Journal of Ambient Intelligence and Smart Environments 5 (2013) 47–64
DOI 10.3233/AIS-120190
IOS Press



# A context-aware service provision system for smart environments based on the user interaction modalities

Charles Gouin-Vallerand [a,*], Bessam Abdulrazak [b], Sylvain Giroux [b] and Anind K. Dey [c]
[a] *LICEF Research Center, Télé-Université du Québec, 5800 Saint-Denis Street, Montreal, Quebec, H2S 3L5, Canada*
[b] *DOMUS Laboratory, Université de Sherbrooke, 2500 Boul. de l'Université de Sherbrooke, J1K2R1, QC, Canada*
*E-mail: {bessam.abdulrazak,sylvain.giroux}@usherbrooke.ca*
[c] *Human Computer Interaction Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, 15213, United States*
*E-mail: anind@cs.cmu.edu*

**Abstract.** Ambient and pervasive technologies provide several ways to assist people with special needs in smart environments. However, the system's complexity and the size of the contextual information of these environments lead to several difficulties in deploying and providing the assistance services. A service provision mechanism which is aware of the environment context can simplify the deployment of assistance services on environment devices, by taking care of the decision processes. Moreover, the integration of the interaction modalities in the decision processes of such mechanisms allows deliveries of services to users based on their capabilities and preferences. In this paper, we present a context-aware service provision system for smart environment, which takes into account a whole set of contextual information: user profiles, device profiles, software profiles and environment topology. In regards to our previous work, this paper focuses on the modeling of the user interaction capabilities, built around the notion of interaction modalities. We also detail the integration of the model to the service provision reasoning process, as well as its implementation. Finally, we demonstrate the functionalities of this system through technical validations and scenarios carried out in a real smart apartment.

Keywords: Service provision, ubiquitous computing, context-awareness, user interaction modalities

## 1. Introduction

The fast development of the information and communication technologies has opened a large variety of new services and assistances to help people with special needs (PwSN) with, for instance, physical or cognitive restrictions. Combining assistance and ubiquitous technologies allows these people to benefit from augmented environments that are assisting them in their daily living activities.

A Context-aware system hosts agents that infer additional, synthetic Context from the raw Context provided by sensors and from other synthetic Context. Context-awareness enables such system, by assisting users in performing daily life activities or warning specialized personnel that human intervention is required. Agents can consume Context, produce Context for others to consume, or use Context to decide upon an application domain-dependent course of action.

Numerous efforts have been made in the development of platforms to support Context-awareness for pervasive computing [8,20]. Most applications and studies today rely on smart spaces i.e. physical locations equipped with a set of sensors and actuators where the basic physical layout is known beforehand. These spaces include any controlled environ-

*Corresponding author. E-mail: charles.gouin-vallerand@teluq.ca.

ment where Context-awareness could play a role such as assisting people with disabilities (e.g. hospitals, hotel rooms, apartments, houses, classrooms).

Such kind of smart environments will be increasingly more frequent in the industrialized countries (like in North America, Europe and Japan) as their proportion of aged populations is increasing [30]. In addition to helping dependant people in increasing their autonomy and quality of life, the smart environments allow to reduce the needed interventions from professional healthcare caregivers, reducing by the way the induced costs.

In a way to assist these users in their daily living activities, a dynamic, context-aware and intelligent mechanism is required to deploy assistive services on the different devices present in the smart environments, such as smart phones, tablets, desktop computers, laptops, embedded computers, etc. However, the complexity of the smart environments [28], with their heterogeneous devices, specific configurations, and the important quantity of information to process, turn the service provision into a serious challenge when dynamism and context precision are some of the system's requirements.

In our precedent work [11–13], we proposed a context-aware middleware, which was helping environment managers and professionnal caregivers in organizing and deploying new software in smart environments. Rallying several technologies e.g. Web services, OSGi, OWL ontologies and fuzzy logic, the middleware uses the contextual information related to the device's hardware and the position of the components in the environments. Moreover, this middleware takes into account the devices' position in specific zones during the deployment of the software components. However, this system had some issues:

- user profiles was not taken into account in the reasoning process while users are in fact the focus point in most of the organization process;
- context reasoning was centralized in the coordinator node (Section 3.1) of the system while we were working toward a context awareness model that is distributed and layered (Section 3.1).

We re-designed and implemented a second version of this middleware, called Tyche project [13], which focus on providing software components (i.e. services) to users by taking into account the users' interaction capabilities, based on the interaction modalities, users' preferences and their locations/orientations in the smart environment. In fact, these additions represent major enhancements compared to the precedent work, where the system more than double in code line size (around 40 000), integrate a new context awareness model (Section 3.1), have extended functionnalities with service provision functions and has been tested/validated in a real smart environment.

Therefore, in this paper, we present the service provision functionalities that we integrated to the core mechanisms of this context-aware middleware. These functionalities allow to deploy services toward the users, located in the smart environment, based on a large array of contextual information, including:

- the user profiles: users' capabilities and preferences;
- the environment topology: object locations, contextual zone descriptions, environment physical structures (walls), etc.;
- the device profiles: devices' resources, utilization, location, connected interaction peripherals, etc.;
- the service profiles: service needs and characteristics, e.g., needed resources, topology, software and hardware dependencies, GUI description.

The major innovation of the proposed service provision system consists in the integration of the user profile in the provision's reasoning process around the computer interaction modalities [19]: the sense, the perception, the motivity (i.e. mobility) and the cognitive modalities. By reasoning over these information on the base of the interaction modalities, our system is able to provide assistance services to the users on the devices of an environment that are most adequate to the current context (user location, environment topology, etc.) and users' capabilities/preferences.

To evaluate the viability and efficiency of our proposed solution, we also conducted a series of tests and validations. We put in place ten scenarios to evaluate the proposed functionalities in different contexts with different types of users.

In the next section, we present an overview of the existing work on the service provision system for the smart environments. The second section presents our service provision model based on the contextual information and the user's interaction modalities. This section is divided in several subsections describing the different strategies used to process the interaction modalities. The third section presents the validation scenarios and the results collected during the implementation of the scenarios. We finally conclude this

paper by resuming the proposed solution and introducing some future works.

## 2. Related work

In this section, we are presenting a short overview of some work related to the service provision in smart environments and in mobile computing setting. We are also presenting work on the adaptation and modelization of the interaction between the users and the devices. These works guided us during the conception and the implementation of our system.

In the Gaia project, Ranganathan and Campbell [22] propose some mechanisms related to the service provision and software self-organization. Their solution uses an ontology and semantic matching to find the right device configuration face to the needs and dependencies of software that must be deployed in a given mediated space. Ranganathan's work on the software organization is mainly focusing on the hardware analysis and the environment spatial description, evacuating the user needs, capabilities and preferences.

Ghorbel et al. [10] propose a solution to provide service to users' smart phone while they are entering pervasive environments. Their work focuses on the utilization of the semantic reasoning to choose the right end-user services, for mobile devices, according to the profile of the smart phone users. However, their work doesn't take in to account a full mediated pervasive environment with several end user devices, which can support the users instead of their smart phones.

Syed et al. [27] present an architecture for a service provision and software self-organization framework for pervasive systems. This framework used a knowledge base where devices, processes and tasks are defined through recipes, capabilities, rules and properties concepts. The reasoning process in their work is similar to a case-base reasoning, where contexts are matched with a context base (the Recipes) and linked to actions (the Rules). As Ranganathan et al., Syed et al.'s solution finds the right devices to support services by processing the contextual information linked to jobs, tasks, processes and devices; their work doesn't evaluate the role of the users in the service provision scheme.

The EasyLiving project [25] is a well-known project from Microsoft Research. About the service provision, the EasyLiving Geometric Model (EZLGM) [5] proposes a mechanism that determines which devices, in a given environment, can be used by a user during human-machine interactions and help in the selection of the right devices. The EZLGM provides some tips and ideas about user tracking and device selection based on the environment context. However, it doesn't take in account a more complex environment context with user capabilities, preferences, device resources and capabilities.

The European project *AMIGO* proposes a framework for smart environment that enhances the assistance of users through context-awareness. In the context of this project, Vallee et al. [29] propose a system to dynamically create end-user services through services composition. The service composition is initiated by abtracted plans, describing environment state/context the plans are responding to, which actions should be taken and the notification to the users. These different plan steps are matched with services in the environment through a composition manager. At some points, the user profile is considered by taking into account the possible handicaps of the users.

