Context-Aware Service Provision in Ambient Intelligence: a case study with the Tyche project

Charles Gouin-Vallerand, Télé-Université du Québec, 5800 St-Denis Boul. Office 1105, Montreal, Quebec, Canada, H2S 3L5, charles.gouin-vallerand@teluq.ca

Introduction

Smart environments are, since twenty years, an important topic of research in computer science. The major reason is that smart environments offer solutions to several problems that our society face. For instance, with the rising proportion of the elder’s population in most of the Occidental and Asian countries and the scarcity of caregiving resources compel for a new vision of caregiving. Ubiquitous technologies, such as smart homes, give the technological support to ensure personal cares at home (Rialle et al., 2008). On the other hand, the smart cities initiative corresponds to the approach, where the urban spaces are instrumentalized with sensors, citizen are providing data through crowdsensing and data collected are analyzed to increase the effectiveness of the management and to propose enhanced services to the citizen (Ratti and Townsend, 2011). Both kinds of smart environments have their own requirements and purpose; however it is essential to support users across these environments, notably for people with special needs.

In a way to assist the users in their daily living activities, context-aware and intelligent systems are required to provide assistive services to users on their devices: smart phones, tablets, desktop computers or embedded devices; depending of the available devices and environment type. By context-awareness, we mean the ability of a system to capture, model and use specific information about the environment surrounding the system, such as location, time, user profile (Ryan et al. 1997). For instance, a context aware system can host software components that infer synthetic context from the raw context provided by sensors and from other synthetic context (e.g.
other devices). Context awareness enables such a system by assisting users in performing daily life activities or warns specialized professionals 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 ambient intelligence (Dey et al., 2001)(Preuveneers et al., 2004). 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 people with special needs (PwSN) and a number of projects proposed in the last years propose solutions that increase the quality of life of PwSN. For instance, (Giroux et al., 2008) proposes 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., 2012) 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, 2012) uses contextual information from smart phones to recognize wandering behaviors with people with dementia. A large number of other projects and publications propose 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.
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 (Roy et al. 2007). 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 (SOA) the service provision functionalities of the system can be used by several systems in a smart environment. Thus, the complexity of the provision reasoning processes 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.

To do so, a directive or a recommended based service provision approaches are available, depending of several factors: context, type of services to deploy, user profiles, type of devices, etc. However, the complexity of the smart environments 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. Thus, building and deploying context-aware service provision system in smart environments such as smart homes or smart urban environments is a complex process, for instance its implementation and the management of the implicit complexity, caused by the important number of components (e.g. software or devices) and their heterogeneity. The

Chapter in “Assistive Technologies in Smart Environments for People with Disabilities” (Taylor & Francis Group), 2016
complexity of the smart spaces is similar to the problems of the large enterprises that owned several servers and large applications on them (Talwar et al, 2005). Deploying systems that provide a “plug and play” way to provide service provision in smart environments, by managing all the configuration and device heterogeneities, can help to a broader deployment and usage of the smart environment technologies.

This book chapter focuses on the topic of the service provision, more specifically for software services (e.g. assistive software), in smart environments such as smart homes and smart urban environments. Therefore, the first section presents a review of the literature, from the first work on topic to the current state of the art. This review focuses on three groups of work: the researches around the service provision and interaction delivery, the self-organization and configuration of smart environments and the context-aware recommender systems, which in a way provide services to users and devices. The second section presents some technologies that can help in supporting the service provision: the OSGi framework, the OCAP platform and the Android operating system. The third section presents a case study of a middleware for providing dynamically services in smart environment, while coping with the complexity and the heterogeneity of the smart spaces: the Tyche middleware. We conclude this chapter with a discussion on the emerging trends.

Review of literature

In the first definition of the ubiquitous computing by Mark Weiser (Weiser, 1993), provision of services to the user was implicit with its “hundreds of computer per room”. By getting several tabs, pads and other interaction objects in an environment, the existence of algorithms that cope with the analysis of the current context and manage the deployment/running of the user interface was clearly a key element.

Chapter in “Assistive Technologies in Smart Environments for People with Disabilities” (Taylor & Francis Group), 2016
However, the first works to describe the service delivery or provision in smart environments were published around the beginning of this century, such as the Microsoft’s EasyLiving project (Shafer et al., 1998), Carnegie Mellon’s Aura project (Sousa and Garlan, 2001) or the BASE project (Becker et al., 2004).

**Service provision and interaction delivery in smart environments**

The EasyLiving project (Shafer et al., 1998) is a well-known project from Microsoft Research about the development of technologies dedicated to the smart spaces. About the service provision, the EasyLiving Geometric Model (EZLGM) 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 models the relation (with *measurements* which describe the position and the orientation of an entity’s coordinate frame) between *entities* and the EZLGM can also represent entities’ expense with the *extent* concept. Then, the EZLGM uses geometrical transformation to determine if there is relationship between entities. If the EZLGM can manage the spatial context of a smart environment, it doesn’t take in account a more complex environment context with user capabilities, preferences, device resources and capabilities.

The Aura project (Sousa and Garlan, 2001) from Carnegie Mellon University was a research on the transparent delivery of human-machine interaction in ubiquitous computing environments. The authors propose a framework that supports the interaction with users while they are in mobility in an environment. Their work is based on the concept of a personal Aura, which describe the user profile, its current activity, etc., and support the deployment of several types of interaction on different types of devices (e.g. Unix and Windows systems). Therefore, the Aura framework includes five modules that support the interaction provision to the users: “[…] first, the Task Manager, called Prism, embodies the concept of personal Aura. Second, the Context Observer provides information on the physical context and reports relevant events in the physical
context back to Prism and the Environment Manager. Third, the Environment Manager embodies the gateway to the environment; and fourth, Suppliers provide the abstract services that tasks are composed of: text editing, video playing, etc. From a logical standpoint, an environment has one instance of each of the types: Environment Manager, Context Observer and Task Manager.” (Sousa and Garlan, 2001).

In (Becker et al., 2004), Becker et al. present PCOM, a component based framework that was developed for the BASE project, which allow:

- The deployment of small applications on resource-constrained devices with the J2ME virtual machine;
- The management of the app’s life cycle;
- The dynamic adaptation of the device’s software depending on the environment’s available services.

