Assistive Computing

A Human-Centered Approach To Developing Computing Support for Cognition

Charles Consel
Bordeaux Institute of Technology / Inria
Talence, France
Charles.Consel@inria.fr

ABSTRACT

The growing population of cognitively impaired individuals calls for the emergence of a research area dedicated to developing computing systems that address their needs. The nature of this research area requires to bridge the many disciplines needed to develop human-centered, assistive computing systems. Such bridging may seem unattainable considering the conceptual and practical gaps between the related disciplines and the challenges of propagating human-related concerns throughout the many stages of the development process of assistive technologies. As a consequence, existing assistive technologies lack a proper needs analysis; their development is often driven by technology concerns, resulting in ill-designed and stereotype-biased systems; and, most of them are not tested for their effectiveness in assisting users.

In this paper, we propose a systematic exploration of this vast challenge. First, we define *Assistive Computing* as a research area and propose key principles to drive its study. Then, we introduce a tool-based methodology dedicated to developing assistive computing support, integrating a range of disciplines from human-related sciences to computer science. This methodology is purposefully pragmatic in that it leverages, aggregates and revisits numerous research results, concretizing it with a range of examples.

More generally, our goal is *i*) to provide a framework to conduct research in the area of Assistive Computing and *ii*) to identify the necessary bridges between disciplines to account for all the dimensions of such systems.

KEYWORDS

Interdisciplinary research, Assistive computing support, Cognitive impairment, User-centered software development

ACM Reference Format:
Charles Consel. 2018. Assistive Computing: A Human-Centered Approach To Developing Computing Support for Cognition. In *ICSE-SEIS'18: 40th International Conference on Software Engineering: Software Track, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 10 pages.
https://doi.org/10.1145/3183428.3183431

ICSE-SEIS'18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5661-9/18/05…$15.00
https://doi.org/10.1145/3183428.3183431

1 INTRODUCTION

A large and growing population is cognitively impaired, whether because of neurodegenerative causes (aging, Alzheimer disease, Parkinson disease, *etc.*), neurodevelopmental causes (autism, intellectual disability, Down syndrome, *etc.*), or accidental causes (traumatic brain injury, stroke, *etc.*). Specifically, individuals with mental disability represent about 30% of the general population of individuals with disability in the United States [7], for example. This population critically needs assistive support in their daily activities to gain or maintain autonomy, and improve well-being.

Assisting users with computing systems in their daily activities is a promising direction that has never been so much within our reach. Indeed, devices offer ever richer functionalities, covering an expanding range of purposes, from silo-based devices (*e.g.,* pill-box) to multi-purpose ones (*e.g.,* smartphone). Technology classifications are being proposed to facilitate matching user needs with technologies. Researchers in human factors are defining guidelines to design assistive technologies [21]. So, despite these advances, why is it that so few assistive technologies are being adopted by users that need support the most for their daily activities?

A number of reasons can explain this paradoxical situation, to name a few [21]: lack of needs analysis, technology-centered design resulting in ill-designed systems, user stereotypes, insufficient evaluation studies in the wild, lack of evidence-based effects.

These shortcomings emphasize the fact that research and development in assistive technologies are mainly conducted in isolation from other disciplines. This situation becomes strikingly clear when human-related disciplines (*e.g.,* health, psychology, social work) evaluate the effects of existing assistive computing systems using their own criteria and methodologies [5]; not surprisingly, the outcomes are often disappointing. This problem is exhibited at a large scale by Martin *et al.* in their literature review where they could not identify studies that support or refute the use of smart home technologies within health and social care, which is significant for practitioners and health consumers [27]. As another illustration, Stevenson *et al.* conducted a large randomized controlled trial of 2,600 participants with social care needs, who were equipped with a range of devices (*e.g.,* fall detector, smoke detector, bed occupancy sensor) and subscribed to a telecare service [39]. This clinical trial showed that this extensive assistive support did not significantly alter rates of health or social care service use or mortality over 12 months. Similar outcomes with assistive technologies for users with autism are reported in the literature [19].

As can be seen, assisting users via a computing system remains a research topic that is vastly unexplored. To delineate the scope of assistive computing, we formulate the following key challenges.

- Can an assistive computing system be designed to be interwoven in the daily life of users with cognitive impairments and support their autonomy?
- Can it produce evidence-based improvements on autonomy and health (social, cognitive, and/or physical)?
- Can it be accepted and adopted over the long term?
- Can it facilitate inclusion of users with cognitive impairments in mainstream environments (*e.g.*, school, home, work)?

Let us exemplify these challenges. *Improving autonomy*: a system that accurately tracks the daily activities of an older adult at home (*e.g.*, waking up, preparing meals, bathing) and reminds them only when they are missed. *Improving health*: a system of sleep coaching that is based on a monitoring system, user's preferences and goals, and a professional's customized setting. *Facilitating inclusion in mainstream environments*: a monitoring system that measures the emotional level of a user with autism and delivers the appropriate intervention to regulate the emotion.

Addressing these examples of assistive goals and the underlying research challenges should lead to a breakthrough in the realm of computing and beyond. This work requires to combine computer science with a range of disciplines, including health, psychology, geriatrics, human factors, and social work.

Outline. We introduce the area of *Assistive Computing* as an interdisciplinary research avenue; it addresses the key challenges of developing computing support to assist users with cognitive impairments (Section 2). We propose principles that should organize and structure research efforts on Assistive Computing (Section 3). These principles provide a foundation for a tool-based methodology that implements an interdisciplinary development process, from human-related modelling to software development activities (Section 4). This process is defined in detail and illustrated with research results, each of which concretizing the global vision of Assistive Computing.

2 ASSISTIVE COMPUTING

To address the above-mentioned challenges, we promote *Assistive Computing* as an interdisciplinary research area that ensures a continuum of expertise from user modeling to needs-driven development of assistive services,[1] to experimental evaluation of services in natural setting. We define this research area as follows.

> *Assistive Computing refers to computing systems that neutralize a mismatch between the cognitive capabilities of an individual and the environmental demands. In doing so, assistive computing systems facilitate the inclusion of individuals with cognitive impairments in mainstream environments* (e.g., *home, work, and school).*

By nature, we ground Assistive Computing in human-centered foundational principles. Specifically, services must be developed based on a *needs analysis* and must deliver assistance driven by *empowering user interactions.* Assistive Computing strives to demonstrate *evidence-based effects* of assistive services (rehabilitation and/or neutralization of impairments) and to achieve their *long-term adoption.* The key target outcomes of Assistive Computing include improved health and user autonomy *across daily activities.* The overall approach to Assistive Computing is depicted in Figure 1 and is further introduced in the next section.

