspective, the results are also positive: caregivers found that Maloya was easily understandable, especially because it uses a natural-language style. Furthermore, they valued the flexibility that the language provides by allowing the detection of an activity to be customized with respect to the situation of each individual; they noticed that this capability covers a large set of older adults needs. As a consequence, Maloya should give caregivers the confidence to understand available services and anticipate their behavior, allowing them to support their care-receivers during service deployment.

While participants perceived the end-user language positively, they found that users needed to receive a short initial training. This finding is confirmed by the results of our quantitative study where our participants were slower to produce an answer for the first script, regardless of which script was picked first. Participants seem to need a startup time to get accustomed to the language; an initial training could address that need.

The understanding of Maloya has been tested over a wide range of real context-aware services, revolving around the key dimensions of daily activities, home/user security and simple maintenance issues. As such, it is a promising vehicle to provide assistive support to older adults that is customized with respect to the expert knowledge of caregivers.

Our study has a number of limitations. First, the sample of participants is small: 9 participants. However, all of them were professionals with expertise in the domain of aging. Second, four participants knew the HomeAssist platform before taking our test. Thus, this situation may have facilitated their understanding of the scripts. Third, we assessed the comprehensibility of the end-user language with only ten scripts. Nevertheless, these scripts were representative of the catalog of existing services and their comprehensibility was assessed with fifty scenarios.

A more general limitation of our study is that it only covers the comprehension of scripts expressed in our language by end users, and does not address the production of such scripts. Nevertheless, the comprehension part is a useful intermediate step towards empowering caregivers, as explained previously. More details about the potentials of this intermediate step are given in the next section. Finally, our user study did not include interviews of older adults. Addressing the production of scripts by caregivers, and gathering feedback from older adults are promising avenues for future work.

6. Selection-Based Customization

In this section, we sketch a method for customizing assistive services, enabled by the Maloya language. The method, applicable by caregivers, would consist in selecting a set of services in a catalog that match the profile and needs of a care-receiver. More precisely, we show by means of examples, how Maloya can support the different forms of service selection which were used in the HomeAssist project. Recall from the Introduction that, in that project, each service of the catalog was presented to the participants to determine whether it was (1) useful, provided some parameterization, (2) useful, but required some adaptation, or (3) inappropriate.

6.1. Recognizing useful services needing parameterization

In the simplest case, the caregiver may directly recognize a service that is useful without any parameterization. For example, the "Fridge opened" service is probably useful for most care-receivers as it addresses a common situation, and uses a reasonable delay parameter. Note that the primary goal of this assistive service is to prevent food from degrading and cause health problems; a secondary goal could be to avoid wasting electricity.

Other services may need some parameterization to address the needs of a care-receiver. For instance, consider the "Long inactivity" service, detecting an absence from the bedroom during an entire day and night. While this service may be suitable for most users, consider the case of Ms. Hill, who has a condition keeping her in bed virtually all the time. She only gets up a few times a day for short activities around the house, risking a fall. Such a situation is covered by the sample scenario for script 1 in Table 4.1.1. To address this risk, the caregiver may decide to customize the service by shortening the delay during which she is not in her bedroom to an hour or less, say. This parameterization should result in a service raising alerts that are more relevant to Ms. Hill's situation.

Parameterization may also concern more complex service logic, as illustrated by the "Departure alert" service. This service detects a night wandering episode when the entrance door remains open for a long period of time during the night. This rule is more complex in that it combines a contact sensor (on the entrance door), a time interval (to delimit the night), and a duration (5 minutes). Each of these temporal dimensions may need to be parameterized for specific cases. Consider, for instance, the case of Mr. William, whose Night interval is between 9PM and 6AM, declaring in effect never to go out during that time. However, his son regularly comes late in the evening to bring his groceries while he sleeps, and usually leaves the door open for more than 5 minutes while unloading the car. This situation was covered by the sample scenario of script 5 in Table 4.1.1, involving Mr William (jr). Knowing this situation, Mr. William's caregiver may recognize that the declared Night interval has to be shortened, starting later (*e.g.,* say 12AM). Otherwise, false alerts may be raised that can wake him up, disturbing his sleep, and hence doing more harm than good.

In order to take such decisions, the caregiver must correctly understand the impact of a time parameter on the context logic, such as a minimal or maximal duration, or a time interval. The overwhelming correct answers of our study participants matching scripts with scenarios are encouraging in this regard.