The originality of PCOM is based on the adaptation mechanism of the system and the utilisation of “contracts” to describe the devices’ needs in system components. Therefore, each component (software module that regroups services and computation processes) defines with a “contract”, the services that are exported and the services that are required by the component to work. For instance, a component could require an input device such as a keyboard. This input device could be replaced by a SMS services to provide text inputs, if no keyboard are available. If the required components are not available, PCOM can hold component until a component giving the required service is available. The adaptation strategies of PCOM are built by the evaluation of the “contracts” are can be pre-defined by developers for specific situations (e.g. a system which require a specific keyboard model). The work of Becker et al. was among the first to propose a flexible and self-configurable system to support the provision of services among a collection of ubiquitous computing devices.
The O2S system (Paluska et al., 2003) proposes a solution similar to PCOM. The innovation of the system resides in the utilization of a decision tree to describe the states that must be meet to trigger the deployment of software components. In this decision tree, it is possible to describe the software dependencies toward external components, join actions to the states such as deploying specific components, verify and validate some contextual informations, etc. For each state defined in the tree, an action plan (“Planlet”) is linked. During the state evaluation, O2S verify the plan viability and if the constraints are respected with a reasoning engine developed in PROLOG. Unlike PCOM, O2S allow to integrate, more easily, constraints based on the contextual information (e.g. a user in a location triggers a service).

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., 2005) proposes a system to dynamically create end-user services through services composition. The service composition is initiated by abstracted 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.

Self-configuration and organization of smart environments

Works on the self-configuration of software in smart environments include several aspects of service provision. As part of the Autonomic Middleware for Ubiquitous Environments (AMUN) project (Trumler et al., 2004), the authors propose a middleware to facilitate the management and deployment of software components in smart environments by integrating autonomic computing-based features. AMUN integrates a control loop more or less based on MAPE-K approach of the autonomic computing (Kephart and Chess, 2003), with:
- A system event monitoring module;
- A knowledge base divided into three categories: information specific to applications (deployed applications, resources), information specific to the events from the monitored items (past events) and metrics on the same system (CPU usage, network usage, etc.);
- Some control algorithms;
- A configuration module using the system information and control algorithms to implement the measures in response to the sensed events.

The autonomous processes between AMUN devices are based on a strategy of choreography, where each entity has its own environmental manager of autonomy (control loop), which are grafted modules for managing communications with partners. The application deployment process of the participating nodes to the middleware is managed by the set of nodes, using a negotiation protocol. AMUN was used, among others projects, to deploy software in a Smart Doorplate project, A project offering information and assistance on screens installed on corporate office doors.

(Trumler et al., 2006) describes in detail an original negotiation protocol based on social behaviors in the distribution of tasks within a group. Under this protocol, a coordinating entity (itself being one of the participating nodes), distributes the list of applications that need to be deployed unto environment’s entities. They then evaluate their ability to run each application and, in turn, communicate its capabilities to other entities via the entity coordinator. Then, the coordinator node dispatch the software components that need to be deployed in the environment based on the entities availability.

As part of the Gaia project, Anand Ranganathan and Roy Campbell of the University of Illinois at Urbana-Champaign also offer computer-based mechanisms that deploy automatically software components in a smart environment (Ranganathan and Campbell, 2004). Given the complexity of

---

Chapter in "Assistive Technologies in Smart Environments for People with Disabilities" (Taylor & Francis Group), 2016
an environment software configuration, they offer a solution based on the STRIPS planning algorithm. This algorithm is used to find a deployment solution based on user needs, then start multimedia on devices (displays, speakers, audio players, etc.) present in a conference/presentation room. Based on a strategy of orchestration, an entity manages the configuration; the method proposed in GAIA could be adapted, with several changes to the internal mechanics, to other content as media such as applications, services and modules.

(Ranganathan et al., 2005) also proposes a second strategy for automatically managing the software provision in a smart environment, which this time is based on ontologies and semantic matching. In this solution, the goals of users are first adapted to the context of space. For example, if the user expresses the need to control a media from a mobile device, the system reflects this need through the list of mobile devices capable of meeting the need. Once the list produced, a semantic comparison is made between the specific needs of the user and descriptions of entities from the list. From this comparison emerges the configuration that must take the intelligent space to meet the user needs.

(Syed et al., 2010) propose an architecture for organizing autonomously software processes among devices of a smart space. To do this, the authors propose the use of an intelligent system which is based on a knowledge representation of the system entities divided into four types of data: recipes, capabilities, rules and properties. The recipes define the contexts in which the system can responds by applying rules, in reaction to a particular context. The capabilities are used to define the entities participating in the system and their functionalities. Finally, the properties refer to the capacity of entities which they define the presence or absence of devices, features, etc. For example, at the arrival of a request to play a song on one of the multimedia systems in a smart space, the system compares the context of the query with the contexts of basic recipes. If the conditions in the recipe are checked and there is the presence of a device with a music player (properties and capacity), a deployment policy is implemented. Similar to

Chapter in “Assistive Technologies in Smart Environments for People with Disabilities” (Taylor & Francis Group), 2016
Ranganathan’s approach, Syed et al. solution is based on the orchestration. The coordinator node, the intelligent algorithms and the knowledge base are centralized on a system in each smart space.

**Context-aware recommender systems**

As we presented in the introduction, it is possible to impose services to users (such as in Syed or AMIGO) or recommend them using different techniques. Using the context-aware models and recommendation algorithms to provide services or contents, (Adomavicius et al., 2011) was one of the first to propose context-aware recommender system which work on integrating contextual information in a multidimensional analysis of the users’ preferences (in collaborative filtering) depending of the period of the day. Other works has been done on location-based recommender system. For instance, (Levandoski et al., 2012) propose a solution based on three types of location ratings (spatial rating for non-spatial item, non-spatial rating for spatial item, and spatial rating for spatial item). The approach of Levandoski et al. is similar to the work of Adomavicius, where they used four-tuples or five-tuples to qualify the ratings and use multidimensional analysis techniques to compare ratings, but with an extended definition of the context.