On the other hand, several work have been done on the adaptation of the user interactions according to the user capabilities and preferences or on the interaction modalities. For instance, the *Supple* project [9] proposes mechanisms to adapt visual interface based on the preferences of the users and history of utilizations.

Moreover, Sakurai et al. [24] worked on the modelization of the interaction in environments composed of several displays. In their work, they address the problem of the interactions (mostly the mouse's cursor) and visualization of the information in an environment where displays are disposed at different positions with different angles. To adapt the presentation of the information to the users, they take into account several parameters such as the distance between the users and the displays, the vision angles, etc. These parameters are definitely interesting for the provision of services in smart environment with several displays, as we are proposing in the next section.

The work contexts and results of *Supple* and Sakurai et al. are different than those we are aiming but target the same goals: provide better services and interactions to the users. They provide interesting perspectives and approach that motivate our work.

Our service provision solution addresses several issues from these related works. For instance, the implemented context-aware middleware takes into account the user interaction capabilities and preferences, where Trumler et al., Ranganthan et al., Syed et al., EasyLiving and even AMIGO solutions, did not managed

these data. The proposed solution takes into account a full mediated environment with several interaction devices, compared to the solution of Ghorbel et al. Finally, our choice of using fuzzy logic for service provision over other reasoning approaches, e.g., description logic (Ghorbel et al.), semantic matching (Ranganathan et al.) or case-based reasoning (Syed et al.), provide fast computing times, versatility and scalability. It also allowed us to rapidly designed first prototypes even in a context where it was difficult to model the reasoning process, due to a large amount of heterogeneous contextual information.

## 3. Service provision system

An intelligent service provision system allows dynamic, fast and adapted service deployment toward the users in the environments, based on the context of the environment, and takes into account different constraints such as the users' capabilities and their preferences. The main goal of the proposed service provision system is to support the deployment of the assistive services into the smart environments for other smart systems like activity recognition or errors and failures recognition systems [23]. These systems use the service provision functionalities by sending a deployment request to service provision system, by supplying the needed information related to the assistance that needs to be deployed: Which user?, Which software?, What are the software needs?, Is there a specific zone of the environment that is targeted by the assistance request? There are several benefits from encapsulating the service provision into a different system than the recognition software. By using a Service Oriented Architecture (in our case the OSGi framework and the Web Services), the service provision functionalities of the system can be used by several systems in a smart environment. Thus, the complexity of the provision's reasoning process are hidden for other environment's software (like in the Facade design pattern) and it is even possible to have several service provision systems (or services, thanks to the SOA mechanisms) for different kinds of provision needs.

### *3.1. Context-awareness model and strategies*

The service provision mechanisms use four main context's elements: the user profiles, the device profiles, the topology of the environment and the software profiles. Each service that needs to be deployed in the environment has their hardware, software or contextual needs. On one hand, assistive applications like a meal preparation assistant or a context-aware calendar and organizer, can target particular users in the environments and can require specific peripheral devices. On the other hand, users have physical capabilities and preferences about the environment devices; the devices also have a profile with capabilities e.g. their resources, connected peripheral devices, etc. Finally, all these components are present in the smart environments at different locations (or not) and are related to contextual zones e.g. the kitchen, the bathroom, the living room, etc. Figure 1 shows the different information used by the service provision reasoning engine.

Different strategies exist to process the contextual information of a given system or environment. For instance, it is possible to divide the context awareness of a given system in several sub-contexts called micro-contexts [4], related to specific devices or closed locations. A first reasoning process on the information of this micro-context can be done, abstracting or aggragating at some point the information, and later shared with the other micro-contexts of a system, enabling a micro-context awareness. At another level, it is possible to aggregate and combine the micro-contexts with other sources of information, forming a second level of information, the macro-context or macro-context awareness. Therefore, micro-context awareness revolves around the subjective perception and the understanding an environment entity has of its environment, while macro-context awareness is the global, emergent picture that components help build of entities in their environment [2]. In addition to dividing the contextual information in different layers, reducing to some point the coupling, another benefit of the micro and macro-context awareness is to prevent the divulgation of sensible information between software components.

In our work, we used this micro and macro model to divide in layers the reasoning on the contextual information related to the service provision. The service provision functionality used a reasoning engine, the Fuzzy Logic Organization Reasoning Engine (FLORE) [12], to match the needs of the services to deploy with all the related contextual information of the smart environments.

Fuzzy Logic [16] allows to "fuzzify" the contextual information into input i.e. transforming the numeric values into fuzzy values related to a quantitative set, compare and process them through a set of reasoning

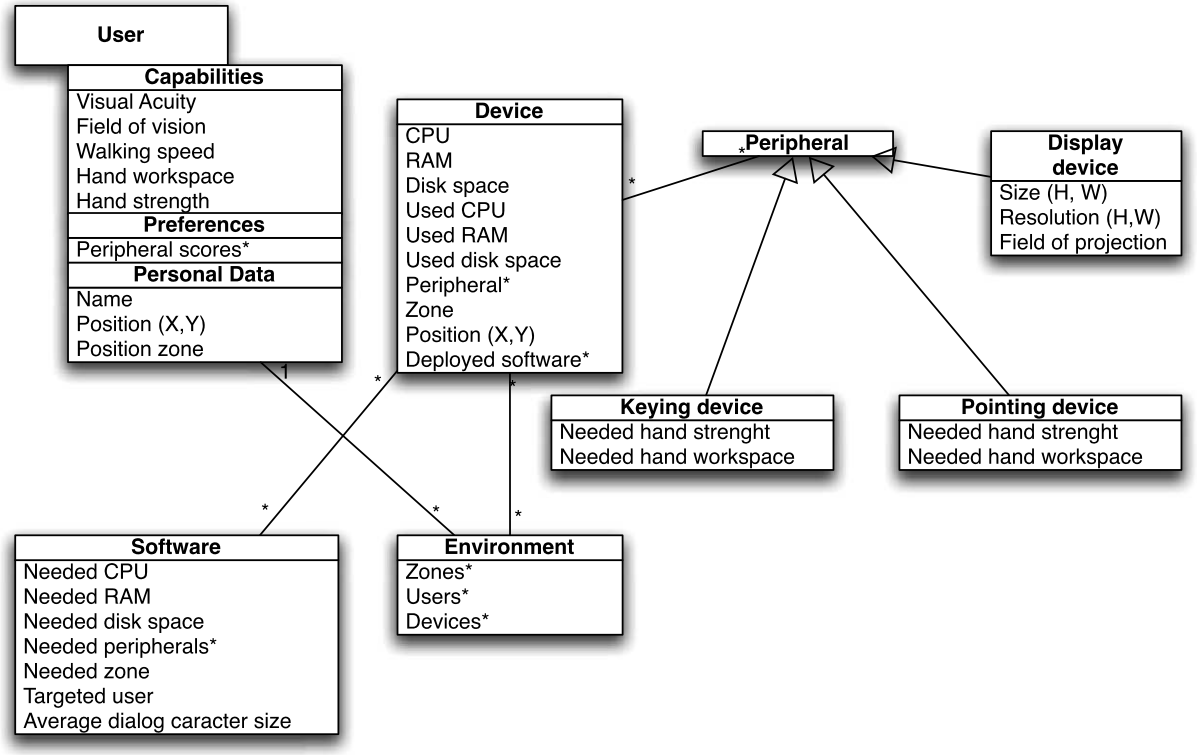


Fig. 1. Context information used in the service provision reasoning process.

rules then "defuzzify" the outputs in numeric values which can be used by the system. The contextual information of a pervasive environment cover a large variety of data type, ranging from quantitative information such as the room's temperature to qualitative information such as the user's state. Thus, the Fuzzy Logic approach allows to easily compare quantitative and qualitative information in a same set of reasoning rules. Moreover, a reasoning algorithm based on the Fuzzy Logic doesn't need to have an accurate knowledge of the model and can work with a high level of imprecision, which is the case of the smart environment, where it can be difficult to describe precisely the model and get accurate data. Finally, Fuzzy Logic gives us the support to describe a situation where no clear evaluations and statements can be carried out, like in the case where a device's resources are used or partially used or to evaluate the membership of the user's walking speed to a linguistic term such as slow or fast.

