Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Multi-Level Reconfiguration in the DANAH Assistive System

Saïd LANKRI
European University of Brittany
Lab-STICC Laboratory
F-56321, Lorient, FRANCE
said.lankri@univ-ubs.fr

Pascal BERRUET
European University of Brittany
Lab-STICC Laboratory
F-56321, Lorient, FRANCE
pascal.berruet@univ-ubs.fr

Jean-Luc PHILIPPE
European University of Brittany
Lab-STICC Laboratory
F-56321, Lorient, FRANCE
jean-luc.philippe@univ-ubs.fr

***Abstract*—Nowadays, interaction with our surrouding environment has increased due to the presence of numerous devices that provide us with services. This is especially true in Smart Homes and can be of great help for the disabled people and the elderly that can no longer perform daily tasks they used to. However, in case of failure, corrective actions can be heavy to take, thus the need for the system to recover by itself and ensure service availability. Service availability is provided through service reconfiguration. This papers deals with service reconfiguration in smart homes. It presents a multi-level approach in which both off-line and on-line reconfiguration schemes are used to gradually recover from failed services. Static, effect-based, path and resource reconfiguration levels are described. They have been successfully implemented in the DANAH assistive system, which combines both navigation and service provision for smart homes.**

***Index Terms*—Service Reconfiguration, On-line/Dynamic Reconfiguration, Assistive Systems**

I. Introduction

The rapid deployment of computer software and hardware leads to the emergence of complex environments where sensors, networks, devices and human machine interfaces are seamlessly integrated to enhance the way people interact with their environments [1], possibly with the addition of some intelligence that makes them smart. The Ambient Intelligence paradigm (AmI) [2] represents a new paradigm shift in which the user interacts with more than one computer device within the same environment that offers a great amount of useful *services*. It contrasts with the historical paradigm of the *mainframes* in which several people worked on the same computer, and with the *personal computer* paradigm in which each user possesses one computer. Due to the large amount of the immersed systems within these environments, it becomes more difficult to predict their evolution and the failure probability rapidly increases. In order to provide *service availability*, one can either design safe systems by eliminating failures as in avionics, or let failures happen and address them. In pervasive environments, systems cannot be safe by design since the environment, which is uncertain, is a part of the system itself. Therefore, these systems must be built with *reconfigurability* in mind. *Reconfigurable Systems* are those capable of rapidly adapt themselves to context and objective changes, including market requirements, service failures and new quality standards.

This paper deals with *service reconfiguration* in Smart Homes [3]. An Assistive Technology System (ATS) named DANAH [4] has been implemented to provide environmental control, autonomous navigation and service reconfiguration in domotized living spaces. The services are advertised to the user through a suitable interface, which selects the desired service and DANAH shall deliver it. When a service fails, DANAH follows a gradual reconfiguration process based on static, effect-based, path and resource compensation to ensure service availability. It combines both off-line reconfiguration through statically defined reconfiguration rules, and on-line reconfiguration that dynamically computes a new configuration using current system status.
This paper is organized as follows. Section II gives a general overview of reconfigurable systems in the fields of electronic engineering and computer science. Section III focuses on reconfiguration specifically in ATS and section IV deals with reconfiguration in the DANAH ATS. It presents the underlying service model and the reconfiguration process. Finally, section V presents the DANAH experimental platform.

II. Related Work

The meaning of the term *reconfiguration* in a reconfigurable system may vary with the domain to which it is applied. However, the aim of a *reconfiguration process* is to perform changes to the system to ensure correct suitability to the current context. Changes can be made to the software, the hardware, or to both, and are triggered due to either objective changes (adaptivity), or to service failures (fault-tolerance).

A. Reconfiguration in Electronics

Reconfigurable electronic systems [5] are nowadays widely present thanks to the underlying hardware technologies such as Field Programmable Gate Arrays (FPGAs). FPGAs contain programmable logic cells that allow designing customized systems specially tailored for application needs. They are configured by downloading a circuit description (called a *bitstream*) into the FPGA. This allows reduction in component count (and hence cost), improved time-to-market and improved flexibility and upgradability.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

FPGA Reconfiguration consists in changing totally or partially (partial reconfiguration) the bitstream they contain. Reconfiguration allows one to adapt the bitstream with regards to performance issues (e.g. satisfy higher data troughput) or to power saving issues (better battery duration). In addition, general-purpose processor descriptions can be downloaded into FPGAs (e.g. Intel, ARM, PowerPC ...) to run common OSes and their software applications. As a result, it has been shown that switching from software to hardware versions for some tasks greatly improves both performance and energy savings [6], two parameters that are of first importance in embedded devices. Fig.1 depicts a typical reconfigurable FPGA architecture that switches between software and hardware versions of tasks.

