Intelligible software delivery in smart environments supported by a macro and micro context awareness model

Charles Gouin-Vallerand[1], Patrice Roy[3], Bessam Abdulrazak[3], Sylvain Giroux[3], and Anind Dey[2]

[1] Télé-Université du Québec, Montréal, Canada
[2] Carnegie Mellon University, Pittsburgh, United States
[3] Université de Sherbrooke, Sherbrooke, Canada

Abstract. Smart environments help to increase the autonomy and quality of life of people with special needs (PwSN) through adapted assistive services. In conjunction with context awareness and service delivery mechanisms, it is possible to dynamically deliver services to users by taking into account contextual information, e.g., user's locations, devices' states, current activities. However, implementing such systems in actual smart spaces is not trivial. The micro and macro context awareness model helps in defining layers on which contextual information are processed as close as possible from their sources (micro), without losing the benefits of information processing at a systemic level (macro). This paper describes the micro and macro context awareness model and its uses in the implementation of a service provision system for smart environments. Transparency and intelligibility are important factors in context awareness system, which help developers to fully understand the full behavior of the systems and help users to learn how the systems work. We therefore accompanied our micro and macro model with an intelligibility model which allow the generation of explanations describing the system's behaviors according to the processed context. Finally, results coming from the evaluations of the micro/macro model and comparison with related works are presented.

1 Introduction

The growing population of elders [33] with cognitive deficiencies and physical impairments, combined with the high costs and minimal supplies of caregiving resources, provide a compelling argument for a new vision of assistance at home [28]. Ubiquitous computing, context awareness and artificial intelligence techniques can transform human habitats into smart spaces able to provide assistance, contextual help and remediation by the environment. To assist these users in their daily living activities, dynamic and intelligent mechanisms are 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. Such service provision mechanisms rely mainly on the use

of contextual information from the environment and user profiles to accurately identify on which devices each service must be deployed.

Context awareness is needed in situations where software and hardware must collaborate in order to cope with complex data. A context aware system hosts software components that infer additional, synthetic context from the raw context provided by sensors and from other synthetic context. Context awareness enables such a system by assisting users in performing daily life activities or warns specialized personnel that human intervention is required. Software components 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[9][26]. 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 environment where Context-awareness could play a role such as assisting people with disabilities (e.g. hospitals, hotel rooms, apartments, houses, classrooms). Thus, Context-aware services can have several benefits for PwSN and a number of projects proposed in the last years present solutions that increase the quality of life of PwSN. For instance, Giroux et al. [12] propose a framework to support people with cognitive deficiencies, by monitoring the current states of users' activities through context awareness and assisting users step-by-step in their activities when errors or confusion are detected. Skubic et al. [29] use contextual information from smart home sensors to continuously monitor users' activities and assessing health changes, such as cognitive decline. Moreover, context awareness is often implemented in user mobility scenarios, by using mobile devices such as smart phones, embedded sensors and location acquisition system, e.g. GPS. For instance, Hoey et al [16] use contextual information from smart phones to recognize wandering behaviors with people with dementia. A large number of other projects and publications proposes solutions for PwSN based on context awareness and these three last examples give an overview of the possibility of context awareness for assisting and helping PwSN.

In this paper, we present our work around the implementation of a context and user aware service provision system for smart environments (Section 3.2), based on the micro and macro context awareness model (Section 3). The micro context awareness revolves around the subjective perception and the understanding an environment entity has of its environment. On the other hand, the macro context awareness is the global, emergent picture that components help build of entities in their environment [2]. This model helps in managing the complexity related to the quantity of contextual information and context sources, as well as providing headlines to build more versatile, dynamic and autonomous context aware systems. The micro/macro model drove the development of our service provision system, and helps with the organization and use of contextual information.

On the other hand, to reach its goal, a context aware system based on a micro and macro model should tend toward transparence, and propose to users

and software developers mechanisms to understand which contextual information are used by the system and how these information have an impact on the system's behavior. Intelligibility [6] focuses on telling users what a system did and why it did it. Thus, intelligibility aims to reduce the gap between the user's understanding of a system and the intrinsic reasoning mechanisms of this system. Therefore, we also present in this paper a structure to support the intelligibility in a micro and macro context awareness and its implementation in our service provision system (Section 3.3).

Finally, this paper presents also an evaluation (Section 4) of the micro and macro context awareness model, through the service provision system implementation, and a comparison with two related works, Trumler et al. [30] and Ranganathan et al. [27].

2 Related works

Context-awareness is used in situations where software and hardware have to cope with complex data. It often involves Semantic Web technology [10] or devices that use external data to build a model (i.e. an awareness) of their surroundings, relationships with other devices [4] or subject of interest [22]. Context-awareness also applies to software that assists users and takes into account the relationships between users and their environment.

Context can refer to the situation of a device or of a human being. In [23], Context is defined as a setting in which an event occurs, which ties Context to a location. For Abowd, Dey et al. [3], context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. In [11], a distinction is made between situation-awareness, related to the users location, and Context-awareness, related to conditions (temperature, weather, lighting,. . .) that prevail in that location.

Over the last two decades, several works have been performed on context awareness, mostly about its utilization and modeling. Dey et al. [8] are among the first to have specified and developed a solution that implements a framework and a context model for ubiquitous computing. The Context Toolkit [9] provides the functionalities to fill the gap between the hardware which collect the contextual information, e.g. the sensors, and the ubiquitous applications that are using them.

In the CoWSAMI project, Athanasopoulos et al. [5] propose a context aware framework based on web services as interfaces to context sources and dynamically updateable relational views for storing, aggregating and interpreting context. Context rules are employed to provide mappings that specify how to populate context relations, with respect to the different context sources that become dynamically available.

In the SOCAM architecture, Gu et al. [15] propose a framework to support the rapid prototyping of context aware services. This framework is based on

web ontologies to represent contextual information and use several protocols to discover context providers, bring information to the context interpreters and finally to the context aware services.