(Shin et al., 2012) propose a system that analyzes the context and history of a smart phone for classifying the different installed apps depending on their probabilities of use. One of their conclusions is that the app transition data are one of the most important contextual information to predict which app will be used next. Such system can help in recommending services to the users depending on his current smart phone usage. Similar in their approach, (Huang et al., 2012) propose a system that predicts app usage based also on the mobile phone context. In their case, they focused on five contextual informations: last used app, hour of the day, day of the week, location and the user profile. The two kinds of information about the time are correlated with the location of the phone. About such recommender system, in one of our latest work on the analysis of the mobile applications usage on smart phone (Gouin-Vallerand & Mezghani, 2014), we conclude that the transition between application usages is a distinct contextual information versus
the probability of an application uses in a same periods of time. Our analysis shows also that the probabilities of transition become less obvious when the number of application usage occurrences is high (more an apps is used, more it is difficult to recommend it in a specific context). This conclusion also means that it can be relatively easy to recommend assistive services that are specialized, thus not use so often compared to apps such as the web browser or the mail application.

In conclusion, several researches have been done on the service provision in different domain of application, from smart homes to smart phones. The Table 1 presents a resume of this review of the literature under specific aspects: the research settings, the types of provision approach, the type of service provided and the used technologies. Of course, our overview of the domain does not include every works on the domain of service provision. Other works such as (Ghorbel et al., 2006) on the assistance, in ubiquitous displays (Kruger et al., 2012) or in augmented reality in smart environment (Shin et al., 2009) also include different aspects of context-aware service provision systems.

Table 1 – Comparison of research projects based on the research settings, the types of provision approach, the type of service provided and the used technologies

<table>
<thead>
<tr>
<th>Works</th>
<th>Settings</th>
<th>Provision approaches</th>
<th>Type of service provided</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>EasyLiving</td>
<td>Smart apartment</td>
<td>Orchestration</td>
<td>HCI and multimedia</td>
<td>Microsoft environment</td>
</tr>
<tr>
<td>Aura project</td>
<td>Office &amp; home</td>
<td>Orchestration</td>
<td>HCI for office &amp; home environment</td>
<td>Microsoft and Unix</td>
</tr>
<tr>
<td>PCOM</td>
<td>Smart environment</td>
<td>Orchestration</td>
<td>Components for data processing and user interactions</td>
<td>Java</td>
</tr>
<tr>
<td>Technology</td>
<td>Domain</td>
<td>Methodology</td>
<td>Components</td>
<td>Language</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>O2S</td>
<td>Smart environment</td>
<td>Orchestration</td>
<td>Components</td>
<td>Prolog and Java</td>
</tr>
<tr>
<td>AMIGO</td>
<td>Smart home for PwSN</td>
<td>Mix of choreography and orchestration</td>
<td>Assistive and home services</td>
<td>OSGi and Java</td>
</tr>
<tr>
<td>Trumler et al.</td>
<td>Smart office</td>
<td>Choreography</td>
<td>Office contents and software</td>
<td>Native</td>
</tr>
<tr>
<td>Ranganathan et al.</td>
<td>Conference room</td>
<td>Orchestration</td>
<td>Multimedia and HCI</td>
<td>Java, STRIPS planning</td>
</tr>
<tr>
<td>Syed et al.</td>
<td>Smart environment</td>
<td>Orchestration</td>
<td>Assistive Services</td>
<td>Java and case base reasoning</td>
</tr>
<tr>
<td>Adomavicius et al.</td>
<td>Web apps for content provision</td>
<td>Recommendation</td>
<td>Multimedia</td>
<td>Web</td>
</tr>
<tr>
<td>Levandoski et al.</td>
<td>Smart phones</td>
<td>Recommendation</td>
<td>Mobile applications</td>
<td>Android OS</td>
</tr>
<tr>
<td>Shin et al.</td>
<td>Smart phones</td>
<td>Recommendation</td>
<td>Mobile applications</td>
<td>Android OS</td>
</tr>
<tr>
<td>Ke et al.</td>
<td>Smart phones</td>
<td>Recommendation</td>
<td>Mobile applications</td>
<td>Android OS</td>
</tr>
</tbody>
</table>

**Review of technologies**

Different technologies exist to support the provision of services, depending on targeted devices. As discussed in the introduction, the heterogeneity of the devices compels for technological solutions that are multiplatform and usable on several types of hardware (imbedded devices, smart phones, etc.).
**OSGi framework**

One example of such solution is the OSGi framework specification. The OSGi Alliance, which was better known as the Open Service Gateway initiative, is an enterprise consortium, regrouping companies such as Ericsson, Nokia, Siemens and IBM, which works together to create an open specification (OSGi, 2014) for a service oriented software platform. This specification defines every parts required for a fully functional SOA platform such as mechanisms to manage the life cycles of plugins and services, update mechanisms for services and plugins, services’ description and discovery, etc. OSGi is well known to be the software base of the Eclipse IDE plugin system, Oracle Weblogic Web Server or JBoss Application Server. However, at the beginning, the OSGi specification was primarily created to reflect the requirements of the consortium’s members to be able to deploy modular software in a fast and easy way. This possibility, to deploy software rapidly and efficiently, attracted several researchers and enterprise to develop service provision systems.

As mentioned earlier, the OSGi specification is based on the Java language technology. It particularly uses the Java language introspection and class loading mechanisms to instantiate modules and services. In OSGi, a module is called a bundle, which is a typical Java jar compressed file with specific data in its manifest file and specific classes. This manifest file, for instance, determine the role of the bundle, its activation class (or main class), code package needed, code package exported, etc. Each bundle exports or imports codebases or services, depending of their functionalities, to other bundles, creating complex functionalities such as GUI or WebServer. For instance, Weblogic, Websphere and the JoNAS web server are based on the OSGi framework and where each web application represents one or more bundles.

In OSGi, a specific service is a bundle’s class instance that offers methods (i.e. functions), which can be called by other OSGi bundles. What is interesting in the OSGi SOA model, is the capabilities to create dynamic relation between modules, add, update or remove in runtime.
several bundles while reducing the impact of these actions on the quality of service. The exchange of data and utilization of services are managed by four specified layers:

- Security layer: manages the bundle validities through signature validation and hashcodes;
- Module layer: manages the loading of the codebases and their executions;
- Life cycle layer: manages the life cycle of bundles;
- Service layer: allows service exchanges between modules.