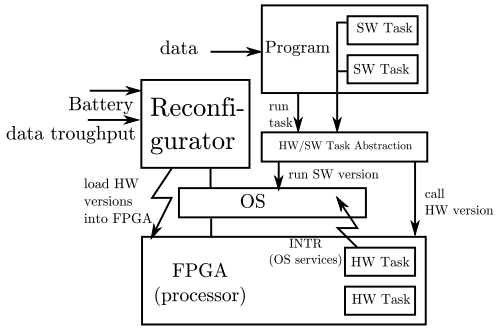


Fig. 1. Reconfigurable FPGA architecture with SW/HW task versions

B. Reconfiguration in SOC

Service Oriented Computing [7] is the computing paradigm in which applications *consume* services. The traditional approach wants applications to provide services. SOC goes beyond by allowing application to be built themselves using a *composition* of services. *Services* are provided through *software components* located in foreign repositories. When a service is requested, the application queries repositories in order to find the suitable component that provides it.

Reconfiguration in SOC involves replacing a component that is no longer able to deliver the service, for instance because of network failures (unreachable service). The main advantage of SOC is composition of services. Atomic services can be plugged together in order to provide high-level, more complex services. For instance, a high-level service that provides weather reports can be built using a service that provides standard information (temperature, wind), a service that provides satellite images and a service that shows live images from the desired location. Reconfiguration of a failed service consists in replacing it with a *similar* service. Similar services are generally *interface*-compatible and must be *function*-compatible. If interfaces are easily described using Description Languages (IDL, RDF, ...), functions can be described by adding tags or ontologies and performing *semantic* reasoning, thus the emergence of the *semantic web*. Semantic reasoning not only allows reconfiguration by replacing a failed service with a similar one [8], [9], but also allows replacing a single service by a composition of several other services [10].

III. Reconfiguration in ATS

A. Assistive Technology Systems

The changing distribution of age groups in our populations leads to more and more elderly people and disabled persons in need of some *rehabilitation technology* to maintain their mobility, autonomy and quality of life. *Assistive technology systems* (ATS) are a kind of pervasive systems that help dependant people improve their lives. By using some kind of *automation technology* to act on their environment, they can perform daily tasks they are no longer able to do. The Smart Home concept [3] (or Intelligent Environment) emphasizes on *environmental control* by incorporating electronically controllable devices and sensors to provide automated services and monitoring. *Active* (or on-demand) environmental control is achieved through the use of some automation technology and *passive* monitoring uses sensors to increase user's safety by triggering appropriate actions when harmful situations are detected (e.g. fall-off). Smart homes have become an active research field, we can cite [11], [12], [13] as few examples.

B. Reconfiguration in ATS

Reconfiguration has not yet widely gained Smart Homes and previous works either focused on navigation and addressed path planning problems in robotics, or focused on service provision and addressed service availability in computer science. We can also cite [14] in which a system for disabled people that combines wheelchair navigation and service provision has been built. Main drawback of this system is the lack of a service model and thus service reconfiguration consist in finding alternative paths or devices (and thus new paths for the found devices) and not reconfiguring services by themselves. This kind of reconfiguration is heavy since the user on its wheelchair will suffer from path change each time a service fails.

Regarding services, ATS can inherit many aspects of services in SOC, except that in ATS services are provided by independent physical devices present in smart homes (lamps, TVs, ...) and therefore cannot be composed in the same way as in SOC. Moreover, service failures in SOC are addressed by searching similar components, which implies the availability of these components. In smart homes, this is quite unfeasible since services are tightly related to physical devices and it is quite rare to have device redundancy. It is therefore essential to first suggest a service model which is compatible with the smart home concept, that establishes relationships between the hardware and the software, describe what is a service and how services can be composed together, then propose service reconfiguration schemes in addition to path reconfiguration when necessary.

The following section details the DANAH service model and the reconfiguration process to address service availability in smart homes.