Several other works exists, proposing different approaches and technologies to support context awareness, e.g. the ESCAPE project [31] and Preuveneers [25].

On the other hand, some works in the area of service provision have developed context aware mechanisms to support service delivery in ubiquitous environments. Trumler et al. [30] are proposing a solution based on the social behavior of a cooperative group in the context of job assignments. This job assignments is based on the context of each participating devices, reasoning on the capacity of nodes is made locally on each device and then forwarded to a central node responsible for the repartition of the services among the devices. In the Gaia project, Ranganathan and Campbell[27] 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 software needs; software that must be deployed in a given mediated space. Ranganathan's work on the software provision is mainly focusing on the hardware analysis and the environment spatial description, evacuating the user needs, capabilities and preferences.

As these works are addressing the issues of defining models to represent contextual information and proposing mechanisms to distribute information among context consumers, they are not addressing particular questions about today's smart environments. How these systems can resolve the complexity of today's smart spaces with hundreds or thousands of context sources ? How the user profiles can be integrated to the context awareness and how the privacy of the profiles can be ensured ? How the scalability of the context awareness can be supported for open environments such as smart cities, with all the uncertainty related to these kind of environment ?

Some of these works propose answers by using a standardized definition of the contextual information through a Semantic Web language or addressing the complexity and heterogeneity of the smart environment through web service technology. These works provided a basis for the conception of our context awareness model and the implementation of our service provision system.

3 Context awareness model and strategies

Different strategies exist to compute on the contextual information of a given system or environment. Such strategies are required to manage the complexity related to the quantity of data and information sources in complex smart spaces. For instance, it is possible to divide the context awareness of a given system in several sub-contexts called micro contexts [7], related to specific devices or closed locations. At a second level, it is possible to aggregate and combine the micro contexts with other sources of information to build a more global representation

of the context in a given environment, the macro context or macro context awareness.

The macro context awareness is presented as something directly tied to a systemic model of the user and the conditions around this individual; the human is the center of attention. The system's main goal is to try to keep an up-to-date representation of a human and its environment. The human, given its central role in the world, becomes not only the main mutator of context, but is also the focus of the activity, making macro context awareness particularly well suited to applications that assist a given user in a smart environment.

On the other hand, the micro context awareness can be defined as the context awareness for devices that can be split up into three components: activity, environment and self. The activity describes the task the user/component is performing at the moment or more generally what his or her behavior is. The environment describes the status of the physical and social surroundings of the user/component. Finally, the self describes the intrinsic information about the user/component, e.g., preferences, capabilities, etc. Micro context awareness in smart environments relies on distributed computing to share and publish information between micro context components. Moreover, micro context components have to deal with ad hoc communications directly related to the ad hoc topologies of some environments. The hallmark constituents of controlled spaces, for example key nodes, can be used in micro context but cannot be depended upon by individual components. Micro context awareness is not only awareness of a single component, sometimes named raw context, but rather a model of context awareness that focuses on information available to the acting components and that maintains no dependence on system-wide knowledge or tools.

In comparison to the works presented in the previous section, we believe that a context aware system should use both perspectives to build software that better adapt to different situations and are less complex to deploy and manage. We will now present how we build the service provision system using both layers.

3.1 Architecture overview

We designed a distributed multi-level architecture ($L_0..L_n$) that integrates the micro and macro Context awareness models by providing structures, Context descriptions, ontologies and services to embedded middleware and software.

Figure 1 presents this multi-level architecture from a macro perspective. From this perspective, the architecture is instrumented with customized nodes, fixed or mobile. Each L_i node offers L_i-level services. L_1 nodes are hardware abstraction gateways that enable the system to interact with L_0 real world data (sensors and actuators). L_i nodes are concentrators that aggregate the services offered by L_j,$j < i$ nodes in a zone (location of influence). These concentrator nodes aggregate lower-level services in a given zone. Zones group nodes according to some criteria (for example, physical proximity), and can overlap. The hierarchy can be based on Agent needs: location (e.g. L_1 is for devices, L_2 for a room, L_3 for a floor), processing requirements (e.g. a L_i,$i > 1$ node concentrates instances of L_{i-1} nodes, to perform load balancing), or to address security, confidentiality

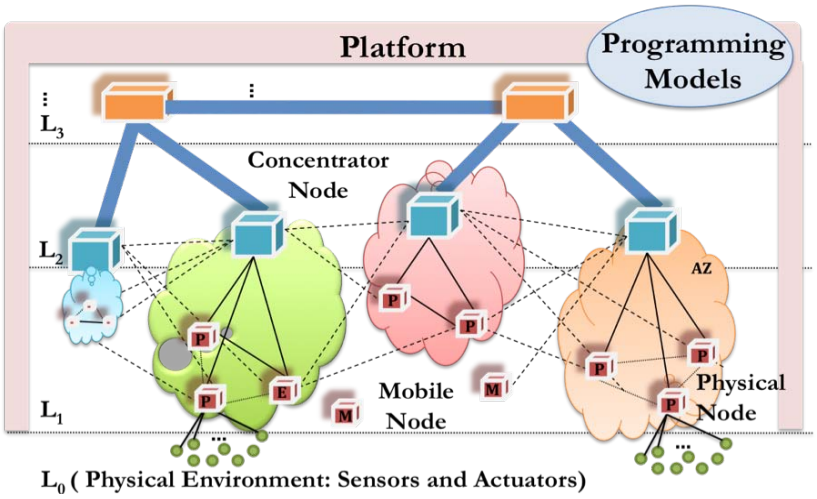

Fig. 1. Overview of the macro and micro context awareness

and ethical concerns. Mobile nodes can, at any time, gain or lose contact with individual nodes due to faults, distance, noise or other factors.