Fuzzy logic reasoning over the information involves three main steps: (i) *fuzzification* of the numeric inputs' values into linguistic terms using membership functions; (ii) *inferences of fuzzy rules* with previous linguistic terms, with three sub-tasks: aggregation (combining the results of the different predicates), activation (assignation of the rules' conclusions) and accumulation (combination of the conclusions to output fuzzy sets); (iii) *defuzzification*: conversion of the output fuzzy sets to a numerical output, where often a centroid method is used to find the average value of the corresponding defuzzification sets. Figure 2 presents an example of how fuzzy logic is used in a simplified case, where the system uses information about devices' resource consumption to identify the best deployment target. Our solution includes lot more rules and fuzzy sets to process contextual information around the user profiles, environment topology, etc.[1]

Therefore, with the distributed nature of the smart environments and in accordance with the micro and macro model, we divided the FLORE in two units, each one having their own fuzzy logic controller:

– the FLORE device unit (FLORE-D): related to the micro-context layer where the device resources and the connected interaction peripher-

[1]Full description of fuzzy sets and rules are accessible in Annexe 2 at http://tel.archives-ouvertes.fr/tel-00681885.

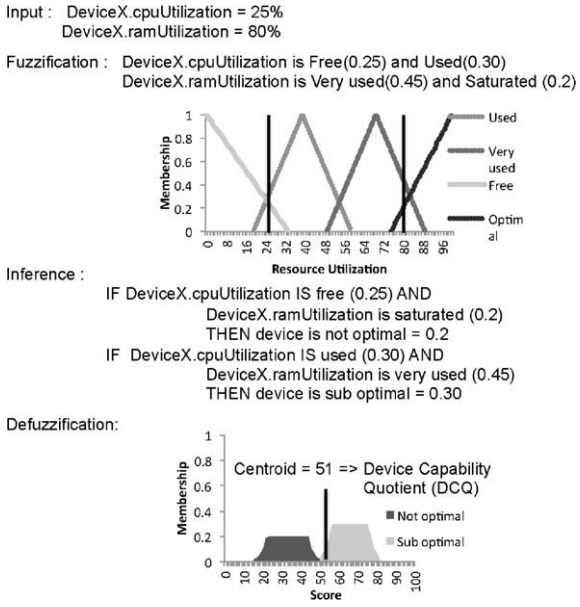


Fig. 2. Fuzzy logic example with device's resources consumption for a Device named X.

als are computed according to the service needs. The results of the micro-context computation are shared with the macro-context layer. Each non-dedicated devices in the smart environments (e.g. desktop computers, mobile phones, tablets or laptops) that can be used as a service provision platform are running an instance of the FLORE-D, along with other software components (see Section 3.7).

– the Flore coordinator unit (FLORE-C): related to the macro-context layer where the result from the FLORE-D are computed along with the user profile, the environment topology and component locations (extracted from the micro-context), and again the service needs.

To determine which device to deploy a service to, contextual information is processed using two implementations of a JFuzzyLogic controller[2], which are embedded in the FLORE-D and FLORE-C. Contextual information are fuzzified using membership functions, fuzzy rules (15 rules for FLORE-D and 41 rules for FLORE-C). The final output of the FLORE-C, ranging from 0 to 100, represents the Device Capabilities Quotients (DCQ), a metric value which represents the optimality of a device according to the general context of a service provision request and more specifically information linked to the user profiles.

The reasoning around the micro-context or the FLORE-D, as we briefly wrote above, concerns mainly the service needs versus the devices' resources and context. The services to provide can need different amount of Central Process Unit (CPU), Random Access Memory (RAM) or Permanent Memory storage (PMS); it can target a specific deployment zone and need specific interaction peripherals such as mouse, keyboard, tactile screen, display, etc. Thus, the FLORE-D computes a first DCQ, which will be used by the FLORE-C to compute the final DCQ along with the macro-contextual information. Moreover, other micro-contextual information are shared with the FLORE-C such as the device location and available interaction peripheral for the macro-context reasoning. Figure 3 presents the utilization and the exchange of the contextual information between FLORE-C, FLORE-D and a management tool, which is a software used by users to deploy services in the environments.

On the other hand, the FLORE-C process the information about the user profiles according to the service needs, combine the results from the FLORE-D and compute the final DCQ. To achieve that, the FLORE-C relies on reasoning on the user profiles using an approach based on the interaction modalities. The interaction modalities describe the general ways humans use to interact with their environments and with other beings [19]. These modalities correspond to the way we, as humans, collect the information about our environment (sense), perceive these information (perception), interpret and retain the information (cognitive) and finally, how we react physically to them (motor sense). In the context of the service provision, we use this classification on the user interaction capabilities and apply it to reasoning process. In the current version, the service provision system uses four interaction modalities:

– Sense – Vision: the field of vision of the users versus the computing devices and their display devices;
– Perception – Vision: the visual acuity of the users versus the application's information on the devices' displays;
– Motor Sense – User mobility: the user's mobility capacity versus the devices and peripheral locations;
– Motor Sense – Manual Interaction: the user's physical capacities versus the peripheral devices physical needs: hand force and hand workspace.

[2]http://jfuzzylogic.sourceforge.net

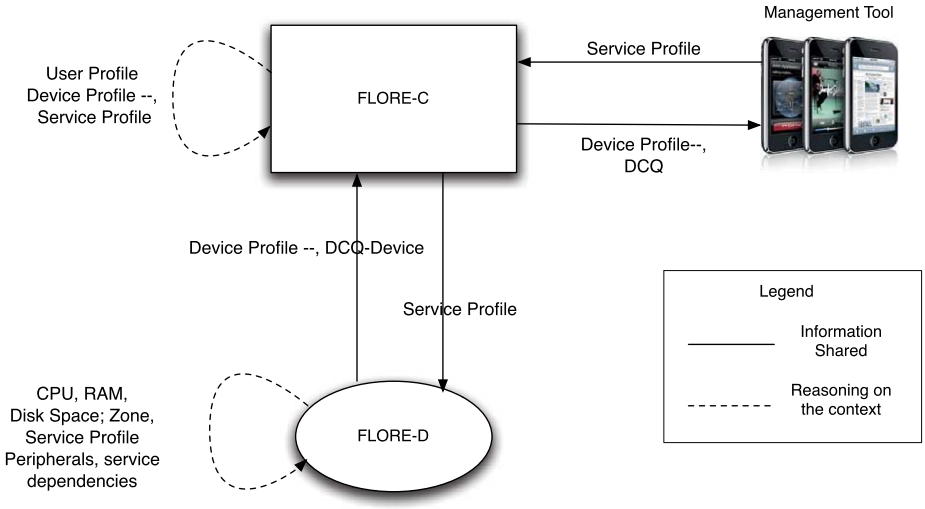


Fig. 3. Schema presenting the utilisation of the contextual information in the service provision system.

These modalities represent the traditional way to interact with computing devices: vision and the sense of touch. We could add to these modalities a fifth one, the cognitive modality related to the peripheral preferences of the users. However, including the preferences in the cognitive modalities is a subject to debate, as the cognitive modalities are more often related to the interpretation of the information, its comprehension and its retention. We would also, in our future work, add more interaction modalities such as the hearing acuity versus the speakers' volume.

Once the information related to these interaction modalities has been processed, abstracted and, in some cases, aggregated, they are injected into the FLORE-C. In this reasoning engine unit, each quantitative information is *fuzzified* into fuzzy universes where different membership functions allow transforming the quantitative information into qualitative information. For instance, we based the membership functions related to the users' mobility on several research on the average human walking speed (more details in the Section *Motor Sense – User mobility processing*). Then, these qualitative data are computed by a series of fuzzy rules, which compute the *Device Capabilities Quotient* with the help of defuzzification functions. For instance, the fuzzy rule: *IF visionAngle IS mutual AND visualAcuity IS optimal AND userMobility IS fast AND walking time IS instant AND peripheralsPreference IS liked THEN deviceEvaluation IS optimal WITH 1*; compute the different qualitative information related to the interaction modalities presented before and assign the result to the defuzzification function *optimal*. In the current version of the system, the FLORE-C has 41 fuzzy rules to compute the DCQ, while the FLORE-D has 15 fuzzy rules to compute on the micro-contexts. Finally, the centroid value of the 41 rule results, in the defuzzification universe, corresponds to the device's DCQ.

### 3.2. A service provision example

Before explaining in detail the four interaction modalities processing and the utilization of the user's peripheral preferences in the computation of the DCQ, a brief example illustrating a service provision would help to assimilate the information presented to this point and understand the next sections of this paper.

Suppose that an inhabitant from a smart apartment is standing at the entrance of its kitchen around lunchtime. This inhabitant suffers from cognitive deficiencies that affect his time organization. Thus, to remind him to prepare his meal, his electronic agenda requests to the system to provision a meal preparation assistant to the user in the kitchen area. The other information contained in the profile of the inhabitant are: the user has a poor visual acuity and an average field of vision, he moves at an average speed, he has a good hand strength and workspace, and he prefers the tactile screens to the mouse peripherals and keyboards. The meal preparation assistant doesn't need great resources: a display to present its interface and a pointing device. In the best case, this software should be deployed in the kitchen zone.