IV. Service Reconfiguration in DANAH

A. Service Model

In contrast to SOC, ATS not only deal with software but also with the controlled hardware. A software part is responsible of advertising, organizing and managing the services that are run by the hardware. If a service fails, it is most likely that the cause comes from the underlying device. Reconfiguration does not involve querying a repository for similar services as in SOC because the architecture of a smart home is more local. Therefore there is tight relation between the device and the service. As a consequence, we defined the *resource* as the representation of a controlled device. Resources provide *operations* which are the basic services of our service model. Examples of resources are Lamps and Doors. 'On' and 'Off' (resp. 'Open' and 'Close') are the operations of Lamps (resp. Doors). The Fig.2 depicts the contents of a resource.

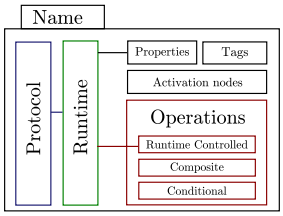


Fig. 2. Resource contents

To be advertised to the user, each resource has a *name*. A *runtime* is responsible for managing the internal status of the resource stored in *properties*, especially monitoring and sending orders to the controlled physical device using some home automation *protocol*.

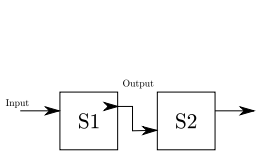
Simple operations are the basic services implemented by a resource that can be *composed* using *operators* to build *composite* operations. Simple operations are those that can be run through the runtime by sending orders directly to the device. Composite operations are built by composing other simple or composite operations to form scenarios. Operators tell in which order these operations are run, as shown in Fig.3.

```
resource "room" {
   operation "Exit" = "SEQ(MainLight.Off Door.Open)" }
```

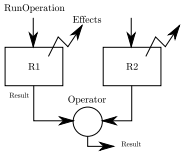
Fig. 3. A composite operation 'Exit' using the sequence operator that turns the lights off *then* opens the door room

In SOC, components require input data and produce output data. In DANAH, running operations does not require any input data and does not produce output other that the success or failure of the operation. As our service model follows a compositional approach, the rule applies also to composite operations, in this case, it is the operator that returns the result depending on its semantics and the result of the individual operations it composes, as shown in Fig.4.

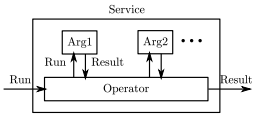
We can notice that when a user requests an operation, it is mostly interested in its impact on the surrounding environment, which we call *effects* [15]. Reconfiguring a failed service thus consists in finding other services that when composed together can result in same or similar effects.



(a) Service composition in SOC



(b) Service composition in DANAH



(c) Compositional approach in DANAH

Fig. 4. Composition schemes in SOC and in DANAH

B. Configurations

As stated above, the resource properties are used to store its current status. For instance, a door may contain a property 'Status' which can be set to 'Open' or 'Closed', and a heater may contain a property 'Temperature' which holds the current set temperature.

The environment is the set of resources. The impact of resource operations on the environment - their effects - is defined by setting up a 'has effect on' relationship between resources. A resource A is said to have effects on a resource B iff running operations of A may lead changes in the properties of B. For instance, if 'Lamp' and 'Room' are two resources, running the operation `Lamp.On` changes the room illuminance, written as `Room@Illuminance`.

The system *configuration* is defined as the set of all resource properties, and is written S^n at instant n. After running an operation that has effects on the environment, the system configuration becomes S^{n+1}, as shown in Fig.5.

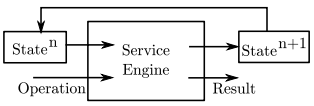


Fig. 5. System Configuration Transition

The new configuration S^{n+1} is computed in a two step procedure. After running the operation, the resource runtime is responsible of updating the resource's own status (that said, a resource has implicitly effects on itself). Then, a set of *effect rules* is used to evaluate the impact of this change

on other resources. An effect rule is described in terms of a pre-condition that if satisfied triggers its corresponding post-condition, as shown in Fig.6. A detailed view of this mechanism is shown in Fig.7.

```
effect { pre="MainLight@Status==On"
         post="Room@Illuminance=100" }
```

Fig. 6. An effect rule

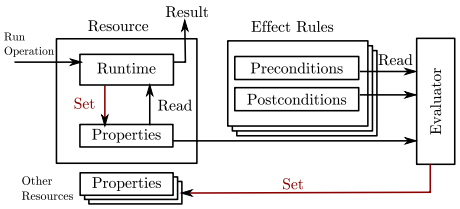


Fig. 7. In-depth view of system configuration computation

C. Service Execution

A service in DANAH can be

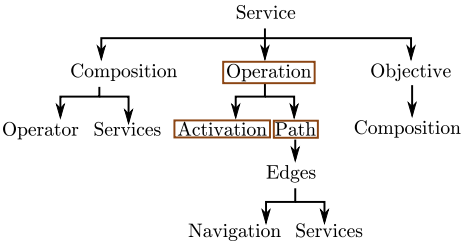
- an operation. e.g. `Door.Open`
- an objective. e.g. `Room@Temperature=20`
- a composition. e.g. `SEQ(Light.On Room@Temperature=20)`