[1]The term *assistive service* designates the functionality of an assistance, abstracting over its concrete implementation (whether hardware or software), which we refer to as a *technology.* For example, a service reminding a user to take their medication can be carried out by a dedicated device or a mobile application.

3 PRINCIPLES

Let us now elaborate on these foundational principles before we present our tool-based methodology to developing assistive computing support.

3.1 Human-Specific Design

Assistive Computing is fundamentally interdisciplinary because it involves not only computing but also many human dimensions. From a computer science viewpoint, leveraging knowledge from other disciplines, such as human factors, is what should give scientific grounds to the smartness/intelligence, claimed or intended by some computing systems assisting users. Furthermore, an interdisciplinary approach should allow the design to be systemic, contributing to the successful introduction of assistive services in the user environment. For example, older adults tend to underestimate their daily difficulties [23], requiring their caregivers to complement the needs analysis.

Assistive Computing is rooted in the specificities of each population targeted by an assistive support. These specificities are drawn from disciplines that study and model human behaviors, including neurology, cognitive science, psychology, human factors, and occupational therapy. These sources of knowledge allow the target population to be modeled in terms of sensory, motor and cognitive functioning. Such a modeling approach is proposed by Meyer *et al.* with their Executive-Process Interactive Control (EPIC) architecture [30], depicted in Figure 2. EPIC has been used to evaluate the effects of aging; it represents the cognitive capabilities of the user as the simplified components of a computing machine (*e.g.*, visual processor, cognitive processor, working memory). Not only can EPIC be used as a building block for user modeling, but it also provides a framework to design various aspects of an assistive service, such as user information layout, interaction channels, user input/output devices, user customization capabilities, and system features, as illustrated in Section 4.4.4.

3.2 User-Centered Dimensions

To further ground Assistive Computing in human-related sciences, we have studied the literature and identified four key user dimensions that should drive the design of assistive services. Specifically, 1) this design should be fueled by user needs, not by technology features; 2) it should empower users, contributing to technology acceptance; 3) it should promote and support adaptive strategies, leveraging proven user practice; and, 4) it should take into account existing classification approaches for assistive services, facilitating the navigation of available services by users and caregivers. Although, researchers commonly promote these dimensions, or identify them as lacking in existing works, more research is needed to evaluate their importance and complement them when necessary.

Let us further examine these four dimensions.

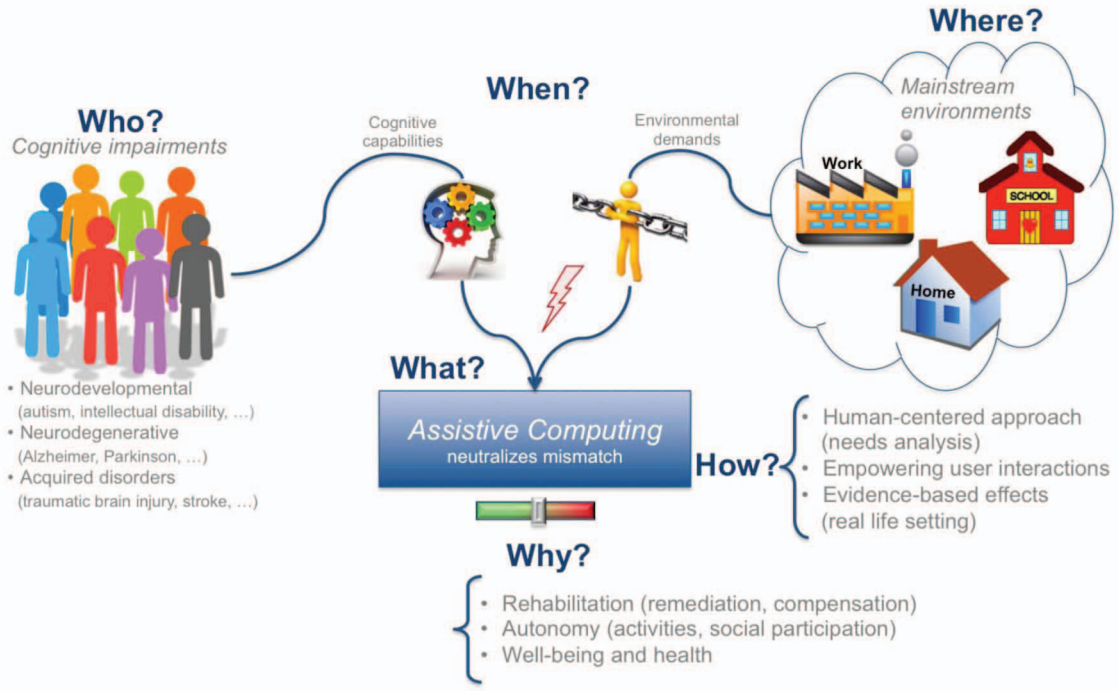

Figure 1: The overall approach to Assistive Computing

3.2.1 Addressing User Needs. Assistive services should be based on a needs analysis, conducted with the target population, as well as their caregivers, when appropriate. Needs analyses use a host of methods to gather information about the user (*e.g.*, questionnaires, task analysis, participatory design) on domains of interest (*e.g.*, activities of daily living) and may adapt or introduce new methods, when necessary (*e.g.*, technology probes [24]).

For example, developing an assistive service may entail performing a task analysis to identify the requirements for accomplishing the task objectives with respect to a user profile. These requirements are addressed when developing the assistive service logic, as well as the type and level of support to be provided to the user, the modalities of the assistive support, the indicators of the task progression, *etc.*

3.2.2 Promoting Self Determination. The logic of an assistive service places the users at the center of the assistive support and enables them to maintain and/or acquire a desired functional status in target daily activities.

Specifically, assistive services revolve around the users, who are presented with choices to make and decisions to take, following the self-determination theory [34]. These empowering interactions are adapted to their cognitive capabilities and their sensory-motor abilities. They are systematically introduced in assistive services to ensure that the user always has an opportunity to handle a situation autonomously, prior to resorting to an external intervention. By promoting systems that deliver assistance in the form of user interactions, Assistive Computing goes beyond systems that solely collect information from user activities, typically for screening purposes (*e.g.*, gait assessment [26]).

3.2.3 Supporting Adaptation Strategies. The service logic supports the adaptive strategies of users with cognitive impairments toward allowing them to reach their desired functional status. Following Baltes' model [3], assistive services should play different roles depending on the capabilities of the users and their goals. Assistive services should allow users to *Select* goals, to *Optimize* cognitive resources, and to *Compensate* when the required capabilities are lacking, hence the SOC model. Although general, these principles should contribute to build a framework to explore and prioritize the needs for assistive services in a target population. Seelye *et al.* explore this avenue of research by investigating the amount of prompting and the types of prompts required to assist individuals with mild cognitive impairment [37].