6.2. Recognizing useful services needing adaptation

Some assistive services require some adaptation instead of, or in addition to, parameterization. Let us consider once again the "Fridge opened" service. As explained above, its parameterization should be suitable for most users, but the action triggered by this service may need to be customized. For instance, the service is useful for Mr. Smith, but the default action of issuing an alert notification on the home tablet is inappropriate because it can wake up Mr. Smith during the night or a nap. Knowing this situation, the caregiver may decide to replace this alert by a more discrete text message to Mr. Smith. This change could actually be performed by the caregiver in the Maloya-written service. Alternatively, the catalog could consist of two versions of this service: one triggering an alert and the other sending a text message.

Such adaptation requires an extensive knowledge of the care-receiver routines and preferences to refine the behavior of assistive services to that level of detail.

6.3. Recognizing inappropriate services

Deciding whether a service is appropriate may require a shallow or deep understanding of assistive service logic. As a simple example, the "Lunch reheat" service would be useless for a person who usually goes out for lunch. In this case, to reject this service, the caregiver only needs to understand that its goal is to detect a lunch preparation routine, unconditionally.

A deeper examination is needed to discard the same service for a person not equipped with a freezer. In this case, the catalog may offer a more appropriate version of the service where a refrigerator is used instead of a freezer.

Finally, an even deeper examination of the "Go to bed" service logic is necessary to recognize that it is inappropriate for Ms. Roberts. This service detects the sleep preparation routine as a room transition between the bathroom and the bedroom. Discarding this service requires her caregiver to take into account the fact that this service logic does not apply to

Ms. Roberts's routine because she usually does other activities, of variable length, on her way from the bathroom to the bedroom, such as watching TV, closing shutters, *etc.* This situation was covered by the sample scenario for script 9.

Similarly, deciding whether a service *is* appropriate for a care-receiver also requires a detailed understanding of its logic. For instance, the "Door alert" service detects an entrance door left open *and* unattended for some time. This logic is still applicable to Mr. Brown, who is usually welcomed by his dog at the entrance, which causes the door to be left open for long, but *not* unattended. This situation was covered by the sample scenario for script 5.

Again, the largely correct answers for the various scenarios and scripts are encouraging for the perspective of relying on caregivers to recognize and discard services.

7. Conclusions and Future Work

Our study shows that Maloya can be disseminated to caregivers as a tool that empowers them to customize the assistive computing support of older adults. The range of assistive services examined in our work builds upon the services developed for the HomeAssist platform. This platform and its catalog of services have already been deployed and assessed in a real life setting; the results have showed a positive experience of users in terms of effectiveness, efficiency, acceptability, self-determination dimensions (autonomy, self-regulation, empowerment, and self-realization), and user experience (see Consel et al., 2017; Dupuy, Froger, Consel, & Sauzéon, 2017).

This paper has shown that Maloya is a pivotal tool to bridge the gap between caregivers and an assistive computing platform because of its ability to make assistive service comprehensible to caregivers.

In the future, we plan to further this line of work on Maloya and develop an authoring environment for caregivers, not only to configure services, but also to define them. This environment could use textual and graphical elements to enhance usability and guide the service developer. This work would need to be assessed by a usability study.

Another line of work may include extending the textual syntax of Maloya with more operators that are useful in the target domain, on a by-need basis, in cooperation with caregivers. It is likely that, at some point, when more and more operators are added, the comprehension of the language tends to decrease. Exploring these limits should contribute to strike a balance between expressivity and comprehensibility, without sacrificing the empowerment of the caregivers.

References

Alegre, U., Augusto, J. C., & Clark, T. (2016). Engineering context-aware systems and applications: A survey. *Journal of Systems and Software*, *117*, 55–83.

Allen, J. F. (1983, November). Maintaining knowledge about temporal intervals. *Commun. ACM*, *26*(11), 832–843.

Baecker, R. M., Moffatt, K., & Massimi, M. (2012). Technologies for aging gracefully. *interactions*, *19*(3), 32–36.

Bertran, B., Bruneau, J., Cassou, D., Loriant, N., Balland, E., & Consel, C. (2014). Diasuite: A tool suite to develop sense/compute/control applications. *Science of Computer Programming*, *79*, 39–51.

Bolmsten, J., & Dittrich, Y. (2011). Infrastructuring when you don't–end-user development and organizational infrastructure. In *International symposium on end user development* (pp. 139–154).