Several additional specifications or specification versions exist, which provide additional capabilities or mechanisms. For instance, the OSGi Bundle Repository (OBR) defines mechanisms to automatically manage the bundles dependencies towards other bundles that provided required codebases or services. For instance, a bundle that would instantiate a J2EE website would require a webserver instance prior to its instantiation. OBR gives the functionalities to define such dependencies and will manage the installation, starting and instantiation of a webserver bundle, prior to installing and running the J2EE website bundle.

Without being as complete as the Debian Package system, OBR gives the required functionalities to reduce the overhead related to the management of the functional dependencies. Moreover, the distributed OSGi specification (OSGi v4.2) give the specification to automatically build and instantiate Web Services version of the OSGi services, allowing remote invocation of the services by other OSGi platform or other systems. The Apache foundation projet CXF proposes a set of OSGi bundle implementing the specification. A similar specification exists for UPnP and there is also an extension to support DPWS on OSGi.

Several implementation of the OSGi platform exists with Apache Felix(Apache Felix website: http://felix.apache.org/), Equinox OSGi (Equinox website: http://www.eclipse.org/equinox/) and
Knopflerfish (Knopflerfish website: http://www.knopflerfish.org/) being the most popular open source implementation and Prosyst mBS (Prosyst website: http://www.prosyst.com/) one of the last remaining commercial versions of OSGi available. In the last years, Apache Felix took an important part of the user market and proposes several pre-build bundles such as the web server Jetty, a UPnP base driver, a Web management console, etc.

In the Pervasive and Ubiquitous computing research community, several works had been done that uses OSGi to support service-oriented architectures. For instance, (Gu et al., 2005) are among the first authors to write on the integration of OSGi to ubiquitous computing system. In their paper, Gu et al. propose the utilization of OSGi as the backbone of their Context-aware system and includes an OWL description and reasoned to automatically deploy assistance services. Moreover, (Vallee et al., 2005) in the context of the European project AMIGO, propose a system to dynamically create end-user services through services composition. The service composition is initiated by abstracted 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. OSGi is the core platform of the AMIGO project, but includes other technologies such as .Net and UPnP. Finally, the Tyche project, which is further describe in the next section, uses also the OSGi platform as the base of middleware to support the autonomous configuration of smart environments based on the context, the users’ profile and the taxonomy of the environment.

**OCAP framework**

The Open Cable Application Platform (OCAP) is a software platform based on the Globally Executable standard Multimedia Home Platform (MHP-GEM), to manage and deliver television services to cable television customers. Designed by Cable Television Laboratories, a research consortium cable, OCAP is both an operating system for a gateway to cable TV and a middleware for the management of services to customers.
Developed from the Java object-oriented language version of Micro Edition (J2ME), OCAP is a middleware that can be deployed across multiple types of gateway, provided there is a corresponding functional J2ME virtual machine hardware. Using the television cable, it is possible to send management requests or update software components to OCAP gateways. This middleware allows total control of the life cycle of applications for installation, starting, shutdown, uninstalling, and updating of applications. OCAP mainly uses the Push method to route management applications, i.e. that the requests are sent from managers to gateways. Moreover, the adopted management strategy is the orchestration, i.e. that it is the managers who manage the content of gateways.

The OCAP middleware allows to remotely manage the entire life cycle of applications deployed on gateways to cable television. It also offers useful tools for logging and monitoring of deployed applications. This solution has been designed for a specific type of system, television gateways, and is more or less usable for the service provision on other type of device.

**Android Operating System**

The last but not the least is the Android OS that is deployed in a large range of end-user devices: smart phones, tablets, television set top box and many other devices. The Android OS is based on a Linux Kernel and is developed by the enterprise Google. About the service provision, the Android OS includes several features that help to support the delivery of software components to users. Firstly, most of the software applications in Android are packaged within a packaging file format called *Android application package* (APK), which is pretty similar to the Debian package system. APK files are bundled as JAR files and included the meta-data needed to run each application in its manifest file. Android applications are usually developed in the Java language that needs to be compliant with the Android Runtime (previously the Dalvik VM), but can also include other code or scripts such as HTML5 content. APK packaging system eases the deployment and management of new applications in Android OS devices. To support even further...
the deployment of APK on Android devices, most of the Android OS version (except some version that are not compliant with Google) include the Google Play application, which is a digital distribution system which host the different Android applications available, manage the software version and the dependencies between the different applications, the available API and the OS versions. In a way, the Google Play application is also similar to the Debian dpkg system. Thus, a service provision system on Android could use the Google Play application via its API and use the different services to deploy new APK, manage the dependencies and the available versions.

As we present in the review of the literature, several researches has been done on context-aware recommender systems on Android devices. These works used the available contextual information provided by the operating system, the usage history and data from embedded sensors to recommend already installed APK or other applications available.

Finally, it is also possible to deploy the OSGi framework on Android devices and deploy adapted bundles using the OSGi functionalities. Among others, it is possible to use Apache Felix, Knopflerfish and Prosyst mBS OSGi framework on Android OS. However, the usual Java code included in OSGi bundles need to be compliant with the Android Runtime and been compiled with the development kit of the Android Runtime. Therefore, it can be difficult or time consuming to adapt and convert existing OSGi bundles for the Android platform.

**A case study: the Tyche project**

The Tyche project (Gouin-Vallerand et al., 2013) is a distributed middleware that is deployed on device nodes within smart spaces such as apartments / residential houses and allow the deployment and management of software on environment nodes based on the device capabilities and users' profile. To automatically manage the service provision, the middleware analyze the contextual information of the environments, provided by the different device nodes and sensor...
networks, to find which devices would fit best for hosting the services. The middleware is based on the OSGi framework and its service-oriented approach; it is then implemented in the Java language.

Before explaining in detail the middleware, a brief example illustrating a service provision would help to understand which kind of service provision and for which type of scenario the Tyche middleware is able to provide service delivery in smart environments.