Complementary to the SOC model is the concept of *environmental support* that aims to improve user performance along two

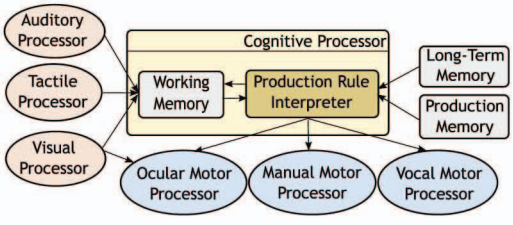

Figure 2: EPIC computational model [30]

dimensions: 1) reducing task demands on cognitive resources and 2) promoting efficient use of these resources [17]. Morrow and Rogers further develop this concept and propose an approach to designing supports for older adults' performance by identifying specific goals that can be achieved by environmental support [31]. For example, along the first dimension (*i.e.,* reducing task demands), the aim of environmental support can be to enhance task-relevant information (*e.g.,* by increasing font size), or to increase processing opportunity (*e.g.,* by slowering speech) [31]. We could build upon their approach to guide the development of principles for designing applications that optimize performance in assisting users in their daily tasks.

3.2.4 Classifying Assistive Services. Various classification approaches have been proposed, allowing assistive products to be grouped according to such dimensions as health conditions [33], needs [1], and activity types [22]. These classifications should be leveraged to position a new assistive service along the existing different classification dimensions and to ensure it matches existing requirements.

3.3 Cross-Cutting Assistive Support

Users with cognitive impairments simply cannot operate an unbounded number of assistive technologies as their needs for assistance increase, either because of declining cognitive resources or expanding activities of interest. In particular, Scherer has shown that older adults should only operate a few assistive technologies for them to be used and accepted [36]. Consequently, assistive services should be as integrated as possible in some form of a computing platform that unifies user interactions and shields the user from service variations. Such an approach reduces the cognitive cost of technology and allows scaling up the number of assistive services. This approach is exemplified by the notification system, introduced by Consel *et al.* [12], which enforces a programming interface to service developers, unifying resulting services in their interaction protocol with users.

3.4 Validation

The abundance of assistive technologies raises major challenges for potential users: Will a given assistive technology produce the desired or expected effects? Will it be usable regardless of how frequently it needs to be operated, or how much maintenance it requires? Will it reach acceptability, fostering long-term adoption?

To address these challenges, Assistive Computing should rely on field studies conducted as clinical trials, scientifically validating an assistive service with respect to three dimensions: clinical, ergonomic, and acceptability. Measuring these dimensions should be inherently part of developing assistive computing support; it should prevent false and erroneous claims, and promote validation with respect to criteria from human-related disciplines.

Field studies should go beyond a lab setting and eventually be performed in a natural setting. These field studies should be conducted over a long enough period of time to allow robust measurements to be collected and with diverse enough samples to allow for generalization. In doing so, experimental results, whether positive or negative, will provide researchers with valuable information to further research in Assistive Computing.

Although necessary, the above validaton guidelines are not sufficient if the technology development does not involve interdisciplinary work. Typically, when human-related researchers study assistive technologies, their work consists of conducting a clinical trial using an off-the-shelf product (*e.g.,* pill reminder, alarm, phone) in a variety of settings, from laboratory to natural. In doing so, they do not apply their knowledge of a given population to identify its needs and fuel the design of an assistive computing system. Symmetrically, computer scientists often take a technology-centered approach to developing a computing system for a given population, neglecting a proper needs analysis, and often falling into the trap of stereotypes. Furthermore, as underlined earlier, the resulting system does not undergo a rigorous validation, and may even remain at a prototype level.

This gap between human-related disciplines and computer science stems from the large spectrum of concerns involved in developing an assistive computing system and the lack of a global approach. To close this gap, we argue that a methodology that bridges the many dimensions of Assistive Computing is needed. For lack of space, the validation of this methodology is not discussed here.

4 A TOOL-BASED METHODOLOGY TO DEVELOPING ASSISTIVE COMPUTING SUPPORT

The overarching goals of our methodology can be formulated as follows. 1) Human-related experts should be provided with support to formalize their knowledge; 2) This knowledge should percolate throughout the development process; 3) It should be supported by mechanisms and tools toward systematizing both the development of assistive computing systems and the re-use of expert knowledge. These goals should contribute to ensure a variety of properties on the resulting assistive systems (*e.g.,* ergonomics, effectiveness).

Our proposed methodology provides a vehicle for a stepwise refinement process, starting with the *characteristics of target users* and *assistive goals*. These knowledge inputs are mapped into *assistive guidelines*, streamlining key design dimensions of assistive services. These guidelines are then captured by specific *computing abstractions*, depending on their nature. The resulting computing abstractions form a software foundation dedicated to specific assistive guidelines and providing a framework for the *software development* process. This approach is depicted in Figure 3 and should be understood as an iterative process. Let us now present the main phases and illustrate them with concrete examples.

4.1 Characteristics of Target Users

Following a human-centered approach, the goal of this phase is to gather the characteristics of the target population of users by analyzing their capabilities (cognitive, sensory and motor), their preferences, their needs, *etc.* These characteristics define a range of commonalities and variabilities to take into account when developing assistive services for a given population of users. For example, age-related declines lead to sensory impairments in the population of older adults. For another example, individuals with autism are known to process visual information more effectively than auditory information.

