A Norm-based Approach for Personalising Smart Environments

Patrizia Ribino, Carmelo Lodato, Antonella Cavaleri, and Massimo Cossentino

Istituto di Calcolo e Reti ad alte Prestazioni
Consiglio Nazionale delle Ricerche - Italy
{ribino,c.lodato,a.cavaleri,cossentino}@pa.icar.cnr.it
http://ecos.pa.icar.cnr.it

Abstract. People have a great variety in their needs. There is a great demand for personalized services, especially those interacting with their environment. In this paper, we propose a norm based approach for personalizing smart environments that constraint user requirements by means of non functional requirements expressed in terms of permissions, obligations or prohibitions.

Keywords: Smart systems, norms, smart environment

1 Introduction

In recent years, a growing trend is the development of smart systems to improve well-being of individuals in their environment by making everyday activities more convenient and enjoyable. Smart systems aim at augmenting real environments to create smart spaces where users are provided with pervasive electronic devices. Usually each device can provide a set of services and functionalities. A smart system connects such electronic devices into a network and control them by using advanced ICT technologies in such away the devices satisfy user requirements. A common architecture for smart environment is sketched in Figure 1(a). The highest layer is related to the interaction with users. The second layer provides intelligence of the environment, often using artificial intelligence capabilities. Finally, the lowest layer is related to the electronic devices based on current technologies used for smart systems.

The use of a smart environment is variable from a user to another. Designing ad-hoc systems for each individual is not realistic economically. As well as, it is not possible to hard-wire all possible user scenarios. Hence, how to personalise smart environments? The most reasonable answer is that users have to be able to define their own requirements, thus defining their own usage scenarios. There are several kinds of systems that try to address this question [8]:

- *Predefined Scenario Systems* include centralized systems based on predefined scenarios. They only provide the users the possibility to choose the scenarios they want to execute. A new scenario definition generally consists in assembling existing components [3, 6].

- *Service Control Systems* include systems that allow users to control available services by providing automatic detection of devices in their environment. A user may interact with the system for triggering service executions, but he cannot define complex scenarios [9, 14].
- *Scenario Definition Systems* enable users to define their own scenarios. Only some of them allow runtime scenario definition and in some cases such scenarios are sequences of service calls [8].

Such kind of systems do not allow to manage non functional requirements[1] during a predefined scenario execution. In this paper, we propose a norm based approach for the definition of highly personalised scenarios during the normal execution of the system. The proposed approach allows for constraining user requirements by means of non functional requirements (i.e: variable user personalizations) expressed in terms of norms (i.e: permissions, obligations or prohibitions). In order to give the users the capability to define their own initial requirements and to modify the behaviour of the system during its normal execution according to user personalizations, we introduced a Normative Layer in the classical smart environment architecture (see Figure 1(a)).

The main contribution of this paper is an algorithm implemented in the Normative Layer, which integrates non functional requirements expressed by means of norms (i.e: permissions, obligations and prohibitions) into user requirements (expressed in terms of goals) in order to introduce personalizations into predefined user scenarios.

In order to show some practical examples, in this paper, we use GoalSpec [13] and SBVR [7] for modelling user requirements and personalizations in the semantic layer and MUSA [4] for the composition and the orchestration of services in the Execution Layer. GoalSpec is a language designed for specifying user-goals and enabling at the same time goal injection and software agent reasoning. The Semantics of Business Vocabulary and Business Rules (SBVR) is an adopted standard of the Object Management Group (OMG). It is designed for formalizing complex business rules. MUSA (Middleware for User-driven Service Adaptation) is a holonic multi-agent system for the composition and the orchestration of services in a distributed and open environment that is founded on the BDI paradigm[11].

The remaining of the paper is organized as follows. Section 2 presents the proposed norm-based approach along with some definitions is based on. Section 3 shows some application scenarios. Some related works are illustrated in Section 4. Finally, in Section 5 some conclusions are drawn.

2 The Normative Layer

Many studies have been conducted in order to develop systems that satisfy user requirements in smart environments. These systems generally provide predefined

[1] For the scope of this paper, we consider only user constraints. Thus, hereafter we use non functional requirements, user constraints or user personalization as synonymous.