On the other hand, the smart apartment is divided in to several zones, e.g. the kitchen area, the living room,

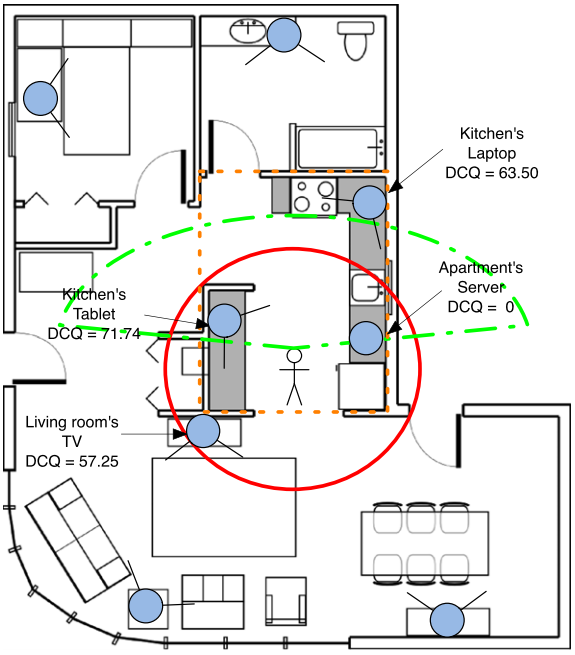


Fig. 4. An example of a service provision with some of the processed contextual information, user's mobility (circle), user's visual acuity and field of vision (arc) and targeted zone (rectangle).

etc. Several devices and their interaction peripherals are located in these zones. Especially, four devices are in the proximity of the user: a laptop at his one o'clock, a tablet at his ten o'clock, a server in a closet at his four o'clock and finally a TV with its multimedia computer behind him in the living room. Each of these devices have their own resources and different kinds of interaction peripherals. Figure 4 illustrates this example with a map of the smart apartment. In this figure, some of the interaction modalities are shown, such as the user's visual acuity and his field of vision (the arc), the user's mobility corresponding to a walking time of two seconds or less (the circle). The kitchen zone perimeter is also indicated (the rectangle). Logically, the most suitable device in this context corresponds to the device in these three zones: the kitchen tablet. However, several other contextual information can change this logic, depending on the preferences of the user or the resources' utilization of the devices.

Once the contextual information has been submitted to the service provision reasoning engine (FLORE), the different DCQ for the environment devices are computed. In this example, the device with the highest DCQ is, effectively, the kitchen tablet with 71.74 points, followed by the kitchen laptop with 63.50 points, the living room TV with 57.25 points and the apartment's server with 0 points. These examples' scores correspond in fact to one of the validation scenarios presented in the *Validation and results* section. The kitchen laptop received a lower score due to the distance between the user and the display versus the user's visual acuity. Moreover, the user prefers to use a touch screen (like that of the tablet) rather than the keyboard or the touch pad of the laptop. The living room TV is situated behind the user and outside of the kitchen area, which both reduce its DCQ score. Finally, the apartment's server did not have a connected display and a pointing device, which fail in providing these peripherals to the meal preparation application, and caused a DCQ attribution of 0 points. Therefore, the most suitable device to support the user and the meal preparation assistant in this context is the kitchen tablet.

The above example will be used to explain in details, the interaction modalities and the user peripheral preferences processings. Thus, the next sections will present the four interaction modalities computing and the user peripheral preferences processing, starting with the user's vision processing related to the sense and perception modalities.

### *3.3. Sense and Perception – Vision processing*

On one hand, based on the user field of vision (or field of view, FOV), the users orientation and his location in the environment, the service provision system verifies which displays in the environment the user can see and interact with. On the other hand, device's displays have fields of projection, which correspond to the area where it is possible to clearly see the presented information. Therefore, the field of view evaluation requires a double verification: find which device are viewed by the user and verify if he can see the information projected by the display.

Therefore, the system uses the user's FOV angle, its location in the environment based on a Cartesian coordinate system and its orientation angle to detect which device's displays are in the user's field of vision. For the detected displays, the system verifies if the user is in the displays' field of projection, which is the same process as with the user's FOV. All these information are processed with Algorithm 1, where a change in the orthonormal base is made with the user as the new center of the orthonormal plan and the center of his FOV as the y' axis. The devices that are in the user's FOV will be preferred as deployment targets.

**Algorithm 1** Algorithm of the mutual field of vision/projection checking

1 – Compute the translation matrix where u = User

$$M^t = \begin{bmatrix} 1 & 0 & u.pos.x \\ 0 & 1 & u.pos.y \\ 0 & 0 & 1 \end{bmatrix}$$

$$M^r = \begin{bmatrix} cos(u.\Theta) & -sin(u.\Theta) & 0 \\ sin(u.\Theta) & cos(u.\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$M = M^t * M^r$

2 – For each display devices, re-compute its position and orientation and verify if the user have one of the displays in its field of vision

$\forall d = \in Displays$

$d.pos = d.pos * M$

IF $d.pos.y \geq 0$ THEN

$m_{u,d} = d.position.y/d.pos.x$

$\Theta_{u.fov.right} = (90 + u.fov.right) * (\pi/180)$

$\Theta_{u.fov.left} = (90 - u.fov.left) * (\pi/180)$

IF $d.pos.x < 0$ OR $d.pos.y < 0$ THEN

$\Theta_{u,d} = \pi - |\arctan(m_{u,d})|$

ELSE $\Theta_{u,d} = \arctan(m_{u,d})$

IF $\Theta_{u,d} \geq \Theta_{u.field.right}$ AND $\Theta_{u,d} \leq \Theta_{u.field.left}$ THEN the display device is in the field of vision of the user, re-do the step 1 and 2 with the display device as the center of the plan. If the user is also in the field of projection of the display, then the two entities have a mutual field of vision/projection.

For the user's visual acuity versus the display devices, the service provision system uses the Snellen test [17] to verify if the user is able to read the dialog texts of the application GUI. The Snellen test allows quantifying the users' capability to read characters and texts at specific distances. The normal visual acuity is qualified as 20/20, which means that the users are able to read five arc minutes characters (called an optotype) at 20 feet (6 meters). Someone with a visual acuity of 18/20 is then able to read the same optotype as someone with a visual acuity of 20/20, but at 18 feet. The users' visual acuity evaluation also uses the average dialog text size of the application GUI (in pixels), the display resolution (in pixels) and the display size (in meters) to compute a ratio between these data, the visual acuity of the user and its distance with the environment displays. Algorithm 2 presents the way these data are computed to find the visual acuity ratio.

**Algorithm 2** Algorithm of the visual acuity processing

$\forall d \in Devices | s \in Software,$

$u \in Users, s.user = u :$

$deltaX = u.position.x - d.position.x$

$deltaY = u.position.y - d.position.y$

$distance = \sqrt{deltaX^2 + deltaY^2}$

$perimeter = 2 * distance * \pi$

$caracterSize = d.size.height * \frac{s.caractSize}{d.resolution.height}$

$optotype = 5 * \frac{perimeter}{arcminutes} * \frac{averagevisualacuity}{u.visualAcuity}$

$ratio = 1 - \frac{optotype}{caracterSize} \mid ratio = [0, 1]$

$return\ ratio$

Therefore, more the ratio is going toward 1 and more the application dialog text is easy to read for the user on the device. Thus, this device will be preferred as an application deployment target. The maximum ratio value is limited to 1, corresponding to a readable character size by the users. Not limiting the ratios would tend to give non desired preferences to the large displays with low resolution.

Finally, the results of the evaluation of the field of vision of the users and the visual acuity ratio are transmitted to the FLORE, where they are matched with membership functions (visionAngle = mutual or behind functions, visualAcuity = optimal, borderline, hard or impossible to read functions) like in the rule example showed in the previous section.

### *3.4. Motor Sense – User mobility processing*

The service provision system evaluates the suitable devices from the environments based on the distance between these devices and the targeted users by the service provision requests. Thus, this evaluation takes into account the average walking speeds to compute the time users would eventually spend to reach the environment devices. Moreover, the service provision system categorizes also the users' average walking speed using membership functions (Fig. 5). These categories serve to discriminate the devices that need important walking time to been reach, e.g. users with mobility problems, and favor the devices at a short distance and at quick walking time.