In hierarchical scenarios, L_0 devices gather information from sensors to build Context. This Context is retrieved by L_1 devices and is used to create the first Context-aware functionalities related to a component and its neighborhood, i.e. raw Context. L_2 devices aggregate micro Context information from L_1 devices, building a larger vision of Context for a given zone, leading to Context-awareness from a macro perspective. In layers L_3 and up, devices aggregate Context from lower layers and construct a global vision of the known environment. Mobile micro Context-aware nodes, on the other hand, move freely from zone to zone. Both their network neighborhood and their conceptual level change with time and location. The application perspective of macro Context-aware systems, where Agents have well-known roles in a well defined system, is replaced by localized, per-node roles, resulting in a local rather than systemic perspective. The proposed service provision framework, presented in the next section, focuses on the layers L_0 to L_2. A last aggregation of the contextual information is made in smart environment management tools, deployed on environment managers and professionnal caregivers mobile devices, and represent a last(L_3) macro context layer.

3.2 The 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. We implemented a middleware for smart homes/apartments dedicated to the self-management of the software deployed in these environments [13]. Rallying several technologies e.g. Web Services, OSGi, OWL ontologies and fuzzy logic, this middleware uses the contextual information of the environments to find the devices that are the most adapted to the service needs, user capabilities and preferences, device resources and environment topology. 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 error and failure recognition systems to use. These systems use the service provision functionalities by supplying information related to the service to deliver, e.g. Which users are targeted?, Which assistive services?, What are the service needs?, Is there a specific zone of the environment that is targeted by the assistance request?

Service provision scenario Before explaining in details the service provision system, a brief example illustrating a service provision would help to assimilate the kind of contextual information used by the system and what is type of services that are delivered to the users.

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 deliver 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 into several zones, e.g. the kitchen area, the living room, 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 3 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 shown (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.

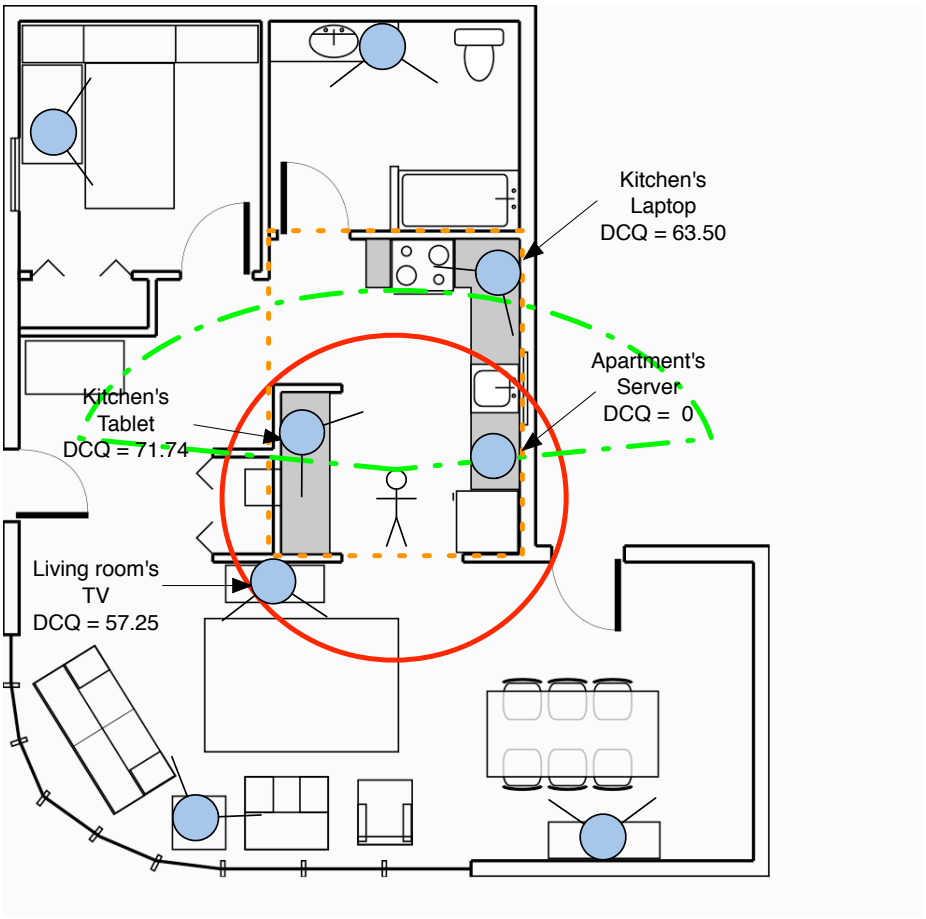

Fig. 2. An example of a service provision with some of the processed contextual information : user's mobility (circle), user's visual acuity and field of vision (arc) and targeted zone (rectangle)

Each device in the environment is collecting the contextual information related to its sensors and internal metrics. These information are delivered to the service provision reasoning engines of each device, at the micro-level (FLORE-D), which analyzed and, if required, abstracted and deliver the information to the reasoning engine at the macro-level (FLORE-C). Both kind of reasoning engines compute scores on how each device have the right resources and context to support the meal preparation assistant. In this example (based on a real deployment scenario), the device with the highest score 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. For instance, 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 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.

In this example, each device have the tools to collect and represent its micro-context and analyze it with a reasoning engine. Depending on the kind of collected information, e.g. an approximation of the user position found by an infrared sensor, the information is abstracted and sent to the macro-level. This micro-level information can be used by the macro-level reasoning engine and matched with other information provided by a smart environment ontology (e.g. user profile and environment topology). By using the macro and micro context awareness model, each layers (devices and environment nodes) are able to compute deployment scores independly, with corresponding inprecision factors, and have a certain autonomy in this computation. However, it is by exchanging information between layers and taking in account macro and micro-level deployment scores (see next section) that the system is increasing its precision and effectiveness.