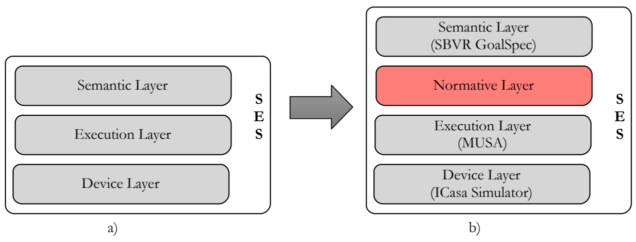


Fig. 1: The proposed Layered Smart Environment System

scenarios corresponding to general requirements and enable users to select those he/she wants to trigger. Such hard-wired behaviours limit the personalization degree of such systems. The main purpose of this work is to introduce more flexibility in the scenario definition for personalising smart environments. In order to reach this aim, we conceived a Normative Layer independent of the Execution Layer (see Figure 1(b)) and we propose a norm based approach in order to introduce personalizations in such smart environment systems.

The proposed approach is based on three key concepts: state of the world, goal and norm.

Definition 1 (State of the world).

Let $\mathcal{D}$ the set of concepts defining a business domain. Let $\mathcal{L}$ be a first-order logic defined on $\mathcal{D}$ with $\top$ a tautology and $\perp$ a logical contradiction, where an atomic formula $p(t_1, t_2..., t_n) \in \mathcal{L}$ is represented by a predicate applied to a tuple of terms $(t_1, t_2..., t_n) \in \mathcal{D}$ and the predicate is a property of or relation between such terms that can be true or false.

A *state of the world* in a given time t ($\mathcal{W}^t$) is a subset of atomic formulae whose values are true at the time t:

$$\mathcal{W}^t = [p_1(t_1, t_2, ..., t_h), ..., p_n(t_1, t_2, ..., t_m)]$$

The *state of the world* represents a set of declarative information concerning events occurred within the environment and relations among events at a specific time. An event can be defined as the occurrence of some fact that can be perceived by or be communicated to the smart system. Events can be used to represent any information that can characterize the situation of an interacting user as well as a set of circumstances in which the smart system operates at a specific time. Definition 1 is based on the close world hypothesis that assumes all facts that are not in the state of the world are considered false.

Definition 2 (Goal).

Let $\mathcal{D}$, $\mathcal{L}$ and $p(t_1, t_2..., t_n) \in \mathcal{L}$ as previously introduced in the definition 1. Let $t_c \in \mathcal{L}$ and $f_s \in \mathcal{L}$ formulae that may be composed of atomic formulae by means of logic connectives AND($\wedge$), OR ($\vee$) and NOT ($\neg$).

A *Goal* is a pair $\langle t_c, f_s \rangle$ where t_c (*trigger condition*) is a condition to evaluate over a state of the world $\mathcal{W}^t$ when the goal may be actively pursued and f_s (*final* *state*) is a condition to evaluate over a state of the world $W^{t+\Delta t}$ when it is eventually addressed:

- *a goal is active iff* $t_c(\mathcal{W}^t) \wedge \neg f_s(\mathcal{W}^t) = true$
- *a goal is addressed iff* $f_s(\mathcal{W}^{t+\Delta t}) = true$

Goals express what is the desired state of the world the system has to result in. Conversely, norms denote the way the system has to operate in order to achieve the desired state of the world in compliance with the normative context in which that process takes place.

Definition 3 (Norm).
Let $\mathcal{D}$, $\mathcal{L}$ and $p(t_1, t_2..., t_n)\in\mathcal{L}$ and let a *state of the world* in a given time t ($\mathcal{W}^t$) as previously introduced in the definition 1. Let $\phi\in\mathcal{L}$ and $\rho\in\mathcal{L}$ formulae composed of atomic formula by means of logic connectives AND($\wedge$), OR ($\vee$) and NOT ($\neg$). Moreover, let $D_{op} = \{permission, obligation, prohibition\}$ the set of deontic operators. A *Norm* is defined by the elements of the following tuple:

$$n = \langle r, g, \rho, \phi, d\rangle$$

where

- $r\in\mathcal{R}$ is the *Role* the norm refers to. The special character "_" indicates that the norm refers any role.
- $g\in\mathcal{G}$ is the *Goal* the norm refers to. The special character "_" indicates that the norm refers to any goal.
- $\rho\in\mathcal{L}$ is a formula expressing the set of actions and state of affairs that the norm disciplines.
- $\phi\in\mathcal{L}$ is a logic condition (to evaluate over a state of the world $\mathcal{W}^t$) under which the norm is applicable;
- $d\in D_{op}$ is the deontic operator applied to ρ that the norm prescribes to the couple $(r, g)\in\mathcal{R} \times \mathcal{G}$.
  In particular $d(\rho) = \begin{cases} \rho & \text{iff } d = \textit{obligation} \\ \neg\rho, & \text{iff } d = \textit{prohibition} \\ \rho \vee \neg\rho & \text{iff } d = \textit{permission} \end{cases}$

In other words, let a state of the world $\mathcal{W}^t$ a norm prescribes to a couple (r, g) the deontic operator d applied to ρ if ϕ is true in $\mathcal{W}^t$ [2]. *Norms* represent system regulations by specifying obligations, permissions or prohibitions to be followed during system activities in case of certain conditions occur, thus relaxing or restricting a process.

Definition 4 (State of Norm). Let a norm $n = \langle r, g, \rho, \phi, d\rangle$ where $g = \langle t_c, f_s\rangle$ and let a state of the world in a given time t ($\mathcal{W}^t$)
A norm can assume the following states:

[2] It is worth noting that in order to be compliant in $\mathcal{W}^t$ with 1) an obligation ρ must be true, 2) a prohibition $\neg\rho$ must be true 3) a permission ρ or $\neg\rho$ may be true. In the context of this paper, we assume that the system does not violate norms.

- n is *applicable at time t* if $\phi(\mathcal{W}^t) = true \vee \phi = \top$
- n is *active at time t* if n is applicable and $t_c(\mathcal{W}^t) = true$
- n is *logically contradictory* if ϕ *is* $\bot$
- n is *in opposition to goal* if $f_s \wedge d(\rho)$ *is* $\bot$

Moreover, let a state of the world (W^t) and let two norms $n_1 = \langle r_1, g_1, \rho_1, \phi_1, d_1\rangle$ and $n_2 = \langle r_2, g_2, \rho_2, \phi_2, d_2\rangle$ where $r_1 = r_2$, $g_1 = g_2$, $\rho_1 = \rho_2$

- n_1 and n_2 are *deontically contradictory* iff $\begin{cases} \phi_1(\mathcal{W}^t) \wedge \phi_2(\mathcal{W}^t) = true \\ d_1 \neq d_2 \end{cases}$

It is worth noting that we talk about *logically contradictory* when the contradiction concerns the logical conditions ($\phi \in \mathcal{L}$) under which the norms are applicable. On the contrary, we talk about *deontically contradictory* when the contradiction concerns the semantic meaning of the deontic operator ($d \in D_{op}$) the norms apply.

Algorithms: The aim of the Normative Layer is to provide some mechanisms that allow to modify the behaviour of the smart environment (that is based on some predefined user scenarios) in order to adapt it to user personalization expressed by means of SBVR Rules in the Semantic Layer[3].

Algorithm 1: Norms into Requirements

```
Data: a set of goals $\mathcal{G}$ and a set of norms $\mathcal{N}$
Result: $\mathcal{G}_{\mathcal{N}}$
$\mathcal{G}_{\mathcal{N}} \leftarrow \varnothing$;
create a list $NormList, size(NormList) = card(\mathcal{N})$;
create a list $GoalList, size(GoalList) = card(G)$;
// Loop for detecting logically contradictory norms
① for $j \leftarrow 1$ to $card(\mathcal{N})$ do
    $\langle r, g, \rho, \phi, d\rangle \leftarrow n_j$;
    if $n_j$ is not logically contradictory then
        add $\langle r, g, \rho, \phi, d\rangle$ to $NormList$;
$GoalList \leftarrow G$;
// Loop for encapsulating norms into goals
② for $i \leftarrow 1$ to $size(GoalList)$ do
    // see Algorithm 2
    $(\phi_{mergedOR}, \phi_{mergedAND}) \leftarrow compose\_norm(GoalList[i], NormList)$;
    // Goal composition
    $\langle t_c, f_s\rangle \leftarrow GoalList[i]$;
    $t_c \leftarrow OR\_composition(t_c, \phi_{mergedOR})$;
    $t_c \leftarrow AND\_composition(t_c, \phi_{mergedAND})$;
    add $\langle t_c, f_s\rangle$ to $\mathcal{G}_{\mathcal{N}}$;
```