When a service is requested, simple operations are run using resource runtimes, while composite ones are first expanded and run according to operator semantics. Objectives are *resolved* to a composition of operations and run. As we combine both navigation and environmental control, resources contain *activation nodes* which specify at which places the user must be in order to deliver operations. When a simple operation is run, navigation is performed by finding a path to an activation node then *activating* the operation, as shown in Fig.8. A path is a set of edges and each edge may contain *navigational attributes* [16] which store services that must be triggered while crossing. For instance, it may be necessary when performing navigation to open/close doors, enable/disable alarms, ...

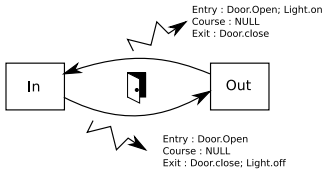
D. Multi-Level Service Reconfiguration

Reconfiguration in DANAH is performed on operations, not on the whole service. This is motivated by the fact that services can be composed of several operations involving resources that have nothing to do with each other. For instance, trying to reconfigure a service that switches on lights and opens a door as a whole would have no meaning. In contrast, operations can be reconfigured by collecting their effects and finding a set of other operation that would produce similar effects. Following the execution steps in Fig.8, the TABLEI shows for each step the possible failures and reconfiguration schemes. Reconfiguration tries always to recover from the failed step in order to avoid fault propagation to the upper levels.

In this paper we deal only with *activation failures*. After reaching the resource by performing navigation to one of its activation nodes, the requested operation is run, we call this step the *activation* of the operation. Resource runtime is responsible of activating the desired operation and reporting a result, which may be success or failure.



(a) Service decomposition



(b) Edges triggering services

Fig. 8. Decomposition of the execution of a service

TABLE I
ENUMERATION OF FAULTS, POSSIBLE CAUSES AND RECOVERING METHODS

Step	Cause	Reconfiguration
Navigation	Obstacle/ Wheelchair malfunction	(Not addressed)
Edge	Navigation/ Service	(Cause dependant)
Path	Edge	Path reconfiguration
Activation	Resource Malfunction	Static/Effect-Based reconfiguration
Operation	Path/ Activation	Resource Reconfiguration
Service	Operation	(Depends on operator semantics)

Activation reconfiguration is triggered on activation failure and aims at finding a set of similar services that will *compensate* the failed operation. DANAH benefits both from off-line reconfiguration using *static rules* and from on-line reconfiguration using *effect compensation*. When performing activation reconfiguration, statically defined alternatives are sought before switching to a dynamically computed alternative.

E. Level 1 : Static Reconfiguration

Static reconfiguration is triggered on activation failure. It performs compensation on an operation basis, using off-line defined reconfiguration rules in resource descriptions.

Each operation can have a static reconfiguration rule which maps it to an alternative service to run in case of failure, as shown in Fig.9.

In this example, we suppose a room equipped with a main light and two alternative lights. This rule specifies if the

```
reconf { fail="MainLight.On"
         reconf="SEQ(AltLight1.On AltLight2.On)" }
```

Fig. 9. A static reconfiguration rule

operation 'On' of the main light fails, both alternative lights can be switched on to compensate.

Static reconfiguration rules are provided as a convenient way of defining alternatives to failed operations. In general, they are written by the expert who installs the system for a particular user. This is most of interest in case of the user is a disabled person.

F. Level 2 : Effect-based Reconfiguration

Effect-based reconfiguration is triggered on activation failure, in the absence of static reconfiguration rules and if the failed operation has effects. It is based on effect compensation and is computed dynamically.

When an activation fails, the effects it would have produced on other resources are collected. Then for each effect, a service that will produce the same or a similar effect is dynamically computed.