System overview The service provision mechanisms use four main context elements: user profiles, environment device profiles, topology of the environment and software profiles. Each service that needs to be deployed in the environment has hardware, software or contextual needs. On one hand, assistive applications like a meal preparer 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 (preferences toward specific interaction peripherals such as keyboard, touch screen, mouse, etc.) about the environment devices; the devices also have profiles with capabilities e.g. their resources, connected peripheral devices, etc. All these components are present in the smart environments at different (or not) locations and are related to contextual zones e.g. the kitchen, the bathroom, the living room, etc. Finally, the user profiles describe a user's physical capabilities, e.g. their vi-

sual acuity, walking speed or their hand workspace, their interaction preferences and their location in the environment.

In our work, we used the micro and macro context model to divide into layers the reasoning on the contextual information related to the service delivery. We therefore divided the system into two kinds of components: Device nodes and Coordinator nodes. To create these nodes and link them to a macro or micro layer, we used our model definitions as decision rules. If a node primarily require instrinsic information to fulfill its job, then it should be a micro context provider. On the other hand, if a node uses intensively data related to the smart environment structure, or data mined from the Web or ontologies, then it should be related to the macro-level context.

Thus, device nodes collect the information related to their hardware, their location in the environment and their connected interaction peripherals. This information is processed, along with an abstracted representation of the service to deploy (provided by the coordinator node), within a Fuzzy Logic controller. Fuzzy logic [17], based on the fuzzy set theory, allows reasoning at a high level using linguistic terms instead of numerical values, with conditional rules (e.g., IF x IS a THEN y IS b). 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 3 presents a simplified example, with only one fuzzy set, of how fuzzy logic is used, where the system uses information about devices' resource consumption to identify the best deployment target.

To determine which device to deploy a service to, contextual information is processed using two implementations of a JFuzzyLogic controller[4]. The contextual information associated with the device's intrinsic data and service component profile are processed in a fuzzy controller in each environment device, producing Device Scores. Another fuzzy logic controller, located in a coordinating device, processes the data related to the user profile and the environment topology, producing User Scores. Contextual information are fuzzified using membership functions and fuzzy rules (15 rules for Device Scores and 41 rules for User Scores). The inference uses the min-max method for the aggregation, minimum function for the activation and the maximum method for the accumulation. Device scores and User scores are merged together to produce a final Deployment score, by computing a weighted mean, with a weight related to the importance of the User profile.

$$Score = \frac{(User.Score * W) + (Device.Score * (1 - Weight))}{2} \quad (1)$$

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

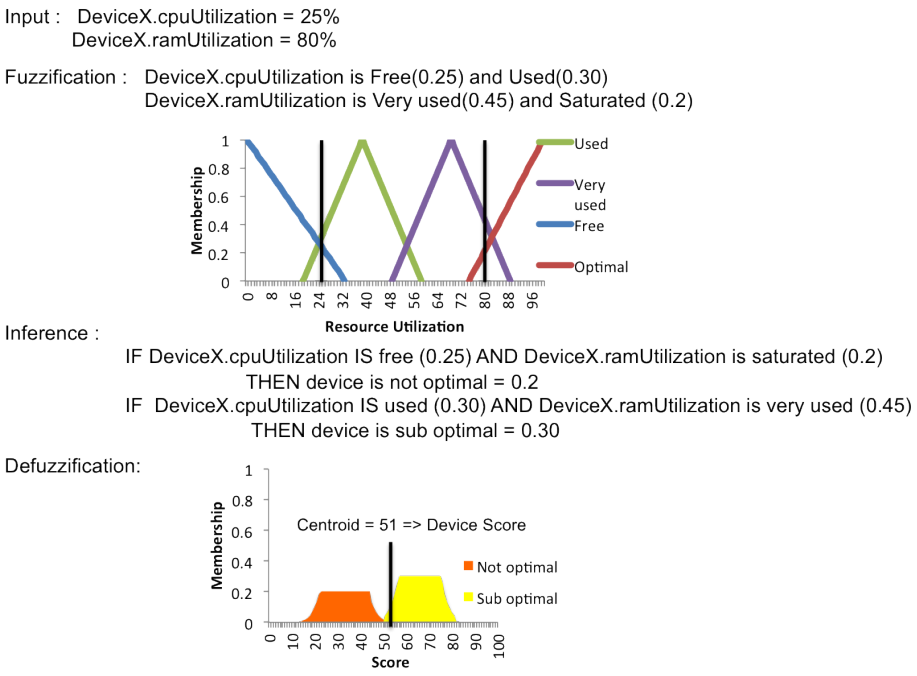

Fig. 3. Fuzzy logic example with device's resources consumption

Therefore, this controller determines (based on fuzzy sets, fuzzy rules and defuzzification set) if each device has the resources to support the services to deliver and at which performance/quality the device can support the service in question (with a performance score between 0 and 100). Such information is shared with the other components of the environment along with an abstracted representation of the device. The Device nodes have subjective views of their surrounding and a partial view of the environment through the data forwarded by the coordinator nodes or the other device nodes.

The coordinator nodes have the responsibility to receive service provision requests and manage the reasoning process about which devices in the environment are the most appropriate to support the services. Depending on the service needs, the coordinator nodes take into consideration the profile of the users to determine which device is best adapted to user capabilities and interaction preferences, and the topology of the environment, i.e. in which environment zones users and components (devices and peripherals) are in. This information is computed with the abstracted representation of the micro context and the performance score computed by the device node. To do so, the coordinator node uses an OWL ontology and a second Fuzzy Logic controller to make the final decision about where each service must be deployed. The contextual information processed by the Coordinator node is directly related to the macro context

awareness level. The users are the central focus point of the reasoning process in the Coordinator node and these nodes have a systemic and consistent view of the environment context, partially provided by the abstracted micro contexts. Figure 4 presents the use and exchange of the contextual information between the Coordinator node and a Device node, respectively related to the macro and micro context levels.