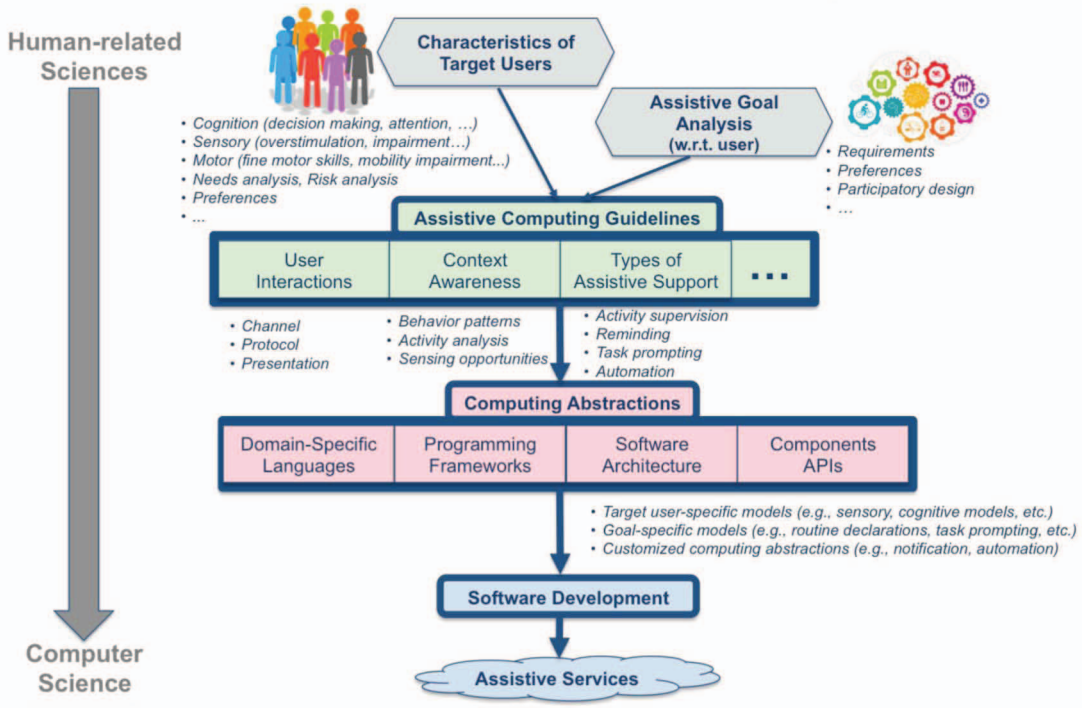

Figure 3: A methodology for Assistive Computing: from human-related sciences to computer science

Needs analysis can be supplemented by a risk analysis to assess potential risks in an environment with respect to a given population, and to identify strategies to manage them. For example, potential risks reported in the literature for a population of single older adults with cognitive difficulties include fire, wandering, and malnutrition.

4.2 Assistive Goals

The identification of an assistive goal emerges from the analysis of the target population and its needs. In turn, a given assistive goal should undergo an analysis phase involving the target users to gather such dimensions as requirements and preferences. Additionally, target users and other stakeholders can be invited to participatory design sessions, allowing a spectrum of practical issues to be collected. For example, developing assistive support for school inclusion, not only requires to address a specific need, but it should also comply with the requirements of the school stakeholders so that the assistive support not be disrupting, nor stigmatizing for the child.

Some assistive goals can crosscut different populations that share cognitive deficits (*e.g.*, attention, emotion), allowing a single assistive service to serve all of them. Gillespie *et al.* argue for the relevance of this approach and develop a framework for prescribing assistive technologies for cognition on the basis of a profile of cognitive deficits [22], rather than etiology or diagnosis. Following the work of Gillespie *et al.*, we should classify assistive goals with respect to the cognitive functions they target, widening the applicability of the resulting assistive applications. An example of such a generalization is given by the work by Fage *et al.*, where a tablet-based, assistive computing system dedicated to school inclusion, named School+, was used by both children with autism and intellectual disability. Both populations exhibited the same benefit of autonomy, when considering classroom routines [19].

4.3 Assistive Computing Guidelines

Once a given assistive goal has been identified, it needs to be decomposed using dimensions that should be meaningful to human-related experts and actionable to computer scientists, forming a bridge to pass expertise across disciplines. The study of existing assistive technologies revealed three key dimensions to consider: *user interactions*, *context awareness*, and *the types of assistive support*. The critical role of these dimensions in our methodology prompted us to propose guidelines to tackling them in the context of Assistive Computing.

We now define these three dimensions and outline a few guidelines for each of them.

User interactions. The first dimension of assistive computing guidelines addresses what interactions occur between an assistive computing system and a user, how they are realized, and when they occur. These interactions should be specific to the characteristics of the target population: information coding, channels, user input modalities, user interaction protocol, *etc.* For example, to improve the response performance of older adults to notifications, dual coding of information can be used (*e.g.*, image and text), as suggested by Warnock *et al.* [42]. For another example, interactions with users with autism should promote the visual channel, as they process visual information more effectively [19].

The level of criticality of a user interaction is determined with respect to the assistive goal, the user and their environment. For example, monitoring a stove may critically require an interaction with the user. Other interactions may not require the user to interrupt an ongoing task, corresponding to low-priority situations.

Context awareness. It is quite challenging to deliver the right assistance at the right time. Nevertheless, achieving this goal is fundamental to making the assistance effective and acceptable. Specifically, a key dimension to take into account is context awareness. That is, determining contextual information that is relevant to an assistive goal. A number of contextual parameters may need to be considered, from direct ones (*e.g.,* time, location, user input) to indirect ones (*e.g.,* environment interactions indicating an ongoing activity or its absence).

The context awareness dimension can heavily depend on the characteristics of the target population, as illustrated by older adults who become increasingly routinized in their daily functioning as they decline cognitively [6]. This situation can be leveraged to detect user activities by monitoring sensor-equipped locations in the environment (*e.g.,* a contact sensor on a drawer). Detecting such activities can in turn provide accurate context information to timely deliver assistance, as shown by Caroux *et al.* [9].

A widely studied alternative to this knowledge-based approach consists of using machine learning to recognize activities. This alternative is proposed by the Casas smart home [15], which is aimed at the general population. Although Casas uses machine learning without training, in practice, it may not accurately recognize a range of activities. This constrasts with a knowledge-based approach, such as that of Caroux *et al.*, which however requires human intervention to collect routines, besides being restricted to older adults. A promising avenue of research is to combine both approaches.

Types of assistive support. This dimension of assistive computing guidelines explores the types of assistance needed by a given population and a given goal. These types include 1) *activity supervision* – ensuring that the user performs their activities, 2) *reminding* – cuing the user for such events as appointments, domestic chores, and medication intake, 3) *task prompting* – providing a given level of step-by-step guidance to perform a task.

Note that these three types of assistive support can be combined to provide an increasing level of user assistance. For example, a user may first be supervised to ensure that a task is performed. If the task is not performed, the user may be cued with a reminder notification. In turn, depending on the user's needs and preferences, this reminder notification may be followed by the launch of a task prompter to guide the user through the task steps.

From the viewpoint of the SOC model, these types of assistive support aim to optimize the existing capabilities of the user. However, when the required capabilities are lacking, the user must delegate a task to an assistive service. Then, compensation can be realized with different levels of automation. For example, when the user gets out of bed at night to go to the bathroom, full automation can be used to turn on the lights to the bathroom.

As the area of Assistive Computing is explored, we believe that additional dimensions of assistive computing guidelines will emerge to provide further refined frameworks for the development of assistive services.

4.4 Computing Abstractions

Our proposed methodology provides four outlets to diffuse the dimensions of assistive computing, presented earlier, towards computing artifacts: *software components*, *software architectures*, *programming frameworks*, and *domain-specific languages*. Furthermore, depending on the scope of the assistive computing guidelines, they may apply to, and thus factorize over, a target population and/or a given assistive goal.