This double evaluation system (walking time and walking speed category) is more adapted to people with mobility problems, as slow walking speeds are generally synonym of physical efforts [3], especially for older people. For instance, the system will favor devices at a quick walking time (0–2 seconds), normal

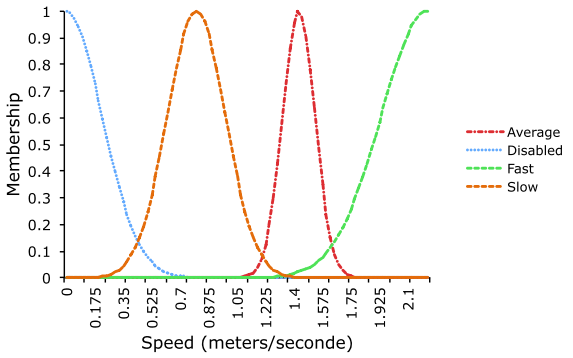


Fig. 5. Membership functions of the user mobility based on their average walking speed.

walking time (10 seconds and less) and even at long walking time (10–20 seconds) for people with a fast walking speed (for which there is not a real physical effort) and will favor quick or normal walking time for people with a slow walking speed.

The reasoning process on the mobility is processed, like for the other interaction modalities, by the FLORE fuzzy rules. The fuzzy membership functions for the user speed and mobility, i.e. the user speed qualification (Fig. 5), are based on data from Kan et al. [3] and Carrey et al. [7] that present an evaluation and a qualification of the walking speed, respectively, for elderly people and pedestrians in an urban setting. These work gave enough scientific information on the average walking speed to create our categorization system that includes two types of potential users (elders and average pedestrians). To represent these categories, we used Gaussian functions with a normal distribution, where the means and the standard deviations (Table 1) were taken from Kan et al. and Carrey et al. For instance, the average walking speed of elderly people with a moderate mobility is 0.8 meter/second with a standard deviation of 0.18 meter/second, which we refered to the *slow category*. The average walking speed of the pedestrians was refered to the *average category*. For the *fast category*, the average speed is 2.22 meters/second, with a standard deviation of 0.285 meter/second. Users with a walking speed above this average speed receive a membership value of one, as there are no other speed categories above fast.

### 3.5. Motor Sense – Manual Interaction processing

To address the users' physical capabilities according to the device peripheral physical requirements, the service provision system process the manual interaction modality in two steps. First, the system checks if the targeted users are able to interact with the device peripherals by verifying if they have enough hand strength to manipulate the peripherals, such as pressing the buttons of a keyboard or moving a mouse. Secondly, the system verifies if the targeted users are able to open their hands enough to interact with the objects. Some users with physical impairments could have difficulty to interact with peripherals, such as older people with arthritis. If these two physical capabilities are not met by the device's peripherals, the evaluated device is discarded and directly received a DCQ of 0 points. This evaluation system, based on the hand strength and the hand workspace, is based on the results of the work of Kadouche et al. [14] on an evaluation of the interaction of people with physical deficiencies.

Table 1

Means and standard deviation of the average walking speed membership functions

| Function | Mean (m/sec) | Standard deviation (m/sec) |
|---|---|---|
| Disabled | 0 | 0.22 |
| Slow | 0.8 | 0.18 |
| Average | 1.43 | 0.1 |
| Fast | 2.22 | 0.285 |

### 3.6. Peripheral preferences processing

The user's capabilities are an important factor in finding the most suitable device for a service provision. At the same time, users also have preferences towards some devices that should not be neglected, as they may prefer a specific display size or keyboard disposition. In the context of the service provision, these preferences have the role to favor the devices with interaction peripherals that are appreciated by the users. On the other hand, they can penalize the devices with unappreciated peripherals. These preferences can be used as a complementary tool to the users' manual interaction modalities in order to determine which peripherals the potential users will be able to use.

The service provision system defines the peripheral preferences by using a Likert scale system [15]. The utilisation of the Likert scale has been inspired from the user preference evaluation in GUI usability [18], where it is often used. Therefore, prior to the service provision request, each user of the smart environment defines their preferences toward their interaction peripherals (e.g. keyboard, touch screen, mouse) in their environment that they can interact with, by us-

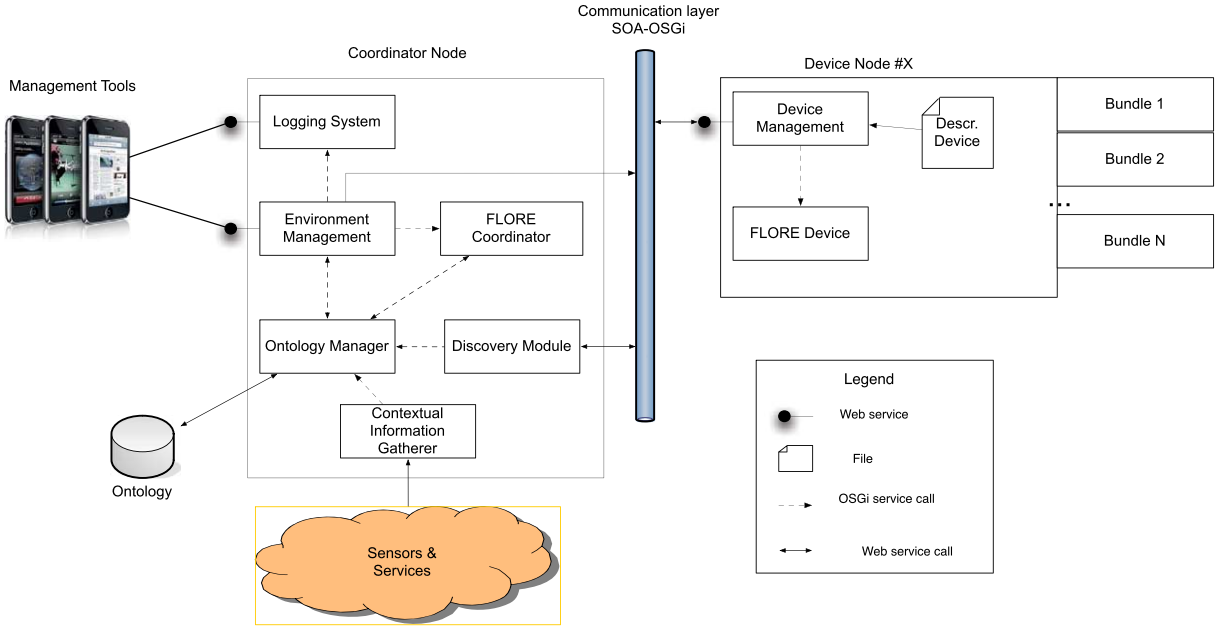


Fig. 6. The service provision system architecture with its two main components: the Environment Management Coordinator and the Device Node.

ing three values: the user likes the peripheral device $= 1$, the user is neutral in using this peripheral device $= 0$, the user dislikes to use the peripheral device $= -1$.

The evaluation process computes the overall preference value of each device, by computing the user's peripheral preferences. Higher is the overall value, more the user appreciates the device. The overall preference value is then transmitted to the fuzzy system and used in the evaluation of the fuzzy rules. This method of computing the preferences is pretty straightforward, but works in most cases.

To avoid the description of the user's preference for each environment peripheral devices, the service provision system uses the inheritance mechanism implemented in the context description (see the section *Implementation*), to define general preference for the main types of interaction devices (e.g. mouse, keyboard, touch screen). If no preference is defined for a given peripheral device, the system uses the inheritance to find the first peripheral in the inheritance tree where the preference is defined.

### 3.7. Implementation

The service provision system is built over the OSGi framework. OSGi is a Service Oriented Architecture (SOA) framework that gives the support for the modularization of the ubiquitous applications and the management of their life cycles. Thus, the service provision system uses these functionalities to deploy and manage the service modules in the environment's devices. As the proposed system is a distributed framework, the Apache CXF dOSGi and WS-Discovery are used as communication support between the device and the coordinating device, which host the service provision reasoning engine (FLORE).

On the top of the OSGi framework, we have implemented several modules that are cooperating to provide the service to the users in the smart environment (Fig. 6). The Environment Management Coordinator node has the job to manage the device discovery, maintain the environment ontology, receive the service provision requests and manage them using the FLORE. The Device nodes are deployed on the environment devices and host the services that are delivered to the users. They also perform some reasoning on the context such as on the hardware resources and the interaction peripheral availabilities.

As the provision functionalities use contextual information and user profiles to find the optimal way to deliver the services to the users, a way to describe and contain the contextual information is needed. A powerful way to represent this information is by using semantic language, describing the environment information and connections residing between the concepts.