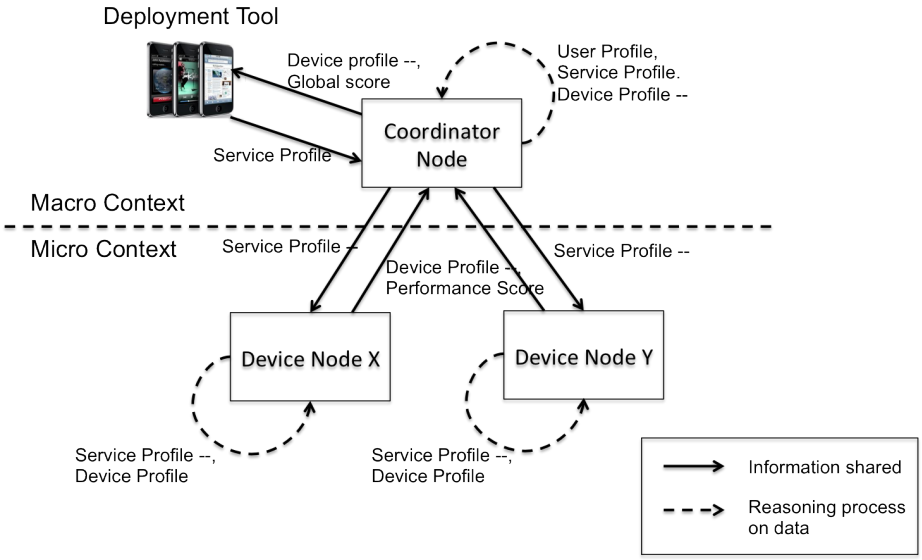

Fig. 4. Utilisation of the contextual information in the service provision system

One direct benefit of dividing the service provision system in two context awareness layers is the reduction, on each layer, of the quantity of data managed. Moreover, dividing the contextual information in different layers reduces the strong coupling that can exist between information, reducing the complexity of updating information across layers. For instance, in the service provision system, the Device nodes know their location in the environment, but don't know directly in which logical zone they are, as this depends on a more global view of the environment.

Another benefit of the micro and macro context awareness layers is to prevent the divulgence of sensitive information between software components. In the case of the service provision system, user profiles are kept in the Coordinator node, avoiding publication of private information such as user interaction capabilities and preferences. Another benefit is the non-divulgence of the service profile, with each Device node receiving an abstracted representation of the services to deploy until they are finally selected for the deliveries, keeping private the real nature of the services and the provided assistance. Finally, distributing the processing

of the contextual information among layers and nodes has an important impact on the performance of the system. Using macro and micro context layers allows leveraging of the processing capacities of the different devices in the environment and reduces the computing time of the macro context layer (less information to process). Some results on the distribution of the reasoning processes for the service provision system are presented in Section 4.

3.3 Context intelligibility

Some context aware application support some level of intelligibility. Vermeulen et al. [32] and Kuleza et al. [18] expose to the users the contextual information (e.g. sensors values) that is used in the reasoning process of their system, along with some explanations. They organized there explanation around an interrogative form: *Why?* (i.e., why the system took a particular action), *Why not?* (i.e., why the system did not take another action instead), *What if?* (i.e., what are the results if the context is that), *How to?* (i.e., how can the system be made to produce these results). Lim and Dey [20] propose more intelligibility functionnalies with the Intelligibility Toolkit [19] by introducing new explanation models, also based on an interrogative form, e.g. *How to ?, What if ?, Certainty*, etc. They applied there intelligibility toolkit to different reasoning models such as bayesian network and rules base model.

In [14], we proposed and evaluated an intelligibility mechanisms for fuzzy logic, loosely based on the Intelligibility Toolkit of Lim and Dey. These mechanisms used the *Why, Why not and What if* explanations as structure to explain system's behaviors to the users. Moreover, we used the software provision framework, described above, as the testbed to implement the intelligibility in a micro and macro context awareness model.

One of the biggest challenge in implementing this intelligibility was to create the right mechanisms to encapsulate the meta-data about each micro and macro contexts, which are used to generate the explanations. For instance, in the original Intelligibility toolkit, as the framework is based on a macro model, there is no real problem with acceding to any contextual information and reasoning engine processes, as must of these data are locales. In the case of a macro and micro model, explanation generators need an access to meta-data about the contextual information and the system behavior across several macro and micro layers.

Strategy and Command design patterns gave the right structures to implement such encapsulation. We designed a strategy object to encapsulate the creation of explanations around the Fuzzy Logic which can generate *Why, Why not and What if* explanations from the reasoning engine embedded in the coordinator node and the device nodes. As the way to get meta-data can differ for each kind of micro and macro context, we encapsulated the way to retrieve the meta-data for each explanation generation methods, in Command design pattern objects. For instance, in the software provision system, to get real time access to meta-data in micro context, the Command objects must send Web services calls to the device nodes, and must send SparQL queries to the OWL ontology for

the macro contextual information. Figure 5 presents an overview of the structure used to create the intelligibility mechanism. During explanation generation, as information are provided by several sources, it is required to merge and aggregate information. We create strategies [14] to reduce, aggregate and simplify the quantity of information presented to the users through the explanations, for instance by using the Weight of Evidence method [24]. This intelligibility model is dynamically scalable, as the number of micro-context providers are automatically integrated into the model through the discovery mechanisms of the service provision system.