Generally speaking, decomposing an assistive goal into dimensions of assistive computing should be viewed as activities aimed to model expert knowledge. These models may 1) address a restricted functionality, which can be encapsulated in a software component, 2) elicit a structural feature, which can be captured at the level of a software architecture, 3) require recurring program patterns, which can be built into a programming framework, or 4) address high-level, expert concepts, which can call for the development of a domain-specific language. Let us examine and illustrate, each of these four computing abstractions.

4.4.1 Software Components. A software component implements a functionality that serves as a building block to develop a software system. In Assistive Computing, this aspect can correspond to a user-visible functionality (*e.g,.* notification system) or an internally-used functionality (*e.g.,* sensor logging). A software component is aimed to be shared/re-used across a software system, ensuring system-wide homogeneity for the corresponding functionality.

From a software engineering viewpoint, a software component exposes contractually specified interfaces and explicit context dependencies, making explicit how it can be composed by client components. In doing so, these composition constraints enforce usage scenarios: what operations can be used? When they can be used? What parameters are valid?

To illustrate this idea, consider a user notification service of an assistive computing system. This service can be developed as a software component, shared by assistive computing services, unifying notifications across the system. This approach was used by Consel *et al.* [12]: they developed such a component for an assisted living platform, dedicated to older adults, named HomeAssist [13]. Taking into account the specificities of this population, this notification service uses multimodal coding of notifications (tone, shape, color, and text) to improve the user response performance, as suggested earlier in our assistive computing guidelines. Also, they introduced two types of notifications: non-critical and critical. Non-critical notifications do not require the user to process them immediately, whereas critical notifications are issued when safety or security is involved. Figures 4 and 5 display examples of a non-critical and a critical notification, respectively. Each of these two types of notifications corresponds to a specific procedure to be followed by the user: non-critical notifications can be ignored by the user, whereas critical ones do not disappear until the user has resolved the situation.

Besides encapsulating assistive computing guidelines, this notification service provides a vehicle for aging experts and software

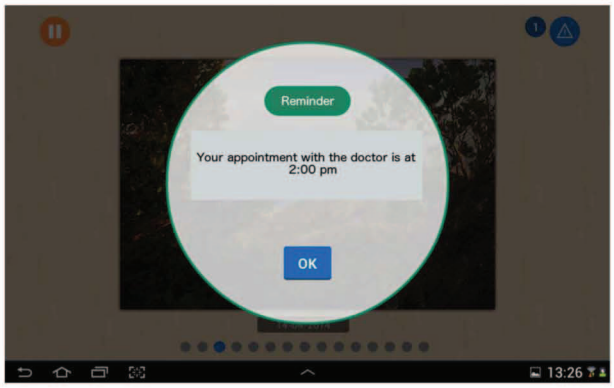

Figure 4: A non-critical notification

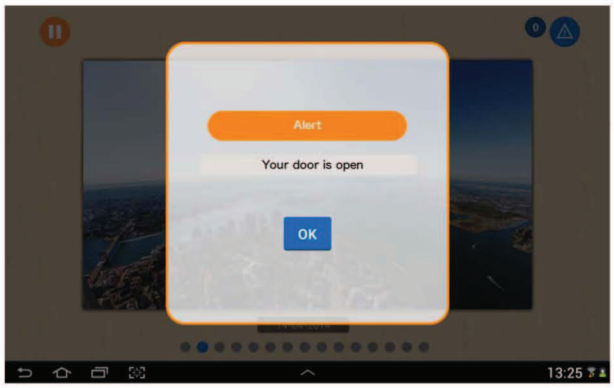

Figure 5: A critical notification

developers to collaborate. In particular, drawing from prior knowledge and/or older adult inputs, the aging expert determines for each assistive service, the types of notifications required, depending on the target user population and the nature of the risks involved. This information is directly mapped into the interface of the notification service by the software developer. As a byproduct, assistive services using this notification service can be guaranteed to offer specific ergonomics properties required by older adults. Furthermore, making mandatory the use of this notification service by application developers unifies user interactions with assistive services and allows the scope of assistance to scale up without incurring additional cognitive cost. As such, the notification component is a prime example of the notion of cross-cutting assistive support, introduced earlier (Section 3.3).

4.4.2 Software Architectures. Sometimes, assistive computing guidelines cut across the entire assistive computing system. Context awareness is a key illustration of this: for home autonomy, it is used by a range of applications to track daily activities (*e.g.*, meal preparations, appointments, medication intake) and to ensure they are performed as planned. As a result, the architecture of an assisted living platform should provide architectural support to deliver context information to applications with dedicated software connectors [40], such as an event mechanism dedicated to a taxonomy of context topics for assisted living.

Because assistive computing systems revolve around context awareness, a software architecture should include an infrastructure dedicated to computing context information. Coutaz *et al.* make a proposal toward this goal [16] with an infrastructure that consists of four layers: 1) sensing – to measure environmental parameters, 2) perception – to produce symbolic values at the appropriate level of abstraction from sensing data, 3) situation and context information – to identify the current situation and context from symbolic values and to detect conditions for moving between situations and contexts, and 4) exploitation – this layer exploits situation and context information to fuel context-sensitive applications. These layers are used by Nehmer *et al.* to explore a logical software architecture dedicated to an ambient intelligent home care system [32]. Although promising, this line of research is still in its early stage, mainly focusing on conceptual concerns. More research is needed to refine and enrich these layers and concepts to cover realistic assistive goals, and put them to practice by using them to develop a range of assistive services.

For another example of architectural support, consider a school inclusion platform with a variety of training applications, designed as serious games, as proposed in the School+ project [19]. These applications could be provided with architectural support to manage scores, access levels, configure user preferences, and define reward strategies.

To account for inter- and intra-individual variabilities, assistive computing systems should provide architectural mechanisms capable of matching needs with assistive services. This requirement can be addressed by a catalog of applications, assisting a range of activities. Users can select a set of specific applications from the catalog to cover specific needs, and evolve this set over time, as necessary. Such an architectural support is used in the HomeAssist platform [13].

4.4.3 Programming Frameworks. Programming frameworks go further in supporting software development than software architectures in that they provide skeletal support to build applications [20]. This skeletal support is often special purpose, allowing software development to focus on the application logic and to maximize the sharing of functionalities. Typical examples of programming frameworks include smartphone platforms (*e.g.*, Android, iOS) and Web services (*e.g.*, Ruby on Rails).