Brush, A., Lee, B., Mahajan, R., Agarwal, S., Saroiu, S., & Dixon, C. (2011). Home automation in the wild: challenges and opportunities. In *proceedings of the sigchi conference on human factors in computing systems* (pp. 2115–2124).

Carmien, S. P., & Fischer, G. (2008). Design, adoption, and assessment of a socio-technical environment supporting independence for persons with cognitive disabilities. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 597–606).

Cassou, D., Balland, E., Consel, C., & Lawall, J. (2011). Leveraging software architectures to guide and verify the development of sense/compute/control applications. In *Proceedings of the 33rd international conference on software engineering* (pp. 431–440).

Cassou, D., Bruneau, J., Consel, C., & Balland, E. (2012). Toward a tool-based development methodology for pervasive computing applications. *IEEE Transactions on Software Engineering*, *38*(6), 1445–1463.

Chauhan, S., Patel, P., Sureka, A., Delicato, F. C., & Chaudhary, S. (2016). Iotsuite: a framework to design, implement, and deploy iot applications: demonstration abstract. In *Proceedings of the 15th international conference on information processing in sensor networks* (p. 37).

Consel, C. (2018). Assistive computing: A human-centered approach to developing computing support for cognition. In *2018 ieee/acm 40th international conference on software engineering: Software engineering in society (icse-seis)* (pp. 23–32).

Consel, C., Dupuy, L., & Sauzéon, H. (2017). Homeassist: An assisted living platform for aging in place based on an interdisciplinary approach. In *International conference on applied human factors and ergonomics* (pp. 129–140).

Coutaz, J., Demeure, A., Caffiau, S., & Crowley, J. L. (2014). Early lessons from the development of spok, an end-user development environment for smart homes. In *Proceedings of the 2014 acm international joint conference on pervasive and ubiquitous computing: Adjunct publication* (pp. 895–902).

Czaja, S. J., Rogers, W. A., Fisk, A. D., Charness, N., & Sharit, J. (2009). *Designing for older adults: Principles and creative human factors approaches.* CRC press.

Dawadi, P. N., Cook, D. J., & Schmitter-Edgecombe, M. (2013). Automated cognitive health assessment using smart home monitoring of complex tasks. *IEEE transactions on systems, man, and cybernetics: systems*, *43*(6), 1302–1313.

Dey, A. K., Sohn, T., Streng, S., & Kodama, J. (2006). icap: Interactive prototyping of context-aware applications. In *International conference on pervasive computing* (pp. 254–271).

Dupuy, L., Froger, C., Consel, C., & Sauzéon, H. (2017, September). Everyday Functioning Benefits from an Assisted Living Platform amongst Frail Older Adults and Their Caregivers. *Frontiers in Aging Neuroscience*, *9*, 1-12. Retrieved from `https://hal.inria.fr/hal-01597680`

Dupuy, L., Sauzéon, H., & Consel, C. (2015). Perceived needs for assistive technologies in older adults and their caregivers. In *Womencourage 15'.*

Eagan, J. R., & Stasko, J. T. (2008). The buzz: supporting user tailorability in awareness applications. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 1729–1738).

Ghallab, M., & Alaoui, A. M. (1989). Managing efficiently temporal relations through indexed spanning trees. In *Proceedings of the 11th international joint conference on artificial intelligence - volume 2* (pp. 1297–1303). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from `http://dl.acm.org/citation.cfm?id=1623891.1623963`

Gillespie, A., Best, C., & O'Neill, B. (2012). Cognitive function and assistive technology for cognition: A systematic review. *Journal of the International Neuropsychological Society*, *18*(01), 1–19.

Huang, J., & Cakmak, M. (2015a). Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing* (pp. 215–225).

Huang, J., & Cakmak, M. (2015b). Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing* (pp. 215–225). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2750858.2805830`

Kaye, J. A., Maxwell, S. A., Mattek, N., Hayes, T. L., Dodge, H., Pavel, M., ... Zitzelberger, T. A. (2011). Intelligent systems for assessing aging changes: home-based, unobtrusive, and continuous assessment of aging. *Journals of Gerontology Series B: Psychological Sciences and Social Sciences*, *66*(suppl_1), i180–i190.

Lago, P., Lang, F., Roncancio, C., Jiménez-Guarín, C., Mateescu, R., & Bonnefond, N. (2017). The contextact@ a4h real-life dataset of daily-living activities. In *International and interdisciplinary conference on modeling and using context* (pp. 175–188).