The service provision framework describes the ubiquitous environment through a meta-ontology [1] described in the Web Ontology Language (OWL), which presents the context information through Resource Description Framework (RDF) concepts and semantic connection in OWL format. To hold and maintain this information, the system uses the JENA framework.

The fuzzy logic controller used by the service provision system was implemented over the JFuzzy Logic API, which uses the Fuzzy Control Language (FCL IEC 61131) to define the membership functions, defuzzification functions and the fuzzy rules. Among other benefits, the FCL allows a clean and easy implementation of the fuzzy logic controller, and allows the proposed system to be rapidly scalable and adaptable to new contexts, by introducing new rules and functions.

The service provision requests that are sent to the framework are received through web service calls, then forwarded to the service provision reasoning engine. These requests can be sent by any other environment's application and by the management tools.

Concerning the retrieval of the contextual information in the environment, one part of the data is manually filled by experts in the macro environment description, e.g. environment topology or user interaction capabilities (visual acuity, hand strength, etc.). On the other hand, experts also provide some of the information about the device profiles, e.g., displays sizes, resolutions, orientations or connected interaction peripherals. In our current prototypes, we didn't use an interior location system to locate the devices in the environment, even if such technologies exist. Some of the used devices didn't use wireless technologies, e.g. the multimedia PC in the living room, and would not be compatible with such systems. However, the users' locations/orientations and the devices' resources (CPU, RAM and disk spaces) are dynamically retrieved through services in the environment. For instance, the users' locations are retrieved with the *Dinamo* application [21] and devices' resources with the *Hyperic System Information Gatherer* (SIGAR) API[3]. In a future version of the prototype, more contextual information could be retrieved such as the connected interaction peripherals, but it could be relatively complex by taking into account the heterogeneity of the operating systems and hardware.

[3]http://www.hyperic.com/products/sigar

## 4. Software self-organization in smart environment

We presented in the previous section the different mechanisms used by the service provision system to provide a service at any time in a smart environment. However, in a specific context, several services (inter connected or not) could be required to be deployed at the same time. Therefore, we developped on the top of the service provision system a second layer allowing to send to the system macro requests containing several service provision requests.This kind of request allows the deployment of complex software composed of several software components on the devices of the smart environments. For instance, such macro request could contain the deployment of end user applications such as an in-house light manager on a mobile device and the back-end light controler service on a server. Since the goal of the macro request system is to organize strategically the deployment of the software components based on the contextual information described in the previous sections, we therefore discuss of software self-organization.

Thus, the service provision system uses the resulting DCQ score of FLORE to classify and find the optimal device according to the software profile and the environment context. It is possible to find the optimal organization for a given list of applications to deploy in an environment by running the FLORE for each application and by selecting as deployment target the device with highest DCQ score for each application. Table 2 presents an example of such self-organization reasoning outputs and its best results (in bold).

As the FLORE takes in account the resources' utilization of the software to deploy and each software is consuming resources, the evaluated DCQ scores could change if previous deployment was done on related devices. For instance, as Device 1 has the highest DCQ score for the Application 1, this device would be choosen as target for the application. As the deployment is consuming resources, its new DCQ scores would change (e.g. in the case of the new DCQ score for Application 2 – Device 1 would be 75). At this point, the first optimal solution would be false as Device 3 is now the most suitable device for Application 2. By choosing Device 3 for Application 2, it also could change the DCQs for Application 3 (and so on for the other application to deploy), resulting each time in a possible cascade of DCQ score modifications.

Thus, the initial selection technique of Fig. 2 cannot be applied to all cases. Such a problem is a classic

Table 2
Example of self-organization DCQ output for 3 given software

| Devices | Soft. 1 | Soft. 2 | Soft. 3 |
|---|---|---|---|
| Device 1 | **80** | **79** | 0 |
| Device 2 | 79 | 48 | 80 |
| Device 3 | 0 | 78 | **82** |

case of Resource-constrained project scheduling problem (RCPSP) [26], the optimal solution can generally be found with an algorithm of NP complexity in polynomial time O($n^k$) where n is the number of devices and k is the number of applications to deploy.

The implementation of a Resource constrained scheduling algorithm for our context would be difficult, if not, impossible. As the DCQ score is partially based on the hardware and the resource consumption at a specific time, it would be difficult to evaluate a future resource consumption as it would be required for a RCPSP algorithm. Therefore, we propose a simple and straight forward solution that is probably the most intuitive. For each application to deploy in a macro request, we find the device with the highest DCQ score and deploy the application. Thus, resources will be consumed and the next DCQ evaluation, for the following application in the macro request will take into account the states of the devices' resources.

This algorithm has several drawbacks: it doesn't find the theoretical optimal organization and establish a priority hierarchy between the applications to deploy. The first application in the list will have priority on the device's resources and the last application will have the remaining device's resources. However, if it is well used, this priority can be useful, allowing important or critical applications to have higher priorities than other applications. Finally, the overall computing time is faster than the optimal solution with O(kn).

A second benefit of such algorithm is the management of the functionnal dependancies that can exist between software components described in the macro requests. Through the use of the *OSGi Bundle Repository* (OBR) API, the provision system is already managing the deployment of several bundles interconnected through code dependancies. This API resolves the local dependencies of a software on a single device, but doesn't support the functionnal dependencies of a software toward remote services.

In SOA architecture, service dependencies are generally verified at runtime, allowing programmers to code different reactions to the availabilities or the non availabilities of services. For instance, when starting, an application could wait for the availability of a specific service before being fully started or simply wouldn't start if a specific service is not available. In the proposed solution, by keeping in order of dependencies the application to deploy in the macro requests, the dependencies can be fullfilled, at the price of the performance (if it's the case) for the last application in the list.

At this point dependencies between software components add another level of complexity. Additional work on the optimization of the software organization will be done in future work. In the perspective of adding to the self-optimization mechanisms, such as load balancing between the device nodes, a more complete solution will be required that should implement RCPSP and software dependencies algorithms.

## 5. Validation and results

During and after the development of the service provision system, we intensively tested and evaluated the proposed solutions through unit tests, scenario validations and performance tests. The whole project contains more than 25 unit test classes, testing different bottleneck points in the system. However, in the context of this paper, presenting these tests would not bring any contributions to the scientific community. Thus, we will focus in the following sections, on the scenario validation, the performance tests and their results. Moreover, we specify that the main goal of our object validation phase was to validate the technical feasability of the system rather than to evaluate the impact of such system on real users. Of course, we believe in the importance of involving users in field studies to evaluate the impact and acceptance of the technologies. As our work's problematic was fairly complex, we believe it was critical to propose an adaptable service provision solution and to evaluate it, prior to evaluating the system with real users. One benefit of such strategy is to rely on a functionnal prototype for extended validation instead of relying on *wizard of oz* techniques. In our future work, thanks to the results we are presenting in this section, we will work on an evaluation to validate the proposed solution with real users in the context of everyday life activities. The results of this evaluation will be presented in a future publication dedicated to the human-computer interaction aspects of our work.

Therefore, while we were planning the validation phase of the service provision system, we set out three objectives to be achieved by the service provision func-

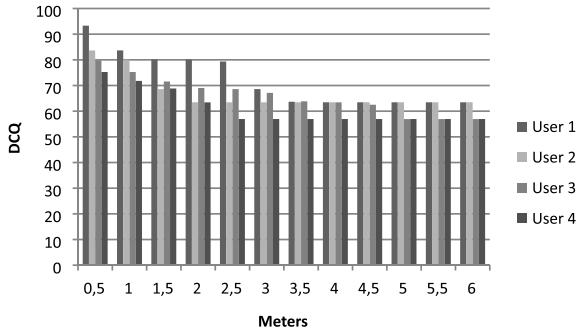


Fig. 7. Results of the validation of the visual acuity and user mobility evaluation.

tionalities. Our first objective was to verify if the provision system was giving the optimal solution according to the environment context and the user profiles i.e. the highest DCQ to the device in the optimal context. Our second objective was to verify if the system was handling the provision requests in a decent interval time. Finally our third objectives was to assure that the system gave stable processing time, reproducible results and small DCQ variation between similar context.

First, we validated the visual acuity and user mobility evaluations by testing the DCQ attribution for a given device, which we moved to different distances from users according to their capabilities. Contrary to the other validations presented in this section, we used a simulated environment where we placed four hypothetical users with variations in their visual acuity and walking speed:

- User 1: Visual Acuity of 20/20 and an average walking speed of 1.43 meters/second;
- User 2: Visual Acuity of 10/20 and an average walking speed of 1.43 meters/second;
- User 3: Visual Acuity of 20/20 and an average walking speed of 0.4 meters/second;
- User 4: Visual Acuity of 10/20 and an average walking speed of 0.4 meters/second.