In the context of Assistive Computing, programming frameworks permit to share accessibility features system-wide, as illustrated by smartphone platforms. Features such as VoiceOver and screen magnification can be uniformly supported at the platform level, as well as at the application level. Specifically, the development of an application is framed within operations that support some accessibility features. Thus, user interactions of an application offer accessibility features by virtue of using the underlying programming framework.

Programming frameworks have another distinguishing feature that separates them from software architectures and software components: inversion of control. This feature allows the framework to mainly govern the flow of control of an application, not the application itself. As a consequence, applications can follow pre-defined behavior patterns, enforced by the underlying framework. This situation improves the homogeneity of applications, reduces the

learning curve for developers, and facilitates the introduction of new applications.

Going beyond this approach, Bertran *et al.* introduced a tool-based, design-driven methodology, named DiaSuite [4], dedicated to developing applications in such areas as pervasive computing [11] and Internet of Things [14]. This methodology allows the developer to design an application as a set of high-level declarations, which are then compiled into a customized programming framework. This generated framework supports and restricts the programming phase [41].

Consel *et al.* proposed various extensions to DiaSuite towards meeting the needs for the development of assistive services for aging in place; these extensions were integrated into the HomeAssist platform [13]. Resulting features include a dedicated notification system (discussed in Section 4.4.1), support for a stationary tablet displaying notifications, and a taxonomy of devices, dedicated to assisted living activities and shared across assistive applications [13]. In doing so, human-related expertise has been introduced from the design, to the programming, to the runtime support of assistive applications.

4.4.4 Domain-Specific Languages. Raising higher the level of abstraction, we propose a methodology that revolves around domain-specific languages [29], to model users and to specify aspects of assistive services. These languages should be used as a conceptual vehicle serving the human-related experts to turn their expertise into artifacts; they should be high level and domain specific, and thus require minimum training for non-computer scientists. Artifacts of expertise should concretize and support interactions between different disciplines and stakeholders. Beyond their documentation purposes, these artifacts should be used to guide and restrict the programming activity of assistive services and/or to automatically generate software layers, ensuring that the implementation of the services conforms to the declarations of the human-related experts. These benefits should further validate the approach of Bertran *et al.*, as exemplified by the DiaSuite project [4].

Characteristics of Target Users. We propose to introduce domain-specific languages to model key dimensions of user populations, including user characteristics, requirements, goals, and preferences. Let us illustrate our approach by presenting a modeling language dedicated to describing user characteristics. This language was developed by Balland *et al.* [2] and focused on specifying the sensory, motor, and cognitive capabilities of users, leveraging the EPIC architecture displayed in Figure 2. The EPIC model allows on the user side to model a target population, declaring what resources they provide, and on the assistive service side to define what resources this service requires. This is illustrated in Figure 6 by the user model whose characteristics are drawn from a study of the common effects of aging using EPIC [30]. This model defines the typical older adult as having a reduced visual capability, thus advising against interactions via abundant textual messages. As well, this user model declares fully functional vocal capability, allowing interactions via vocal commands. Because the working memory is reduced, interactions should be decomposed into manageable steps.

When designing an assistive computing application, another layer of Balland *et al.*'s domain-specific language is used to declare the user interactions of an application, in the spirit of approaches

```
user_model average_olderAdult {
  cognitive_processor = medium;
  working_memory = medium;
  long_term_memory = medium;
  short_term_memory = high;
  visual_processor = medium;
  auditory_processor = low;
  manual_motor_processor = low;
  vocal_motor_processor = high;
}
```

Figure 6: A model of a typical older adult

from the domain of Human-Computer Interactions [38]. Specifically, this layer defines the information exchanged between the user and the application via three *abstract interactors*: *input* defines information provided by the user to the application; *output* is the converse; and, *prompt* defines a round-trip between the application and the user (*i.e.*, an output for a question to the user and an input for their response). These abstract interactors are then instantiated with concrete interactors. For example, consider an application that would prompt a user to read new email messages; this could be realized using a connected TV set to ask the user whether they are ready to read new messages and the TV remote control to acquire their response. Associated with these concrete interactors are sensory, motor, and cognitive costs. As a result, the application instantiated with concrete interactors can be matched against a population of target users, as defined in Figure 6. In fact, this matching can be performed, prior to deployment time, allowing the user to anticipate the interaction requirements of an application. Beyond addressing the user, Balland *et al.* show that their application declarations of user interactions can be used to compile a customized programming framework that facilitates and restricts the implementation of the user interactions of an application (as explained in Section 4.4.3). As a result, applications are guaranteed to correctly announce their user interactions. (More details are given elsewere [2].)

As shown by Balland *et al.*'s work, a domain-specific language provides a framework to express knowledge, with notations and concepts drawn from human-related disciplines (*e.g.*, cognitive science). The syntactic and semantic definitions of such a language allow models to be a communication medium for knowledge and to be processed by tools to support the later stages of development.

A natural continuation of their work would be to measure the cost of interactors for the typical user, given a specific population. Then, for a set of interaction declarations of a given application, a cognitive architecture could simulate the typical user to estimate the overall cost of the application or that of specific tasks. This approach is inspired by the work of Card *et al.*, where they combine their Model Human Processor and a cognitive architecture (GOMS) to generate quantitative predictions of human performance [8]. As shown by Jastrzembski and Charness, this simulation approach successfully predicts older adult performance, given appropriate information processing parameters of the typical user [25].

Other user interaction aspects could be modelled extending Balland *et al.*'s approach. For example, it can express constraints on information layout [19] and modality coding of information [12].

More generally, the domain-specific language approach could be applied to a number of user dimensions, including social abilities for

individuals with autism, homemaking skills for individuals losing autonomy, and leisure preferences.

4.5 Software Development

We have shown that actual software development is much streamlined, and sometimes partially automated, by the computing abstractions discussed earlier, namely, software components, software architectures, programming frameworks, and domain-specific languages. The higher level the approach, the more guarantees it offers: software components guide the development phase through interfaces, whereas domain-specific languages can be used to automatically generate customized programming frameworks, restricting the scope of this phase.

End-user software development. Beyond this spectrum, end-user programming is a promising avenue of research. In particular, visual approaches have been proposed to allow non-computer scientists to develop applications in the form of rules orchestrating services and devices. For example, Drey and Consel propose a visual language dedicated to defining home automation rules, based on a sensor-controller-actuator paradigm and parameterized with a taxonomy of devices [18], in the spirit of what was presented in Section 4.4.3. This work is a step towards providing to users and caregivers the ability to define their own scenarios of assistive computing services. Further research is needed to codesign end-user programming languages with target users, to evaluate their performance in writing their own applications, and assess the range of applications that can be expressed in practice.