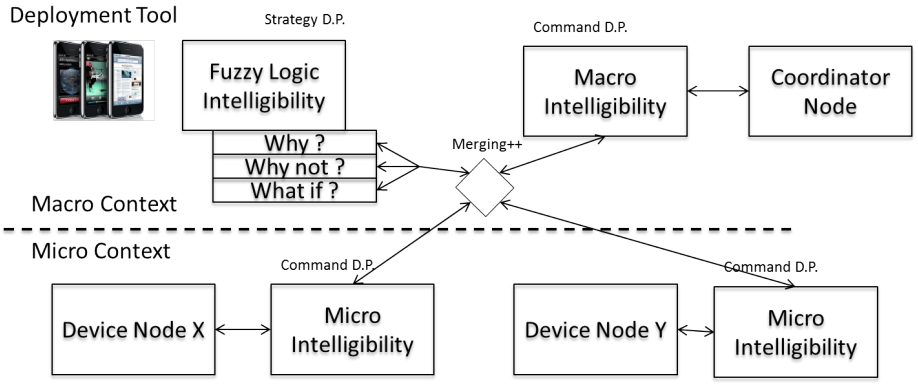

Fig. 5. Overview of the intelligibility feature in the service provision system

To make the software provision system more intelligible to users, we proposed to organize the explanations around a layout based on the three Fuzzy Logic reasoning steps (fuzzification, inference, defuzzification), as Fuzzy Logic is the principal reasoning method used by the provision system. Such a strategy introduces users to the general concepts of fuzzy logic and builds a logical organization modeled on the real implementation of the system. Secondly, each reasoning step must be made more intelligible by reducing the amount of presented information and specific knowledge implications. For instance, it is not required to show a complete definition of the fuzzification functions to users (which requires advanced math skills to understand); instead, a summary introducing the membership values and linguistic terms for each input is more appropriate. Thus, to make fuzzy logic systems more intelligible, we need to support : (i) fuzzification: presents the system's inputs and a summary of the related linguistic terms/membership value; (ii) inference: shows the users a limited list of fuzzy rules and their conclusions/results. Introducing the users to the sub-steps of inference (aggregation, accumulation, activation) should be done with caution as it increases the complexity for users; (iii) defuzzification: summarizes the inference's conclusions, and introduces the system's outputs and the way they are computed (i.e., centroid, center of area, right or left max value).

Adding the intelligibility structure to the service provision system didn't add a significant workload and processing time to the system itself (Section 4). Must of the workload related to the intelligibility is made by the GUI representation of the explanations (Figure 6). Moreover, we evaluated the intelligibility support in our system through a user study. We selected ten participants with two kinds of backgrounds: with computer science and I.T. professionals and experts (5 participants) and with people from the general population (5 participants). Each participants had to use the intelligibility features of the GUI and proceed to the deployment of software in a virtual smart apartment. The results of the study demonstrate that intelligibility and explanations help both I.T./C.S. professionals and less technical people to understand fuzzy logic. Intelligibility even helped the less technical participants to gain an understanding that was close to the technical experts. Moreover, the study highlighted the importance for users to compare data (why not?) and experiment with system modification (what if?). However, the quantity of information in the explanations can rapidly overwhelm users, particularly users without a background in I.T., situation that we have experimented during the study. More work will be required to increase the effectiveness of the design of explanations and the GUI, but the preliminar results shows that our intelligibility strategies provide useful information to users on the behavior of context-aware system.

Fig. 6. Screenshot of the management tool providing intelligibility functionalities

3.4 Implementation

The service provision system is build 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 (Figure 7). The Environment Management Coordinator node have the job to manage the device discovery, maintain the environment ontology, receive the service provision requests and manage them with the help of the FLORE. The Device nodes are deployed on the environment devices and host the service module to provide to the users. They also perform some reasoning on the context such as on the hardware resources and the interaction peripheral availabilities.

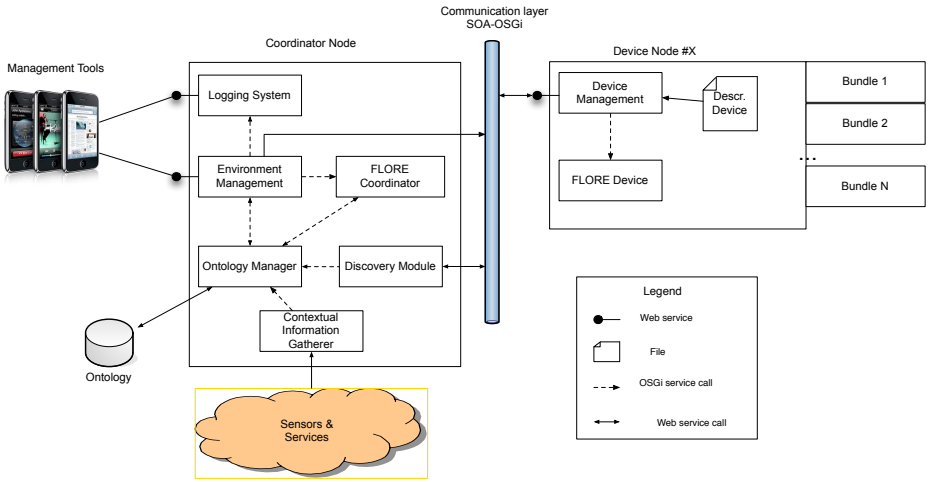

Fig. 7. The service provision system architecture with its two main components: the Environment Management Coordinator and the Device Node

As the provision functionalities use contextual information and the user profiles to find the optimal way to provide the services to the environment users, a way to describe and contain these description 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 describe 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 use 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 and 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 context, by introducing new rules and functions.

4 Evaluation and results

Throughout the technical evaluation of the service provision system, we evaluated the macro and micro context awareness approach applied to service delivery in a smart apartment. The evaluation was conducted in a real smart apartment (apartment in the infrastructure of the DOMUS Laboratory at the University of Sherbrooke), composed of 8 device nodes, 1 coordinator node, 25 and more interaction peripherals and over 50 context sources e.g., sensors, localisation system. We use different kinds of user archetypes in five assistance scenarios based on the validation used by Gouin-Vallerand et al. [13].