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 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 1 illustrates this example with a map of a 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

Chapter in “Assistive Technologies in Smart Environments for People with Disabilities” (Taylor & Francis Group), 2016
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.

Figure 1: A smart apartment overview illustrating the service provision scenario

To fulfill such scenario, the Tyche’s reasoning mechanism uses four main context’s elements to deploy services toward the users: the environment device profiles, the logical configuration of the environment, the user profiles and the software profiles. Each software application that needs to be deployed or managed in the environment has their hardware, software or contextual needs. For instance, assistive applications like user adapted agendas or a cooking assisting applications (Giroux et al., 2008), can target particular users in the environments and can required specific peripheral devices. On the other hand, users have physical capabilities and preferences about the environment devices and devices have also a profile with capabilities i.e. their resources, connected peripheral devices, etc. Finally, all these components are present in the smart
environments at different (or not) locations and are related to contextual zones like the kitchen, the bathroom, the living room, etc.

Therefore, the goal of Tyche service provision mechanism is to manage all these information and find the optimal organization scheme for the service to provide. Tyche functionalities are implemented in a reasoning engine, the Fuzzy Logic Organization Reasoning Engine (FLORE). The objectives of the FLORE are to match the needs of the applications to deploy in a ubiquitous environment with the context of the environment. As the environment context is a mix of quantitative and qualitative data; the Fuzzy Logic (Ross, 2004) allows to do high level reasoning in a 'fuzzy' perspective, where contextual information are processed following a set of fuzzy rules. Thus, Fuzzy Logic 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 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 (assignment 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.

Figure 2: Example of the fuzzy reasoning of the FLORE
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 peripherals are computed according to the service needs. The results of the micro-context computation are shared with the macro-context layer. Each nondedicated 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.

– 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.

The output of the FLORE, ranging from 0 to 100, represent the Device Capabilities Quotients (DCQ), a metric value which representing the viability of a device face to the software application needs. More the quotient value is high; more the device is close from the optimal device target, considering the device’s resources and its context.

Concerning the overall architecture of the Tyche Project and its implementation, the middleware has been built on the OSGi framework and use WebService-* standards to communicate between the environment nodes (web services and WS-Discovery). OSGi gives the support for the modularization of the ubiquitous applications and the functionalities to support the management of their life cycles. On the top of the OSGi framework, we have implemented several modules that are working together to provide the service provision (Figure 3).
Figure 3: The architecture of the Tyche middleware

The middleware includes two central components, an ontology and a reasoning engine, which work in tandem to find the best software organization for a group of applications to be deployed, and, from the context of smart spaces. The reasoning engine called Fuzzy Logic Organization Reasoning Engine (FLORE) uses the description logic and fuzzy logic to give an evaluation metric, the Device Capability Quotient (DCQ), of each device versus the context of the environment. The FLORE is divided into two sections: one for the reasoning about the overall context of the environment (topology, user), the Organization Reasoning module, and the reasoning on the context of each device (hardware feature, location, peripheral, etc.), the Device FLORE module. These two parts of FLORE deal respectively with the macro-and micro-context (Gouin-Vallerand et al., 2012).

The ontology is instantiated and managed by the ontology management module. It is used to define the concepts of intelligent spaces (T-Box) such as devices, users, zones, and their properties and relations between these concepts. It also serves to store instances of these concepts (A-Box) such as a user's profile or the profile of a particular device. The ontology was
implemented using Web Ontology Language (OWL) and its concepts and instances were instantiated using the semantic platform Jena [Jena].

The implementation of the FLORE is divided into two parts deployed in the node coordinator and devices nodes of the smart spaces. The FLORE-D evaluates the performance of the devices versus the needs of an application regarding the use of resources, the area where the device is located and the presence of peripheral devices (e.g. keyboard, mouse, camera, etc). The FLORA-C evaluates at first the basic needs of an application versus the devices resources. If necessary, it assesses in a second step the user profile and context around the user versus the devices and their peripherals. Finally, he merges the DCQ from the two FLORE entities to form the final DCQ.

The FLORE algorithm has three steps according to the needs and characteristics of the applications to deploy: the general step of reasoning on the micro-context, the step about the device and user location processing and the step about processing the user profile. The first step aims the deployment of services that has resources and peripheral requirements, but does not include a deployment in a particular area of the intelligent space. In this step, the major part of the reasoning is on the micro-context of each devices and is therefore done by them.

The second step is the deployment of an application to a particular area of the environment, near a user. In this case, the coordinator node performs an evaluation between the devices in a specific zone and the service requirement. This evaluation uses the semantic links between the smart environments’ zones to define that are the best device to fulfill the software needs. This assessment is then returned to the devices nodes that include the assessment of the first DCQ.
The third step is the deployment of an application for a particular use by users of smart spaces. This assessment then integrates the user profile: its characteristics, preferences and capabilities and profiles of data versus the context of the environments in the calculation of the DCQ.

The sequence diagram in Figure 4 shows the general operation of the middleware during a service provision request. The Software Management Tool initiates the service provision request; an API used by external application send the software deployment requests. To simplify the diagram, only one device node is presented.

![Sequence diagram showing service provision process](image)

**Figure 4 - Different steps during the service provision process**

Thus, the Environment Manager Module receives the request from the user’s tool and according to the type of the requests, forwards them to the FLORE Coordinator Module. The FLORE Coordinator module:

1. Uses the Ontology Manager Module to browse the ontology for context information;

Chapter in “Assistive Technologies in Smart Environments for People with Disabilities” (Taylor & Francis Group), 2016
2. Send a reasoning request to the FLORE Device Module to reason about the micro context of the environment devices. A first DCQ related to the micro-context is returned to the FLORE Coordinator module;

3. Evaluates the devices viability (DCQ) from the macro-context and merge it with the results from the micro-context (prior DCQ);

4. Depending of the set minimal threshold DCQ, select the device with higher DCQ as the most viable device for the software deployment.

For each software application to deploy contained in the service provision request, the Environment Manager Module ask to the FLORE which device is the most viable and send a final software deployment request to the Device Management Module of the selected device.