Critical assistive computing systems. Regardless of the approach used for software development, it is essential to recognize that assistive computing applications are destined to a vulnerable population of users. This observation, compounded by the fact that assistive applications are likely to be intertwined with the daily activities of users, gives a *critical* nature to assistive applications. Indeed, these applications may be involved in the safety of the home by monitoring hazardous appliances (*e.g.,* stove, iron) and home components (*e.g.,* entrance door), the safety of the users (*e.g.,* light path, medication intake), the preservation of autonomy (*i.e.,* basic and instrumental daily activities), and the prevention of social isolation (*e.g.,* accessible social tools, social stimulation).

The critical nature of these dimensions calls for the tracing of the human-related requirements throughout the development process of assistive computing applications, as illustrated by Balland *et al.*'s approach on user interactions [2], implying an integrated approach, which combines computer-related and human-related disciplines.

Reliability and reliance of assistive computing systems. Because assistive computing systems are intertwined with the daily life of users, it is important to examine how users react to system imperfections (*e.g.,* miss and false alarms). As shown by Sanchez *et al.*, these imperfections have an impact on how the users relate to an automated system, leading them to under/over-relying on it during non-alarm states and when alarms are issued [35]. Furthermore, false alarm fatigue may be an obstacle for technology adoption, besides the likely stress caused to the users.

As a result, there is a need for a software development methodology that ensures the reliability of assistive computing applications. This methodology should not only build on extensive user behavior models, but it should also encompass strategies to ensure the reliability of external components, such as sensors, actuators, and networks, as studied by Carteron *et al.* [10].

5 RELATED WORK

We have argued that a tool-based methodology dedicated to developing assistive computing support should leverage software engineering principles and techniques to provide human-related experts with resources to formalize their knowledge and tools to process it. Besides software engineering, a range of computer science fields are paramount for developing assistive computing support. Most notably, they include human-computer interaction, accessible computing, and pervasive computing.

Human-computer interaction (HCI) and accessible computing provide us with principles and techniques to develop technological support that fits the specificities of target users. Pervasive (or ubiquitous) computing explore new services and devices, and propose new usages of technologies. HCI and pervasive computing have traditionally been targeting mainstream users but, in recent years, more works have focused on populations with specific needs, such as older adults and individuals with autism, showcasing systems that provide assistive support to users.

Assistive Computing crosscuts HCI, accessible computing and pervasive computing to cover all the dimensions of this research area. In fact, as illustrated by the work on the notification system for the home of older adults, Assistive Computing feeds on results from these fields (*e.g.,* general-purpose notification systems [28]), while introducing new requirements to take into account the specificities of a target population. As well, the emphasis on evaluating assistive support in a natural setting can lead to new techniques inspired by human-related sciences. For example, the repeated measures of timed usage scenarios to test the learnability of a notification system [12] is inspired by a test from gerontology to assess the functional status of an individual.

6 CONCLUSION

We have examined the pressing needs for the emergence of Assistive Computing as an interdisciplinary research area. We have proposed definitions and fundamental principles to organize and structure research efforts in this area. We have introduced a methodology that bridges the many dimensions involved in developing assistive computing systems. We have shown the pragmatic nature and applicability of the proposed methodology by using it to revisit existing research results, testing and illustrating its main aspects.

We invite the research community to respond to our call to action and address the research challenges and opportunities that exist at each stage of our proposed methodology. This includes 1) creating new domain-specific languages to allow human-related experts to further fuel the development process of assistive services, and computer scientists to further ensure human-specific properties; 2) lifting technology barriers by introducing new user interaction modalities that leverage innovative functionalities of devices; and, 3) enriching context awareness of assistive services by using wearable sensors to refine ambient sensor data.

While conducting this work, researchers will assess our proposed definitions and principles, evolve and extend assistive goals and guidelines, and put to practice our computing abstractions, taking this research area to the next level. As underlined throughout this paper, work in Assistive Computing should be conducted in close collaboration with human-related experts, who will further develop models of users and assess resulting assistive computing systems with field studies to reveal evidence-based effects.

7 ACKNOWLEDGEMENTS

I am very grateful to Nathalie Bier, Julia Lawall, Pierre-Yves Oudeyer, Wendy Rogers, and Nic Volanschi for their feedback on the very first versions of this paper and for inspiring discussions. Joëlle Coutaz, and Hélène Sauzéon provided me with valuable input on the final versions of this paper. Finally, thanks to Diane Cook, Jim Hook, Clayton Lewis, Jeffrey Kaye, and Catherine Plaisant for their encouragements. This material was supported by the European Regional Development Funds, the Nouvelle-Aquitaine province, the Gironde district, CARSAT, RPDAD/UDCCAS, CNSA, and Réunica.

REFERENCES

[1] Ronald M. Baecker, Karyn Moffatt, and Michael Massimi. 2012. Technologies for aging gracefully. *Interactions* 19, 3 (2012), 32.

[2] Emilie Balland, Charles Consel, Bernard N'Kaoua, and Hélène Sauzéon. 2013. A case for human-driven software development. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 1229–1232.

[3] Paul B. Baltes and Margret M. Baltes. 1990. Psychological perspectives on successful aging: The model of selective optimization with compensation. *Successful aging: Perspectives from the behavioral sciences* 1, 1 (1990), 1–34.

[4] Benjamin Bertran, Julien Bruneau, Damien Cassou, Nicolas Loriant, Emilie Balland, and Charles Consel. 2014. DiaSuite: A tool suite to develop Sense/Compute/Control applications. *Science of Computer Programming* 79 (2014), 39–51.

[5] Christina M. Blaschke, Paul P. Freddolino, and Erin E. Mullen. 2009. Ageing and technology: A review of the research literature. *British Journal of Social Work* 39, 4 (2009), 641–656.

[6] Jean Bouisson. 2002. Routinization preferences, anxiety, and depression in an elderly French sample. *Journal of Aging Studies* 16, 3 (2002), 295–302.

[7] Matthew W Brault et al. 2012. *Americans with disabilities: 2010.* US Department of Commerce, Economics and Statistics Administration, US Census Bureau Washington, DC.

[8] Stuart Card, Thomas Moran, and Allen Newell. 1986. The model human processor-An engineering model of human performance. *Handbook of perception and human performance.* 2 (1986), 45–1.

[9] Loïc Caroux, Charles Consel, Lucile Dupuy, and Hélène Sauzéon. 2014. Verification of daily activities of older adults: a simple, non-intrusive, low-cost approach. In *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility.* ACM, 43–50.

[10] Adrien Carteron, Charles Consel, and Nic Volanschi. 2016. Improving the Reliability of Pervasive Computing Applications By Continuous Checking of Sensor Readings. In *IEEE International Conference on Ubiquitous Intelligence and Computing.*

[11] Damien Cassou, Julien Bruneau, Charles Consel, and Emilie Balland. 2012. Toward a tool-based development methodology for pervasive computing applications. *IEEE Transactions on Software Engineering* 38, 6 (2012), 1445–1463.