Firstly, we compared our approach with the similar work of Trumler et al. [30], which can be categorized as a micro context aware system. As Trumler et al. did, we evaluate our system by counting the number of messages exchanged for a given service delivery job. The number of messages exchanged in a distributed system for a specific job, can be a good way to compare the intrinsic complexity of systems [21], when the types of data and the amount of information are fixed. In the case of the macro and micro context aware service provision system, the number of messages exchanged between the nodes correspond to $Messages = (a \times d) + d + 1$ where d is the number of Device node and a is the number of services to deliver. Therefore, for five services to deploy in an environment with ten devices, the number of messages is 61 messages. For the same configuration, the solution of Trumler et al. in a similar configuration used 35 messages. However, if the nature of the information is the same (XML stream) in both works, the amount of information used by each system is different. Compared to Trumler et al.'s work, our service provision solution uses a more complete description of the services to deliver and involve the topology and the user profile in the reasoning process. Moreover, the Trumler et al.'s work groups several informations together in the same message, which reduce the amount of messages but not necessarily the complexity behind their reasoning process.

On the other hand, Ranganathan et al. [27] proposes a centralized solution involving the processing of the information based on a macro context awareness perspective. They measured the performance of their work based on computing, where their system took an average processing time of 0.93 second on a Pentium M 1.7 GHz system, for a scenario with two services deployed among five devices.

In comparison, our micro and macro solution takes 0.55 second for a similar scenario with eight devices, plus the computing related to the user profiles and where the system was composed of devices ranging from Intel Core 2 Duo 2.4 GHz to Tablets processors. This computing time included the Web service calls related to the 23 messages/requests exchanged during the service provision.

Of course, comparing computing times from systems with different hardware and setups is difficult. To push the comparison further, we measured the average computing times related for the system's software components and the communications (Table 1). By using the Passmark benchmark scores of the processors, 434 for the Pentium M and 1508 for the Core 2 Duo [5], which give a ratio of 3.47 in favor of the Core 2 Duo. Using this ratio with the average computing times of the Coordinator and Device node ($0.12+0.06$ seconds), the approximation (with an unknown error margin) of the computing time of the proposed solution on a Pentium M would be of 0.62 seconds (0.18×3.47). One conclusion from this exercise is that the web service calls represent a bottleneck for the performance of our system. In the case of the service provision, the distributed nature of the system cannot be avoided and is an intrinsic part of the solution. However, by adding to approximated computing time the web service calls (assuming that there is only small variations between the two hardware), the performance of the proposed solution is around the Ranganathan et al.'s result, with more data processed and the benefit of the micro context awareness in the favor of our work.

Finally, these results show that the proposed service provision system implemented using a micro/macro context awareness approach gave results that are at least equivalent to the other related works. As the complexity and the amount of data processed by our solution are higher, we had to use state of the art technologies (e.g. web services, OSGi framework, OWL ontologies, Fuzzy Logic controllers) to help us in implementing our approach, which would have been more difficult five years ago without them.

Table 1. Processing times of the service provision system's components

Component	Average computing time (sec)
Coordinator Node	0.12
All Device Nodes	0.06
One Device Node	0.01
All Web service calls	0.28
One Web service call	0.011

[5] Passmark benchmark : www.cpubenchmark.net

5 Conclusion

Today's smart environments are synonymous to context awareness, component dynamism and adapted services to assist the users in their daily living activities. If context awareness brings tools to increase the system adaptation to the surrounding environment, its implementation is not trivial. The micro and macro context awareness approaches help categorize to which kind of context each system component should be related, depending on the role of the components and the type of processed information. In this paper, we present our work around the implementation of a service provision system for smart spaces, based on a model where the processing of the contextual information is divided between a macro and a micro context awareness layer. This approach allows us to manage the complexity related to the contextual information, by processing data as close as possible to their context sources, with the Device Nodes, without losing the benefits of information processing at a systemic level, with the Coordinator nodes. Moreover, we demonstrate that our macro and micro layers approach gave more autonomy, scalability and adaptability than single layer approaches (e.g. Ranganathan's and Trumler's models) without a cost in the system performance.

We are currently working on the formalization of a model of the macro and micro context awareness approach and we are planning to implement it in large-scale systems for controlled smart spaces (e.g. smart homes) and open smart spaces (e.g. smart cities). The formalization of the model will help in implementing the functionalities to support both context layers, in environments where the quality of service can vary greatly.

With smart environments and smart cities being more popular to support dependant people or for mediated environments, our micro and macro context-awareness model will bring tips and clues for developers wishing to build polyvalent, autonomous and scalable context-aware system. Moreover, with the popularity of the cloud computing, where macro-level context-aware bundles can be deployed, and micro-level counterpart in mobile devices, the model proposed in this paper all make sense. This is also the direction that our future work will take.

References

1. Abdulrazak, B., Chikhaoui, B., Gouin-Vallerand, C., Fraikin, B.: A standard ontology for smart spaces. International Journal of Web and Grid Services 6(3), 244–268 (2010)
2. Abdulrazak, B., Roy, P., Gouin-Vallerand, C., Giroux, S., Belala, Y.: Macro and micro context-awareness for autonomic pervasive computing. International Journal of Business Data Communications and Networking (IJBDCN) 7(2) (2011)
3. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. pp. 304–307. HUC '99, Springer-Verlag, London, UK, UK (1999), http://dl.acm.org/citation.cfm?id=647985.743843