*Reasoning of the user profile and context*

Sections of FLORE that are the most complex are those dealing with the user profile and calculating the related DCQ. The DCQ is calculated in the case where an application deployment request target particular user in a smart space. The DCQ related to the user profile/context is combined to the DCQ related to the device profile, to form the final DCQ that is used by the FLORE to provide the service. The reasoning on the user's profile versus the context of smart spaces and applications’ resources requirements is based on four types of data: its interaction capabilities, its location in space, orientation and preferences in terms of usable peripheral devices. Figure 5 represents the contextual information used by the FLORE in its organization reasoning and in its DCQ attribution. This information are managed by an ontology written in the Ontology Web Language (OWL) [1] and implemented in the semantic framework Jena. [12].
The FLORE base its evaluation of user interaction capabilities with the environment on the users’ interaction modalities: the senses, perceptions, motor senses and cognitive abilities (Obrenovic and Starcevic, 2004). More particularly, the current version of Tyche and its FLORE uses the following interaction terms:

- **Sense - Vision:** the field of view 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 - Locomotion:** the user’s locomotion capacity versus the devices and peripheral locations;
- **Motor - Manual Interaction:** the user physical capacities versus the peripheral devices physical needs: hand force and hand workspace.
These modalities represent the traditional way to interact with computing devices: the vision and the sense of touch.

Of course many other methods exist and could be evaluated as users’ hearing capability versus the volume of speakers, the speech strength versus the sensitivity of a microphone or cognitive aspects such as the language used by Human-machine interfaces of applications and devices. The integration of interaction modalities in the service provision process represents an innovation in the domain. In this work, the assessment of user interaction modalities is reduced to only two physical abilities: hand strength and opening them. The type of evaluation is categorical; users have or do not have the interaction capabilities to interact with a device.

**Physical motricity:** Initially, users' ability to interact with the peripheral devices are evaluated. This evaluation is made by comparing the physical capabilities of users and the hardware specifications required to use the devices in question. The FLORE is therefore based on the work of Kadouche [58] in the field, checking the strength of the hand and the workspace users’ hands i.e. the degree of opening of the hands. If this check fails, the devices directly receive DCQ worth 0 points. For instance, if a specific user have a limitation to its hand and can only press keyboard touch with a pressure of 1 newton and a keyboard need 2 newtons to be used, the device is therefore identified as unusable and aDCQ of 0 is attributed.

**Visual acuity:** The FLORE used in a second step, the user's field of view and projection field of devices to determine if the user targeted by the service provision has display devices in his field of vision. These devices would be considered as priority by the FLORE in its calculation of the
DCQ. While the projection field of display devices is characterized in the contextual description by an orientation degree and a degree viewing angle. On the other hand, the user field of view is defined by the user's orientation in degree and two angles of view corresponding to each eye of the users. Thus a user can have a normal field of view for the right eye and a reduced field of vision for the left eye or vice versa. Verification of field of view is relatively simple and uses orthonormal plan changes to the user position and orientation, and then with the display devices.

Thirdly, the FLORE assesses the ability of users to interact with applications, by calculating visual acuity ratios. This ratio value range between [0, 1] and is calculated from the visual acuity of the users, their position, the average size of dialogues characters in the service to deploy, the position of the display devices, the size of the display devices and their resolutions. The objective of this ratio is to quantify the ability of a user to read the text information on an application from its position in the environment. Obviously, users will probably have to move to the devices where the applications are deployed to interact with them, but it can check if the users could be able to recognize the applications and to have minimum information thereon.

The ratio calculation therefore uses the average visual acuity of the users expressed with the Snellen scale (Muraoka and Ikeda, 2004). This scale is widely used in optometry for quantifying visual acuity by verifying the ability of a person to read a character from five arcminutes (height and width) to a traditional distance of twenty feet, or the famous view 20/20 or 6/6 (in meters). In the case of a person with an acuity of Snellen of 10/20, it must be at a distance of ten feet to be able to read five arcminutes of twenty feet distance, while a person with an acute 20/20 recognizes the normal distance of twenty feet. The five arcminutes correspond to scale the size of a circular arc of a five-minute corner for a circle of radius 20 feet, equivalent to a character and a
height and width of 8.9 millimeters. The optotype of the user adapted to the distance of the display devices is compared with the character size of the applications on such devices. The higher the ratio the closer are the characters of the users’ optotype as when the ratio approaches 0, more the users have difficulty to read characters on the devices from their position in the space. Ratio values are bounded on [0, 1] to prevent large displays with a small resolution to have very high ratios and favoring some. Thus, whether users are able to read characters (1 ratio) and either the users are more or less incapable (ratio of 0 or more).

**User mobility:** The fourth mode of interaction is assessed by the mobility of the user. This assessment is made largely using fuzzy logic and actually contains two interrelated sub-evaluations. The first sub-evaluation is to qualify the user speed. To do this, we made a literature review of research on measures of user movement speed. Among the different revised documents, we kept a research report on the walking speed of urban pedestrians (Carrey, 2005) and an article on the evaluation of the mobility of older persons (Abellan et al., 2009), in order to categorize and qualify the mobility of users. We have therefore divided the travel speeds into four types: not moving, slow moving, normal speed and finally fast walking. Each of these types of speed was associated with a normal Gaussian function that quantifies the level of membership to a correspondent walking type. The means and standard deviations were drawn from our literature review for each Gaussian functions. The association of travel speeds users of these membership functions is used by the fuzzy logic controller FLORE-C for its calculation of the user DCQ.

The second sub-evaluation concerns the time allowed for a user to move to a device. This sub-evaluation therefore calculates the time in seconds to reach a device using the average speed of travel and the distance between the user and the device. Travel time is then described in a fuzzy
set which four membership functions are present: instantaneous travel, fast travel, slow travel and long travel. Instant travel has an average travel time between 0 seconds and 2 seconds, fast travel with 10 seconds with a 2 seconds of standard deviation, slow travel with 20 seconds and 2 seconds standard deviation and finally long travel with 30 seconds and a 2 seconds of standard deviation (30 seconds and higher travel time have a 100% membership degree to the long travel function). Like the first undervaluation of user mobility, the degree of membership to these functions is evaluated by the fuzzy logic controller of FLORA-C. The Figure 3 shows these membership functions in the fuzzy set.