More formally, let us suppose the effect $(p, v^{n+1}) \in S^{n+1}$, where p is the property name and v^{n+1} its value after running the operation successfully. Given the current system status S^n which contains (p, v^n), effect reconfiguration searches for a service that if run will bring the value of property p from v^n to v^{n+1}. The point here is that if the operation fails, S^{n+1} is not known because the failed operation has not produced any effects. In our resource model, all resources have a function $EffectOf$ that tells the impact of running their operations on their properties if they are successful. If an operation fails, this function in conjunction with effect rules are used to compute the would be state S^{n+1}, then reconfiguration is performed starting from the current state S^n.

The way the alternative service is searched is through using a 'dry run' mechanism. Given an effect (p, v^{n+1}) to reach, all the resources that affect the property p are searched. For each of these resources, operations activations are faked by calling the $EffectOf$ function and the new system state is computed. The operation that leads to the closest expected value v^{n+1} of p is kept and appended to the list of a composition with a `SEQ` operator. This process is restarted until no operations can lead to a closer value or when the desired value is reached. Finally, the computed sequence is returned as the alternative to the failed operation.

Let us take an example. Suppose we have a resource Heater with two operations '+' and '-'. The current room temperature is 20 and we want a temperature of 22. DANAH searches for resources that affect the temperature property of room and finds the heater. It then starts to dry runs its operations. Running '-' leads to 19 while running '+' leads to 21. It keeps '+' and starts again, with the current value of temperature 21. Operation '-' leads to '20' and '+' leads to '22', the desired value. The final compensation service is then `SEQ(Heater.+ Heater.+)`. As we can notice, not only a composition of several operations has been computed to compensate one single operation, but the compensation is dynamic and takes care of the current system status as well, as it may have been different in other situations.

G. Other reconfiguration levels

When activation definitely fails (e.g. no reconfiguration found), other reconfiguration levels to deliver the service can be triggered. WHen it is a part of a service triggered by a navigational attribute, *path reconfiguration* is performed. This level aims at finding another path to reach the resource to activate the desired operation. This leads to a change in path without changing the initial goal (the operation), and implies a trajectory change for the user. If even the path reconfiguration fails, or the activation is not part of a navigational attribute, another last reconfiguration level called *resource compensation* is triggered. This time the initial resource changes because either it cannot be reached (path reconfiguration failed) of it is out of order (activation definitely failed). Resource compensation aims at finding similar resources on a resource basis using tags found inside resource descriptions. As a change in goal may also imply a new path computation, the list of similar resources is presented to the user in order to always let him keep control on the system and take decisions by himself.

V. Experimental Platform

The reconfiguration levels described in this work have been fully implemented in our system DANAH[4]. DANAH is an ATS the combines both environmental control and navigation, with reconfiguration in mind. It is meant to be deployed quickly in smart homes. Special attention has been focused on cost effectiveness, portability, modularity and installation methodology. A model-driven approach is used in which the installer uses a general purpose drawing program to describe the target environment, then model transformations generate clean structured data for DANAH. Regarding modularity, resource runtimes, protocols and robot drivers for autonomous navigation are implemented as plugins and loaded at startup. Cost effectiveness is achieved through the use of open source software that is portable and requires low performance machines.

The DANAH architecture described in Fig.10 consist of DANAH servers deployed in the smart home, DANAH clients embedded on user devices (PDAs, small computers, ...) and a home automation technology that bridges the software part (the DANAH system) and the physical devices. Thanks to its modularity, DANAH has plugins for the KNX/EIB open standard technology and Infrared, this way it can act on simple electrical devices (switches, doors, ...) and multimedia devices (TV, Radio, ...). Communication between clients and servers is achieved through protocol plugins. Currently, DANAH has both support for WiFi and Bluetooth technologies. It runs

under Linux and has been ported to the NOKIA N800 handheld device.

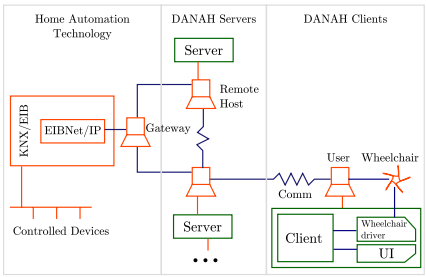


Fig. 10. DANAH architecture and HW/SW mapping

Client user interface shown in Fig.12 is especially designed for disabled people and uses an icon paradigm to display resources and operations. Thanks to effect rules (see Fig.11), resource icon is changed depending on resource status to provide the user with operation execution feedback. Moreover, user interface provides an automatic scrolling mode to help disabled people use it when they cannot perform clicks on the touchscreen.

```
effect { pre="this@Status==On"
         post="this@icon='light-on.png'" }
```

Fig. 11. Icon feedback setup using effect rules. The 'this' special resource name denotes the resource that contains the rule

Fig. 12. DANAH Client GUI connected to a server displaying the resource page

VI. Conclusion

This paper described a multi-level service reconfiguration approach to ensure service availability in domotized living spaces. Static reconfiguration can ensure availability by picking up alternative services from an off-line pool, while effect-based reconfiguration dynamically computes alternatives at runtime using the effect paradigm and effect rules. Taking into account navigation leads to perform path reconfiguration when necessary. Finally, resource compensation is used when the service cannot be delivered using the initial resource.