4. Alonso, E., Kristofferson, P., McCann, J.: Building Ambient Intelligence into a Ubiquitous Computing Management System. In: International Symposium of Santa Caterina on Challenges in the Internet and Interdisciplinary Research. SSCCII-2004, Amalfi, Italy (January 2004), http://dev.pubs.doc.ic.ac.uk/ambient-intelligence/
5. Athanasopoulos, D., Zarras, A., Issarny, V., Pitoura, E., Vassiliadis, P.: Cowsami: Interface-aware context gathering in ambient intelligence environments. Pervasive and Mobile Computing 4(3), 360–389 (2008), http://linkinghub.elsevier.com/retrieve/pii/S1574119207000740
6. Bellotti, V., Edwards, K.: Intelligibility and accountability: human considerations in context-aware systems. Hum.-Comput. Interact. 16(2), 193–212 (Dec 2001)
7. Biswas, J., 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, 505–521 (2010)
8. Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. Ph.D. thesis, Georgia Institute of Technology (2000)
9. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. 16, 97–166 (December 2001)
10. Feki, M.A.: A logic based approach for context reasoning in assistive environment. In: Proceedings of the 2nd International Convention on Rehabilitation Engineering & Assistive Technology. pp. 277–280. iCREATe '08, Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, Kaki Bukit TechPark II,, Singapore (2008), http://dl.acm.org/citation.cfm?id=1983222.1983296
11. Gessler, S., Martin, M., Weiss, S.: Context awareness in future life scenarios: Impact on service provisioning platforms. IEEE/IPSJ International Symposium on Applications and the Internet Workshops pp. 144–147 (2005)
12. Giroux, S., Bauchet, J., Pigot, H., Lussier-Desrochers, D., Lachappelle, Y.: Pervasive behavior tracking for cognitive assistance. International Conference on Pervasive Technologies Related to Assistive Environments, PETRA 2008 pp. 86:1–86:7 (July 2008)
13. Gouin-Vallerand, C., Giroux, S., Abdulrazak, B.: Tyche project: A context aware self-organization middleware for ubiquitous environment. In: 13th International Conference on High Performance Computing and Communications (HPCC) (sept 2011)
14. Gouin-Vallerand, C., Lim, B.Y., Dey, A.K.: Software provision in smart environment based on fuzzy logic intelligibility. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. pp. 774–777. UbiComp '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2370216.2370389
15. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. J. Netw. Comput. Appl. 28, 1–18 (January 2005), http://dl.acm.org/citation.cfm?id=1053030.1053031
16. Hoey, J., Yang, X., Quintana, E., Favela, J.: Lacasa: Location and context-aware safety assistant. In: Proceeding of International Conference on Pervasive Computing Technologies for Healthcare. San Diego (May 2012)
17. Klir, G.J., Yuan, B.: Fuzzy sets and fuzzy logic: theory and applications. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1995)
18. Kulesza, T., Wong, W.K., Stumpf, S., Perona, S., White, R., Burnett, M.M., Oberst, I., Ko, A.J.: Fixing the program my computer learned: barriers for end users, challenges for the machine. In: Proceedings of the 14th international conference on Intelligent user interfaces. pp. 187–196. IUI '09 (2009)

19. Lim, B.Y., Dey, A.K.: Toolkit to support intelligibility in context-aware applications. In: Proceedings of the 12th ACM international conference on Ubiquitous computing. pp. 13–22. Ubicomp '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1864349.1864353
20. Lim, B.Y., Dey, A.K.: Investigating intelligibility for uncertain context-aware applications. In: Proceedings of the 13th international conference on Ubiquitous computing. pp. 415–424. UbiComp '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2030112.2030168
21. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1996)
22. Miaou, S.G., Shih, F.C., Huang, C.Y.: A smart vision-based human fall detection system for telehealth applications. In: The Third IASTED International Conference on Telehealth. pp. 7–12. Telehealth '07, ACTA Press, Anaheim, CA, USA (2007), http://dl.acm.org/citation.cfm?id=1672136.1672139
23. Neovius, M., Sere, K., Yan, L., Satpathy, M.: A formal model of context-awareness and context-dependency. IEEE International Conference on Software Engineering and Formal Methods pp. 177–185 (2006)
24. Poulin, B., et al.: Visual explanation of evidence in additive classifiers. In: Proceedings of the 18th conference on Innovative applications of artificial intelligence - Volume 2. pp. 1822–1829. IAAI'06 (2006)
25. Preuveneers, D.: Context-aware adaptation for Ambient Intelligence. LAP Lambert Academic Publishing (2010), https://lirias.kuleuven.be/handle/123456789/267553
26. Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere, K.: Towards an extensible context ontology for ambient intelligence. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) Ambient Intelligence, Lecture Notes in Computer Science, vol. 3295, pp. 148–159. Springer Berlin / Heidelberg (2004)
27. Ranganathan, A., Shankar, C., Campbell, R.: Application polymorphism for autonomic ubiquitous computing. Multiagent Grid Syst. 1(2), 109–129 (2005)
28. Rialle, V., Ollivet, C., Guigui, C., Herv, C.: What do family caregivers of alzheimer's disease patients desire in smart home technologies? Methods of Information in Medicine 47, 63–69 (2009)
29. Skubic, M., Guevara, R.D., Rantz, M.: Testing classifiers for embedded health assessment. In: Donnelly, M.P., Paggetti, C., Nugent, C.D., Mokhtari, M. (eds.) ICOST. Lecture Notes in Computer Science, vol. 77251, pp. 198–205. Springer (2012)
30. Trumler, W., Klaus, R., Ungerer, T.: Self-configuration via cooperative social behavior. In: Third International Conference on Autonomic and Trusted Computing 2006, ATC. Lecture Notes in Computer Science, vol. 4158, pp. 90–99. Springer (2006), http://dblp.uni-trier.de/db/conf/atc/atc2006.htmlTrumlerKU06
31. Truong, H., et al.: Escape - an adaptive framework for managing and providing context information in emergency situations. In: in EuroSSC,ser. Lecture Notes in Computer Science. pp. 207–222. Springer (2007)
32. Vermeulen, J., Vanderhulst, G., Luyten, K., Coninx, K.: Pervasivecrystal: Asking and answering why and why not questions about pervasive computing applications. In: Intelligent Environments (IE), 2010 Sixth International Conference on. pp. 271 –276 (july 2010)
33. White, C.: Health Care Spending Growth: How Different Is The United States From The Rest Of The OECD? Health Affair 26(1), 154–161 (jan 2007)