For these tests, the simulated device display was of 15 inches, with a resolution of 1920 per 1080 pixels, and the provided service's interface had an average character size of 30 pixels. We can see in the validation results (Fig. 7) that the DCQ scores are stabilizing around 2.5 and 3 meters from the users, where the visual acuity ratio usually tends to zero. Between User 1 and User 2, where the visual acuity is reduced by the half, the results shows that the DCQ varies by about 10 points in the first 3 meters. Between Users 1 and 3, results show that the average walking speed has a more important impact on DCQ attribution than visual acuity, with a drop of 14 points, but the variation thereafter is less than for the visual acuity. Finally, User 4's result presents the combined effect of a low visual acuity and a slow walking speed. Compared to the three other users, it is with User 4 that the DCQ scores are the lowest, which demonstrates that the service provision reasoning succeeded in combining different interaction modality evaluations together (User 2's context and User 3's context). This first validation proves that the visual acuity and the user mobility are taken into account in the DCQ attribution and that the distance between the users and the device has a direct impact with the DCQ scores.

Then, to validate our first objective, we laid down a series of ten scenarios grouped in two categories. The first category concerned placing different user archetypes, with specific user profiles, in a smart apartment and sending a service provision request to the framework for these user archetypes. Then, we analyzed the returned DCQ from the service provision and compared the results between the different types of users. The second category of validation scenarios was to test the impact of the peripheral preferences on the service provision reasoning process and the DCQ attribution. Thus, we defined five types of users with different preference settings and we analyzed the returned DCQ. In both categories, the scenarios were implemented in the fully functionnal smart apartment of the DOMUS Laboratory of the *Université de Sherbrooke*[4], with colleagues playing the role of the user archetypes, as real users (e.g. PwSN) were not needed for these technical and functionnalities validations. The ten validation scenarios were implemented over the eight devices in the apartment (Fig. 4), with four devices in the proximity of the user:

- a Mac Book Pro laptop (Intel Core 2 Duo, Mac OS X and Java 6);
- an HP Tablet (1Ghz Intel processor, Ubuntu and Java 6);
- a PC server (Intel Quad-Core, Debian and Java 6);
- a TV multimedia server in the living room (Intel Core 2 duo, Windows 7 and Java 6).

A fifth one, the user's smart phone (Android OS 2.3) was located on a table in the living room. These devices as been placed in the apartment like in Fig. 4.

[4]www.domus.usherbrooke.ca

Because of compatibility problem between the Apache CXF/WS-Discovery API and Android, we rather used a version of the webservice bundle based on KSOAP for Android and jSLP for the device discovery. All these devices stayed on the same status for the duration of the scenarios, in order to not interfere with the user profile processing. For each scenarios, we deployed a fresh system in the smart apartment, with the specific user profile included in the Macro description of the environment. We also wait 20 secondes before sending the service delivery request, giving time for the device nodes to be discovered by the coordinator node and update the environment ontology by uploading their OWL/RDF description.

For the first categories of validation scenarios, we designed five user archetypes:

- average user: has a normal visual acuity, an average walking speed, a normal field of vision and a good hand strength and workspace;
- myopic user: has a low visual acuity (10/20) and normal capabilities for the other aspects of the profile;
- narrow vision user: has a narrow field of vision (60 degrees) on the left side and normal capabilities for the other aspects of the profile;
- limited hand strength and workspace user: has a hand strength of ten kilos and a workspace of 60 degrees;
- limited mobility user: has a walking speed of 0.8 meter/second and normal capabilities for the other aspects of the profile.

These five users were placed at the same position and orientation in the smart apartment, standing at the entrance of the kitchen, near the living room, face to the kitchen oven (like in the previous example, Section 3.2). Then, we sent to the system a service provision, a request describing the deployment of an assistance service (a *post-it* application) aiming these 5 users. The results of the service provision are presented in Table 3.

Table 3
Results of the validation with user archetypes

| User profile | Comp. time (sec.) | Results | |
|---|---|---|---|
| | | DCQ | Device |
| Average user | 0.549 | 75.56 | Kitchen Tablet |
| | | 63.50 | Kitchen Laptop |
| | | 63.18 | Living room TV |
| | | 45.40 | Smart phone |
| Myopic user | 0.534 | 71.74 | Kitchen Tablet |
| | | 63.50 | Kitchen Laptop |
| | | 57.25 | Living room TV |
| | | 45.40 | Smart phone |
| Narrow vision user | 0.540 | 75.56 | Kitchen Tablet |
| | | 66.28 | Living room TV |
| | | 63.46 | Kitchen Laptop |
| | | 49.93 | Smart phone |
| Limited hand strength user | 0.285 | 76.89 | Kitchen Laptop |
| | | 66.31 | Living room TV |
| | | 49.93 | Smart phone |
| | | 0.00 | Kitchen Tablet |
| Limited mobility user | 0.510 | 75.84 | Kitchen Tablet |
| | | 65.38 | Living room TV |
| | | 63.43 | Kitchen Laptop |
| | | 35.67 | Smart phone |

We can see in the results (Table 3) that the DCQ decreased between the average user and myopic user. This can be explained by the lower visual acuity of the myopic user, which reduced the visual acuity ratio computed for the kitchen tablet and the living room TV. The DCQ values remained the same for the other devices as they were already outside of the visual acuity limit. For the narrow vision user, the kitchen tablet received the same DCQ score as in the average user scenario, as the visual acuity and field of vision of the user allow him to see the device's display. However, the kitchen laptop dropped to the third rank, below the living room TV. This is due to the fact that the laptop's display is situated outside of the user's field of vision, like for the TV that is, however, situated at a closer distance.

For the limited hand strength and workspace user scenario, the user was unable to interact with the tablet's peripherals, caused by the physical capacities needed to use them. Thereby, the kitchen laptop received the highest DCQ and was choosen by the service provision system as the host for the assistive service. Finally, in the case if the user archetype with a limited mobility, the two closer devices (the tablet and the TV) received the highest DCQ. The laptop ranked at the third place and the smart phone DCQ dropped by twenty-two percents, both caused their distance to the user. As we were expecting, the service provision system found the right setting in the five cases, attributing the highest DCQ to the devices with the best configurations according to the provision contexts.

The second scenario category validated the users' preferences evaluation. We sent several service provision requests to the system, where five users with dif-

Table 4
Results of the service provision with user preferences

| Preferences | Comp. time | Results | |
|---|---|---|---|
| | (sec.) | DCQ | Device |
| Love all | 0.330 | 79.74 | Kitchen Tablet |
| | | 79.57 | Kitchen Laptop |
| Neutral | 0.506 | 79.84 | Kitchen Tablet |
| | | 79.68 | Kitchen Laptop |
| Dislike all | 0.610 | 77.16 | Kitchen Tablet |
| | | 76.10 | Kitchen Laptop |
| Dislike touch | 0.710 | 79.74 | Kitchen Tablet |
| screen | | 76.10 | Kitchen Laptop |
| Dislike phys. | 0.488 | 79.57 | Kitchen Tablet |
| interface | | 77.16 | Kitchen Laptop |

ferent peripheral preferences were present at the same location and orientation in the environment. In all of our experiments, the results of these scenarios are the most varied (Table 4). Between two devices where peripherals were liked and disliked, the variation in the DCQ score was low, i.e. a mere 3.64 points. If this gap can favor a device versus another one, we did not find that this difference was well good enough to make a real difference in every context. Moreover, in the *dislike all* scenario, we were expecting to get DCQ under 40 points, as the user dislikes all the peripherals. However, the weight of the negative preferences in the service provision reasoning didn't reduce the DCQ as we expected. Some modifications to the fuzzy rules were required to give to the FLORE the desired behaviors. After modification of the fuzzy rules and membership functions related to the user preferences (which took just a couple of minutes and few tries), we were able to run the scenarios and successfully obtained the expected results. Thus, the devices that have overall preferences corresponding to the dislike category received lower scores, under 40 points for devices with an average performance, and the variation in the DCQ was higher with 5.75 points between the liked and disliked scenarios. In that case, the fuzzy logic and more particularly, the JFuzzyLogic API, demonstrate their performances in rapidly improving a complex reasoning system.