These reconfiguration schemes have been implemented in the DANAH assistive system and experimentation is in progress in a medical structure involving patients with different pathologies. Evaluation will compare time and perceived quality of service when reconfiguration is enabled or disabled, and will show for each pathology the suitable quantity of reconfiguration.

References

[1] J. C. Augusto and P. J. McCullagh, "Ambient intelligence: Concepts and applications," *Comput. Sci. Inf. Syst*, vol. 4, no. 1, p. 1–27, 2007.

[2] M. Satyanarayanan, "Pervasive computing: vision and challenges," *IEEE Personal Communications*, vol. 8, no. 4, p. 10–17, Aug. 2001.

[3] V. Ricquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge, "The smart home concept : our immediate future," in *Proc. IEEE International Conference on E-Learning in Industrial Electronics*, Dec. 2006, p. 23–28.

[4] S. Lankri, P. Berruet, A. Rossi, and J.-L. Philippe, "Architecture and models of the DANAH assistive system," in *SIPE '08: Proceedings of the 3rd international workshop on Services integration in pervasive environments*. New York, NY, USA: ACM, 2008, p. 19–24.

[5] T. Todman, G. Constantinides, S. Wilton, P. Cheung, W. Luk, and O. Mencer, "Reconfigurable Computing: Architectures and Design Methods," vol. 152, no. 2, p. 193–205, Mar. 2005. [Online]. Available: http://pubs.doc.ic.ac.uk/reconfigurable-computing/

[6] G. Stitt, F. Vahid, and S. Nematbakhsh, "Energy savings and speedups from partitioning critical software loops to hardware in embedded systems," *Trans. on Embedded Computing Sys.*, vol. 3, no. 1, p. 218–232, 2004.

[7] M. N. Huhns and M. P. Singh, "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, no. 1, p. 75–81, 2005.

[8] J. Kim, J. Lee, and B. Lee, "Runtime service discovery and reconfiguration using OWL-S based semantic web service," in *CIT*. IEEE Computer Society, 2007, p. 891–896.

[9] H. Hemmati, M. Niamanesh, and R. Jalili, "A framework to support run-time assured dynamic reconfiguration for pervasive computing environments," *IEEE 1st International Symposium on Wireless Pervasive Computing*, p. 6 pp.–, Jan. 2006.

[10] M. Vallée, F. Ramparany, and L. Vercouter, "Composition flexible de services d'objets communicants," in *2nd French-speaking conference on Mobility and Ubiquity computing*. ACM, 2005, p. 185–192.

[11] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The gator tech smart house: A programmable pervasive space," *Computer*, vol. 38, no. 3, 2005.

[12] MIT, "The MIT house_n project." [Online]. Available: http://architecture.mit.edu/house_n

[13] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, and W. Newstetter, "The aware home: A living laboratory for ubiquitous computing research," in *Cooperative Buildings, Integrating Information, Organization, and Architecture, Second International Workshop, CoBuild'99, Pittsburgh, USA, October 1-2, 1999, Proceedings*, ser. Lecture Notes in Computer Science, N. A. Streitz, J. Siegel, V. Hartkopf, and S. Konomi, Eds., vol. 1670. Springer, 1999, p. 191–198.

[14] A. Belabbas, P. Berruet, A. Rossi, and J. Philippe, "A modeling approach to control a handicap technical assistance system," in *WSEAS Transactions on Information Science and Applications*, 2006.

[15] A. Urbieta, E. Azketa, I. Gomez, J. Parra, and N. Arana, "Towards effects-based service description and integration in pervasive environments," in *SIPE '08: Proceedings of the 3rd international workshop on Services integration in pervasive environments*. New York, NY, USA: ACM, 2008, p. 1–6.

[16] B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T. Mossakowski, and R. Ross, "Specification of an ontology for route graphs," in *Proc. International Conference on Spatial Cognition*, Oct. 2004.