[12] Charles Consel, Lucile Dupuy, and Hélène Sauzéon. 2015. A Unifying Notification System To Scale Up Assistive Services. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility.* ACM, 77–87.

[13] Charles Consel, Lucile Dupuy, and Hélène Sauzéon. 2017. HomeAssist: An Assisted Living Platform for Aging in Place Based on an Interdisciplinary Approach. In *Proceedings of the 8th International Conference on Applied Human Factors and Ergonomics (AHFE 2017).* Springer.

[14] Charles Consel and Milan Kabáč. 2017. Internet of Things: From Small- to Large-Scale Orchestration. In *Distributed Computing Systems (ICDCS), 2017 IEEE 36th International Conference on.* IEEE, 303–312.

[15] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. 2013. CASAS: A smart home in a box. *Computer* 46, 7 (2013), 62–69.

[16] Joëlle Coutaz, James L Crowley, Simon Dobson, and David Garlan. 2005. Context is key. *Commun. ACM* 48, 3 (2005), 49–53.

[17] Fergus IM Craik, F. Klix, and H. Hagendorf. 1986. *A functional account of age differences in memory.*

[18] Zoé Drey and Charles Consel. 2012. Taxonomy-driven prototyping of home automation applications: A novice-programmer visual language and its evaluation. *Journal of Visual Languages & Computing* 23, 6 (2012), 311–326.

[19] Charles Fage, Léonard Pommereau, Charles Consel, Emilie Balland, and Hélène Sauzéon. 2016. Tablet-based activity schedule in mainstream environment for children with autism and children with ID. *ACM Transactions on Accessible Computing (TACCESS)* 8, 3 (2016), 9.

[20] Mohamed Fayad and Douglas C. Schmidt. 1997. Object-oriented application frameworks. *Commun. ACM* 40, 10 (1997), 32–38.

[21] Arthur D. Fisk, Wendy A. Rogers, Neil Charness, Sara J. Czaja, and Joseph Sharit. 2009. *Designing for older adults: Principles and creative human factors approaches.* CRC press.

[22] Alex Gillespie, Catherine Best, and Brian O'Neill. 2012. Cognitive function and assistive technology for cognition: A systematic review. *Journal of the International Neuropsychological Society* 18, 01 (2012), 1–19.

[23] David A. Gold. 2012. An examination of instrumental activities of daily living assessment in older adults and mild cognitive impairment. *Journal of clinical and experimental neuropsychology* 34, 1 (2012), 11–34.

[24] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B. Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, et al. 2003. Technology probes: inspiring design for and with families. In *Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, 17–24.

[25] Tiffany S Jastrzembski and Neil Charness. 2007. The Model Human Processor and the older adult: parameter estimation and validation within a mobile phone task. *Journal of Experimental Psychology: Applied* 13, 4 (2007), 224.

[26] Jeffrey Kaye, Nora Mattek, Hiroko Dodge, Teresa Buracchio, Daniel Austin, Stuart Hagler, Michael Pavel, and Tamara Hayes. 2012. One walk a year to 1000 within a year: Continuous in-home unobtrusive gait assessment of older adults. *Gait & posture* 35, 2 (2012), 197–202.

[27] Suzanne Martin, Greg Kelly, W George Kernohan, Bernadette McCreight, and Christopher Nugent. 2008. Smart home technologies for health and social care support. *Cochrane Database Syst Rev* 4 (2008).

[28] D Scott McCrickard, Mary Czerwinski, and Lyn Bartram. 2003. Introduction: design and evaluation of notification user interfaces. *International Journal of Human-Computer Studies* 58, 5 (2003), 509–514.

[29] Marjan Mernik, Jan Heering, and Anthony M Sloane. 2005. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)* 37, 4 (2005), 316–344.

[30] David E. Meyer, Jennifer M. Glass, Shane T. Mueller, Travis L. Seymour, and David E. Kieras. 2001. Executive-process interactive control: A unified computational theory for answering 20 questions (and more) about cognitive ageing. *European Journal of Cognitive Psychology* 13 (2001), 123–164.

[31] Daniel G. Morrow and Wendy A. Rogers. 2008. Environmental support: An integrative framework. *Human Factors* 50, 4 (2008), 589–613.

[32] Jürgen Nehmer, Martin Becker, Arthur Karshmer, and Rosemarie Lamm. 2006. Living assistance systems: an ambient intelligence approach. In *Proceedings of the 28th International Conference on Software Engineering.* ACM, 43–50.

[33] Martha E. Pollack. 2005. Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. *AI magazine* 26, 2 (2005), 9.

[34] Richard M. Ryan and Edward L. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist* 55, 1 (2000), 68.

[35] Julian Sanchez, Wendy A. Rogers, Arthur D. Fisk, and Ericka Rovira. 2014. Understanding reliance on automation: Effects of error type, error distribution, age and experience. *Theoretical Issues in Ergonomics Science* 15, 2 (2014), 134–160.

[36] Marcia J. Scherer. 2011. *Assistive technologies and other supports for people with brain impairment.* Springer Publishing Company.

[37] Adriana M. Seelye, Maureen Schmitter-Edgecombe, Diane J. Cook, and Aaron Crandall. 2013. Naturalistic assessment of everyday activities and prompting technologies in mild cognitive impairment. *Journal of the International Neuropsychological Society* 19, 04 (2013), 442–452.

[38] Ben Shneiderman and Catherine Plaisant. 2010. *Designing the User Interface - Strategies for Effective Human-Computer Interaction (5. ed.).* Addison-Wesley. I–XVIII, 1–606 pages.

[39] Adam Steventon, Martin Bardsley, John Billings, Jennifer Dixon, Helen Doll, Michelle Beynon, Shashi Hirani, Martin Cartwright, Lorna Rixon, Martin Knapp, et al. 2013. Effect of telecare on use of health and social care services: findings from the Whole Systems Demonstrator cluster randomised trial. *Age and Ageing* 42, 4 (2013), 501–508.

[40] R. N. Taylor, N. Medvidovic, and E. M. Dashofy. 2009. *Software Architecture: Foundations, Theory, and Practice.* Wiley Publishing.

[41] Paul van der Walt, Charles Consel, and Emilie Balland. 2016. Frameworks compiled from declarations: a language-independent approach. *Software: Practice and Experience* (2016).

[42] David Warnock, Marilyn McGee-Lennon, and Stephen Brewster. 2013. Multiple notification modalities and older users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 1091–1094.