![Graph showing membership functions](image)

**Figure 6**: Representation of the membership function related to the mobility of the users

**User preferences**: The user peripheral preferences for specific peripheral devices are formulated with a Likert scale to classify the environment devices. Thus, each user is giving some of their preferences.
usability preferences toward the device peripherals with values as: user likes to use the peripheral device = 1, user is neutral face to utilization of this peripheral device = 0, user dislikes to use the peripheral device = -1. The sum of the user’s peripheral preferences for each device are calculated and used in the DCQ evaluation. These preferences can be used as a complementary tool to the physical capacity to determine the types of devices that potential users will be able to employ. In the current version of Tyche project, five types of devices were used: keyboard, mouse, trackball, virtual keyboard and touch screen slider. Other kinds of devices can easily be added to the middleware’s ontology.

In conclusion, the interaction modalities and user preferences are injected into the fuzzy logic controller that compute the user DCQ for each device of the environment that have the minimum resources to run the applications to provide. These data are combined with the fuzzification process, using reasoning rules, and then the user DCQ is calculated using the centroid of the results of all of defuzzification. The fuzzy logic controller FLORE-C thus includes five fuzzy sets, seventeen membership functions and forty reasoning functions. The Figure 7 shows an example of some of these fuzzy reasoning functions in the Fuzzy Control Language (FCL IEC 61131 part 7) related to the fuzzy logic implemented in the FLORE-C.
FUNCTION_BLOCK DCQUser //Block definition (there may be more than one block profile)
VAR_INPUT //Define Input Variables
  visionAngle : REAL;
  visualAcuity : REAL;
  userMobility : REAL;
  time : REAL;
  preference : REAL;
END_VAR
VAR_OUTPUT //Define Output variable
dcq : REAL;
END_VAR

//
// Fuzzification
//
FUZZIFY visionAngle
  TERM behind := 0 . 0 ;
  TERM inFront := 1 . 0 ;
END_FUZZIFY

FUZZIFY preference
  TERM dislike := gauss -1.0 0 . 2 5 ;
  TERM neutral := gauss 0 . 0 0 . 2 5 ;
  TERM like := gauss 1 . 0 0 . 2 5 ;
END_FUZZIFY

FUZZIFY visualAcuity
  TERM impossible := gauss 0 . 0 0 . 0 8 ;
  TERM hard := gauss 0 . 3 3 0 . 0 8 ;
  TERM borderline := gauss 0 . 6 6 0 . 0 8 ;
  TERM optimal := gauss 1 . 0 0 . 0 8 ;
END_FUZZIFY

FUZZIFY userMobility
  TERM impossible := gauss 0 0 . 2 2 ;
  TERM slow := gauss 0 . 8 0 . 1 8 ;
  TERM normal := gauss 1 . 4 3 0 . 1 ;
  TERM fast := gauss 2 . 2 0 . 2 8 5 ;
END_FUZZIFY

FUZZIFY time
  TERM verySlow := gauss 3 0 . 0 2 . 0 ;
  TERM slow := gauss 2 0 . 0 2 . 0 ;
  TERM fast := gauss 1 0 . 0 2 . 0 ;
  TERM instant := gauss 0 . 0 2 . 0 ;
END_FUZZIFY

DEFUZZIFY dcq
  TERM impossible := gauss 0 . 0 6 . 0 ;
  TERM notOptimal := gauss 3 3 . 3 3 6 . 0 ;
  TERM subOptimal := gauss 6 6 . 6 6 6 . 0 ;
  TERM optimal := gauss 1 0 0 . 0 6 . 0 ;
METHOD : COG; //Use ' Center of gravity (centroid) method
  DEFAULT := 0 ;
  RANGE := (0 . . 100) ;
END_DEFUZZIFY
RULEBLOCK No1
  AND : MIN; //Use 'min' for 'and' ( also implicit use of 'ax for 'or' to fulfill the DeMorgan’a law)
  ACT : MIN; //Use 'min activation method
  ACCU : MAX; //Use 'max ' as accumulation method
RULE 1 : IF visionAngle IS inFront AND visualAcuity IS optimal AND userMobilityIS fast AND
time IS instant AND preference IS like THEN dcq IS optimal WITH 1 ;
RULE 2 : IF visionAngle IS inFront AND visualAcuity IS optimal AND userMobilityIS fast AND
time IS instant AND preference IS dislike THEN dcq IS notOptimal ;
RULE 3 . 1 : IF visionAngle IS inFront AND visualAcuity IS optimal AND userMobilityIS NOT fast
AND time IS instant AND ( preference IS like OR preference IS neutral ) THEN
dcq IS notOptimal ;
[…]

Figure 7 : Excerpt of the fuzzy logic rules of the FLORE-C
Discussion

The Tyche project proposes a novel middleware to support the deployment and the organization of software in smart environments such as smart homes. It uses different types of contextual information in its reasoning process and by using the Java language and the OSGi framework, it cope with some difficulties related to the heterogeneity of the smart space devices.

Thus, the adoption of the OSGi platform as technological support to the implementation of Tyche was a wise choice. The dynamism provided and the reducing of the software coupling enabled rapid development, easy modularity and the implementation of several interchangeable solutions. The modular division of OSGi applications and lifecycle control has also played for a lot in the conception of the architecture of Tyche project. Also, the utilization of standard WebService-* (eventing, discovery, etc.) was a major element in the architectural choices, and in the implementation of middleware and service provision API. Web services have allowed a multiplatform use of Tyche and expandability that other standards or protocols do not offer (Juxtapose, RMI, Jini). However, in the current version of Tyche project, the utilization of WS-* standards, the Jetty web server and Apache CXF API represents a computational load and a large utilization of the devices’ memory. In particular, CXF which requires at minimum a virtual machine Java Standard Edition 1.6 (or OpenJDK 1.6) restricts the possibilities of use on some hardware and operating system. Thus, it is impossible to run both types of Tyche node on devices with the Android operating system. However, Tyche includes bundles that use kSOAP to create accessible web services on Android.