Maceli, M. G. (2017). Tools of the trade: A survey of technologies in end-user development literature. In *International symposium on end user development* (pp. 49–65).

Mihailidis, A., & Fernie, G. R. (2002). Context-aware assistive devices for older adults with dementia. *Gerontechnology*, *2*(2), 173–188.

Nehmer, J., Becker, M., Karshmer, A., & Lamm, R. (2006). Living assistance systems: an ambient intelligence approach. In *Proceedings of the 28th international conference on software engineering* (pp. 43–50).

Ovadia, S. (2014). Automate the internet with "if this then that"(ifttt). *Behavioral & social sciences librarian*, *33*(4), 208–211.

Paternò, F. (2013). End user development: Survey of an emerging field for empowering people. *ISRN Software Engineering*, *2013*.

Rashidi, P., & Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. *IEEE journal of biomedical and health informatics*, *17*(3), 579–590.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... others (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.

Salber, D., Dey, A. K., & Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 434–441).

Schulz, R., Wahl, H.-W., Matthews, J. T., De Vito Dabbs, A., Beach, S. R., & Czaja, S. J. (2014). Advancing the aging and technology agenda in gerontology. *The Gerontologist*, *55*(5), 724–734.

Soi, S., Daniel, F., & Casati, F. (2014). Conceptual development of custom, domain-specific mashup platforms. *ACM Transactions on the Web (TWEB)*, *8*(3), 14.

Stefanidi, E., Foukarakis, M., Arampatzis, D., Korozi, M., Leonidis, A., & Antona, M. (2019). Parlami: a multimodal approach for programming intelligent environments. *Technologies*, *7*(1), 11.

Stefanidi, E., Korozi, M., Leonidis, A., & Antona, M. (2018). Programming intelligent environments in natural language: an extensible interactive approach. In *Proceedings of the 11th pervasive technologies related to assistive environments conference* (pp. 50–57).

Tang, P., & Venables, T. (2000). 'smart'homes and telecare for independent living. *Journal of Telemedicine and Telecare*, *6*(1), 8–14.

Terrier, L., Demeure, A., & Caffiau, S. (2017a). Ccbl: A language for better supporting context centered programming in the smart home. *Proceedings of the ACM on Human-Computer Interaction*, *1*(1), 14.

Terrier, L., Demeure, A., & Caffiau, S. (2017b). Ccbl: A new language for end user development in the smart homes. *IS-EUD 2017*, 82.

Tetteroo, D., & Markopoulos, P. (2015). A review of research methods in end user development. In *International symposium on end user development* (pp. 58–75).

Truong, K. N., Huang, E. M., & Abowd, G. D. (2004). Camp: A magnetic poetry interface for end-user programming of capture applications for the home. In *International conference on ubiquitous computing* (pp. 143–160).

Ur, B., McManus, E., Pak Yong Ho, M., & Littman, M. L. (2014). Practical trigger-action programming in the smart home. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 803–812).

Volanschi, N., Carteron, A., & Consel, C. (2018). A domain-specific approach to unifying the many dimensions of context-aware home service development. In *2018 ieee smartworld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (smartworld/scalcom/uic/atc/cbdcom/iop/sci)* (pp. 480–489).

Volanschi, N., Serpette, B., Carteron, A., & Consel, C. (2018). A language for online state processing of binary sensors, applied to ambient assisted living. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *2*(4), 192.

Wong, J., & Hong, J. I. (2007). Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 1435–1444).

Short biographies

Stéphanie Giraud is a cognitive ergonomist with a Ph.D. in cognitive and ergonomic psychology whose interests lie in Human-Computer Interaction, especially the usability of computer interfaces for impaired people. She is a teacher-researcher in the laboratory of clinical, cognitive, and social anthropology and psychology at the University of Côte d'Azur.

Nic Volanschi is an associate researcher at Inria Bordeaux. His research concerns using programming language technologies for automating various aspects of software engineering, such as: IoT applications development (Ambient Assisted Living, Smart cities), software maintenance (migration, mass refactoring), and software development (pattern matching). His research resulted in various domain-specific languages.

Charles Consel is a Professor of Computer Science at Bordeaux INP (School of Engineering). He served on the faculty of Yale University, Oregon Graduate Institute and University of Rennes. His research contributions include assistive computing, pervasive computing, software engineering, and HCI with 100+ articles in major journals and conferences.