Concerning the second objective on computing time, our first requirement was to obtain results faster than the user's experienced response time [6], i.e. around 1 or 2 seconds. As the service provision system can be used by professional caregivers to provide services to their patients or by smart environment managers, one of the requirements of the system was to provide service in a way that time to deliver services to users should be perceived as being instantaneous. Moreover, the second requirement was to provide service in lapses of time briedf enough to ensure a fast assistance to the user (for a given smart environment context). The average computing time for the ten scenarios was 0.5 second (network, system latencies and webservice calls included), well under the user's experienced response time and convenient for a fast assistance. The coordinator node with the service provision reasoning engine, which was doing the major part of the reasoning process, was ran on a Core 2 Duo 2.4 GHz, with 2 Go 667 MHz RAM and the J2SE 1.6 Java virtual machine. Moreover, we used the JENA framework version 2.6.2 with the file loading and persistence features.

Finally, the service provision systems produced stable processing times. The results were reproducible for a given context and the DCQ variation was low between similar context, e.g. in the case of the preference evaluation of the *like all* user and *neutral all* user, even if we would have like more variation for this case.

## 6. Conclusion

Today's smart environments are synonym of context-awareness, component dynamism and adapted services to assist the users in their daily living activities. Intelligent and easy to use systems are needed to provide services and software to the users in their natural environments, for instance to assist them in the realization of a task. We presented in this paper a service provision system for smart environements based on the interaction modalities. This system uses the contextual information on the smart environments and user profiles to find the most suitable device to host services and software that need to be provided to the environment's users. A fully functional prototype of this system has been implemented and deployed in a real smart apartment. The innovation of our work resides in the specification, the implementation and the integration of the user interaction modalities (e.g. the visual acuity, the user's field of vision, the user's mobility, the interaction capabilities) to the reasoning processes on the contextual information of the environment. By taking into account this information, the proposed system perform adapted service deliveries on the smart environment's devices.

In light of the validation results, the service provision system found the most suitable device in every case and computed Device Capability Quotient (DCQ) score in accordance with the context of the environment, i.e. the scores are respecting the performance and the situation of each device according to a given service provision request and the current context of an environment. Moreover, the related processing time are meeting our objective, allowing fast service provision.

With the broad utilization of smart environments and their expansion toward smart cities, such system will be increasingly more on demand. However, several improvements are needed to deploy our prototype into real clients' environment. In our future work, we aim to add more modalities, such as the cognitive modalities or the hearing acuity. As we are extending the service provision system, more validation scenarios will be required. The next step of our validation process will be to test the proposed system with dependant users in smart space settings, verify the impact of the system on their daily activities and evaluate the adoption of the system by the users.

## References


[1] B. Abdulrazak, B. Chikhaoui, C. Gouin-Vallerand and B. Fraikin, A standard ontology for smart spaces, *International Journal of Web and Grid Services* **6**(3) (2010), 244–268.

[2] B. Abdulrazak, P. Roy, C. Gouin-Vallerand, S. Giroux and Y. Belala, Macro and micro context-awareness for autonomic pervasive computing, *International Journal of Business Data Communications and Networking (IJBDCN)* **7**(2) (2011).

[3] G. Abellan Van Kan, et al., Gait speed at usual pace as a predictor of adverse outcomes in community-dwelling older people an international academy on nutrition and aging (iana) task force, *The Journal of Nutrition, Health & amp; Aging* **13** (2009), 881–889.

[4] J. Biswas, et al., Health and wellness monitoring through wearable and ambient sensors: Exemplars from home-based care of elderly with mild dementia, *Annals of Telecommunications* **65** (2010), 505–521.

[5] B. Brumitt, B. Meyers, J. Krumm, A. Kern and S. Shafer, Easyliving: Technologies for intelligent environments, in: *Proc. of the 2nd International Symposium on Handheld and Ubiquitous Computing*, HUC '00, pp. 97–119.

[6] S.K. Card, G.G. Robertson and J.D. Mackinlay, The information visualizer, an information workspace, in: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology*, CHI '91, pp. 181–186.

[7] N. Carrey, Establishing pedestrian walking speeds, Tech. rep., Portland State University – ITE Student Chapter, 2005.

[8] A.K. Dey, G.D. Abowd and D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, *Human-Computer Interaction* **16** (2001), 97–166.

[9] K. Gajos and D.S. Weld, Supple: Automatically generating user interfaces, in: *IUI '04: Proc. of the 9th International Conference on Intelligent User Interface*, New York, NY, USA, ACM Press, 2004, pp. 93–100.

[10] M. Ghorbel, M. Mokhtari and S. Renouard, A distributed approach for assistive service provision in pervasive environment in: *Proc. of the 4th International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, WMASH '06, 2006, pp. 91–100.

[11] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux and M. Mokhtari, A self-configuration middleware for smart spaces, *International Journal of Smart Home* **3**(1) (2009), 7–16.

[12] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux and M. Mokhtari, A software self-organizing middleware for smart spaces based on fuzzy logic, in: *12th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2010, pp. 138–145.

[13] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux, Tyche project: A context aware self-organization middleware for ubiquitous environment, in: *Proc. of the 2011 IEEE International Conference on High Performance Computing and Communications*, HPCC '11, 2011, pp. 924–929.

[14] R. Kadouche, B. Abdulrazak, M. Mokhtari, S. Giroux and H. Pigot, A semantic approach for accessible services delivery in a smart environment, *International Journal of Web and Grid Services* **5** (2009), 192–218.

[15] M.C. Kaptein, C. Nass and P. Markopoulos, Powerful and consistent analysis of likert-type ratingscales, in: *Proc. of the 28th International Conference on Human Factors in Computing Systems*, CHI '10, ACM, New York, NY, USA, ACM, 2010, pp. 2391–2394.

[16] G.J. Klir, G.J. and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[17] T. Muraoka and H. Ikeda, Selection of display devices used at man-machine interfaces based on human factors, *Industrial Electronics, IEEE Transactions* **51**(2) (2004), 501–506.

[18] J. Nielsen and J. Levy, Measuring usability: Preference vs. performance, *Commun. ACM* **37** (1994), 66–75.

[19] Z. Obrenovic and D. Starcevic, Modeling multimodal human-computer interaction, *ACM Computer Journal* **37** (2004), 65–72.

[20] D. Preuveneers, et al., Towards an extensible context ontology for ambient intelligence, in: *Ambient Intelligence*, P. Markopoulos, B. Eggen, E. Aarts, and J.L. Crowley, eds, Lecture Notes in Computer Science, Vol. 3295, Springer, Berlin/Heidelberg, 2004, pp. 148–159.

[21] Y. Rahal, P. Mabilleau and H. Pigot, Bayesian filtering and anonymous sensors for localization in a smart home, in: *AINA Workshops* (2) (2007), 793–797.

[22] A. Ranganathan, C. Shankar and R. Campbell, Application polymorphism for autonomic ubiquitous computing, *Multiagent Grid Syst.* **1**(2) (2005), 109–129.

[23] P.C. Roy, B. Bouchard, A. Bouzouane and S. Giroux, A hybrid plan recognition model for alzheimer's patients: Interleaved-erroneous dilemma, in: *Proc. of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Washington, DC, USA, IEEE Computer Society, IAT '07, 2007, pp. 131–137.

[24] S. Sakurai, et al., *A Middleware for Seamless Use of Multiple Displays*, Lecture Notes in Computer Science, Vol. 5136/2008, 2008, pp. 252–266.

[25] S. Shafer, J. Krumm, B. Brumitt, B. Meyers, M. Czerwinski and D. Robbins, The new easyliving project at microsoft research, in: *Proc. Joint DARPA/NIST Smart Spaces Workshop*, 1998, pp. 30–31.

[26] R. Sirdey, J. Carlier, H. Kerivin and D. Nace, On a resource-constrained scheduling problem with application to distributed systems reconfiguration, *European Journal of Operational Research* **183**(2) (2007), 546–563.

[27] A. Syed, J. Lukkien and R. Frunza, An architecture for self-organization in pervasive systems, in: *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1548–1553.

[28] V. Talwar, D. Milojicic, Q. Wu, C. Pu, W. Yan and G. Jung, Approaches for service deployment, *IEEE Internet Computing* **9**(2) (2005), 70–80.

[29] M. Vallée, F. Ramparany and L. Vercouter, Flexible composition of smart device services, in: *The 2005 International Conference on Pervasive Systems and Computing (PSC-05)*, Las Vegas, 2005, pp. 27–30.

[30] C. White, Health care spending growth: How different is the Uunited States from the rest of the OECD? *Health Affair* **26**(1) (2007), 154–161.