The service provision reasoning engine, the FLORE, of the Tyche project uses contextual information in a microscopic and macroscopic context approach. The various contextual informations on users, applications, devices and other media concepts were described using OWL and RDF. This information is stored in an ontology in every smart space where Tyche is
deployed, which it is possible to make queries, data mining, inference concepts and instance, etc. The utilization of OWL / RDF allows a description of standardized bodies according to the concepts defined in the DOMUS labs, independence from hardware architectures and increased capacity for extension. Moreover, since the OWL / RDF is actually an extension of XML, it can be used in multiple systems where the deployment of the Jena framework is impossible, as it was done with the FLORE-D.

Through the user descriptions and the FLORE, the Tyche project integrated the user profile to the service provision process by taking into account the abilities, preferences and user contexts. The Tyche reasoning model calculates the DCQ capabilities of a device to host applications by dealing with user profiles and applications needs. How user profiles were incorporated into the reasoning process, the intensive use of fuzzy logic for the service provision and use of DCQ represent an innovation in the community working on the service provision and software organization in smart environments. Although the model presented is limited in term of covered modalities, milestones for their use and their inclusion in service provision have been laid and the model can be refined and extended in future work.

If the Tyche middleware offers great mechanisms to support the service provision, there are still a lot of functionalities to develop and research problems to resolve. Firstly, while software applications are deployed in the environments, resources in the devices are consumed like CPU, RAM or hard drive. Thus, each request’s results are dependant from the prior provision requests. Our way to deploy the various software between the environment devices, create a priority hierarchy between the applications to deploy. When it’s well used, this priority can be useful, allowing important or critical applications to have higher priorities than other applications. However, this kind of organization doesn’t find the optimal organization solution. The overall problem of the software organization is of NP complexity, the optimal solution would be find in...
polynomial time $O(kn^2)$ where $n$ is the number of devices and $k$ is the number of applications to deploy. The current algorithm is faster than the optimal solution with $O(kn)$, but does not offer the optimal one.

Secondly, as it was presented in the review of literature, several works has been done in the last five years about the recommendation of mobile applications. As the usage of smart phones and tablets become pervasive, recommender systems will be more and more used. The service provision approach used by the Tyche middleware is unilateral, the users does not have any way to specify which application they would prefer to assist them in their daily activities. Therefore, it would be interesting to integrate recommending algorithms that would take integrate more the user preferences but also used the user’s history and data mining algorithms to refine the service provision mechanisms.

About the covered interaction modalities and without going into specific cases, the FLORE reasoning process on the user profile should be extended to include modalities related to the hearing, but especially the cognitive aspects of user, since the Tyche middleware aim, among other, the assistance to people with cognitive impairment. A simple method would be to implement language processing used between devices (operating system), users and applications. On the other hand, a measure for quantifying or qualifying mnemonic capacity or functionality and/or users general cognitive capacities versus the applications functionalities would contribute to a more adapted service provision. Such metrics exist in neuropsychology in the form of results of cognitive functions tests such as the Mini-Mental State Examination (MMSE) (Folstein et al., 1975) or the Disability Assessment for Dementia (DAD) (Gélinas et al., 1999). However, it would be required to be able to quantify or qualify the required cognitive capacity for the utilization of devices or software, which is not an easy task. Such work requires, a priori, an extensive research with neuropsychology and usability / ergonomics aspects.
Finally, the implementation of Tyche project is based on the hypothesis of deployment in closed and controlled smart environments. With a goal to widen its use as in smart urban environments, with mobile users, other interaction modalities, mechanisms of security and intensive multi-interaction devices (such as public displays) would be required.

Conclusion

Ambient intelligence technologies deployed in living environments such as homes and apartments can assist users in their activity of daily life through context-aware and personalized assistive services. With the aging of the population in the majority of the developed countries, for instance 25% of the Canadian population will be aged of 65 years old and over in 2036 (icis, 2011), several studies shows that there will be important impacts on the societies: raise of the healthcare costs, difficulty to find specialized labor, reduction of the quality of life in several senior citizens, etc. Service provision systems deployed in smart environments and on mobile devices can help to assist users with special needs, give them more autonomy and impact positively on their quality of life.

Several researches has been done on the topic of the service provision in specific scenarios such as the assistance in smart homes, smart office initiatives or for mobile devices. The Tyche middleware proposes a service provision system for smart environments 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. If the utilization of technologies such as the OSGi framework, the WS-* stack, the OWL ontology and the fuzzy logic allow an easy adaptation and deployment of the system to any types of smart environments, several additions to the framework are required to cover fully the users’ interaction modalities, the different topologies of the smart environments, the security/privacy of users and so on. Thus, as for Tyche, we feel that a lot of work is required
by the scientific community before we can see real deployment of service provision systems on end-user devices/environments. With the idea of broader utilization of such technologies, we believe that the mobile technologies and the approach of context-aware recommender systems is a particularly interesting field of research for the service provision. In that way, we are actually working on this aspect and we already published some of works on the topic [Gouin-Vallerand and Montero, 2013], [Gouin-Vallerand & Mezghani, 2014]. Also, with the increasing popularity of smart technologies, connected devices (internet of things) and initiative such as the Smart Cities, we believe that service provision systems can help users in their daily activities by providing the support to deliver cognitive support to the users and improve the user experience with theirs environments. Context-aware applications will be more and more used by the public and mechanisms to deploy and manage such software will require and the research works presented in this chapter are going to be the scientific base for such systems.

References


goalorientated programming semantics. In Workshop on System Support for Ubiquitous 
Computing (UbiSys’03), 5th International Conference on Ubiquitous Computing.

Ambient Intelligence, Lecture Notes in Computer Science, Vol. 3295. Springer 
Berlin/Heidelberg. 48–159.

International Conference on Autonomic Computing. 80–87

ubiquitous computing, Multiagent Grid Syst. 1(2), 109–129.

Ratti, C., and Townsend, A. 2011. The social nexus, Scientific American, August

Disease Patients Desire in Smart Home Technologies? Contrasted results of a wide 


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

Ryan, N., J. Pascoe, and D. Morse. 1997. Computer Applications in Archaeology, Chapter in 

easyliving project at microsoft research. in: Proc. Joint DARPA/NIST Smart Spaces 