[3] It is out of the scope of the paper the algorithms that convert SBVR and GoalSpec in the formalisms managed by the Normative Layer

Algorithm 1 is the core of the Normative Layer. It allows to modify goals, making them norm compliant. By encapsulating the condition expressed by the norms inside the goal they refer, it is possible to modify the activation of that goal thus making it compliant with the norms. Algorithm 1 consists of an initial pre-filtering (Step ①) of logically contradictory norms (see Definition 4). Then norms are encapsulated into goals (see Step ②) by composing new trigger conditions for goals from the norm conditions (see Algorithm 2).

Algorithm 2: Compose_Norm

```
Data: a goal $g_{current}$, a list of norms $NormList$
Result: a couple $(\phi_{mergedOR}, \phi_{mergedAND})$
$List\phi\_OR \leftarrow \varnothing$;
$List\phi\_AND \leftarrow \varnothing$;
// Identification of norm types
①for $j \leftarrow 1$ to $size(NormList)$ do
    $\langle r, g, \rho, \phi, d \rangle \leftarrow NormList[j]$;
    $\langle t_c, f_s \rangle \leftarrow g$;
    // Choose among norms of the current goal, which are directly
       linked to the goal final state and which are not in opposition
       to the same goal
    if $(g = g_{current}) \wedge (f_s = \rho) \wedge ((f_s \wedge d(\rho)) \neq \perp)$ then
        switch $d$ do
            case Obligation
                break;
            case Prohibition
                add $\neg\phi$ to $List\phi\_AND$;
            case Permission
                add $\phi$ to $List\phi\_OR$;
// Permissions give alternatives (OR)
②if $Size(List\phi\_OR) \neq 0$ then
    $\phi_{mergedOR} \leftarrow List\phi\_OR[1]$;
    for $h \leftarrow 2$ to $Size(List\phi\_OR)$ do
        $\phi_{mergedOR} \leftarrow OR\_composition(\phi_{mergedOR}, List\phi\_OR[h])$;
// Prohibition are mandatory (AND)
③if $Size(List\phi\_AND) \neq 0$ then
    $\phi_{mergedAND} \leftarrow List\phi\_AND[1]$;
    for $h \leftarrow 2$ to $Size(List\phi\_AND)$ do
        $\phi_{mergedAND} \leftarrow AND\_composition(\phi_{mergedAND}, List\phi\_AND[h])$;
```

Such composition takes into consideration different types of norm and addresses the following question: when norms regulate a goal, in what cases that goal is activated? The activation table shown in Figure 2 shows all the possible cases in which the system can pursue a goal: i) when the trigger condition is true and the norm is not applicable or ii) when the norm is applicable and its deontic operator is permission; iii) when the norm is active and its deontic operator is an obligation. As we can see in Figure 2, when an obligation is introduced in the

$n = \langle \_, g, \rho, \varphi, d \rangle \quad g = \langle t_c, f_s \rangle$

A) Permission

$t_c \lor \varphi$	t_c	φ
1	1	0
1	0	1
0	0	0
1	1	1

B) Obligation

$t_c \lor (\varphi \land t_c)$	t_c	φ
1	1	0
0	0	1
0	0	0
1	1	1

C) Prohibition

$t_c \land \neg\varphi$	t_c	φ
1	1	0
0	0	1
0	0	0
0	1	1

Fig. 2: Goal activation regulated by norms.

system, it does no effective change to the original activation of the goal. Because in any case the original trigger condition has to be satisfied.

In common smart environments all the parameters that trigger the execution of some scenarios have to be defined a priori for each of them. Our approach allows to define prohibitions or permissions (that express conditions under which something is allowed or forbidden) without a specific relation with a scenario. In our approach norms are transversal to several scenarios, thus decoupling what we want the system do and the boundary in which it has to move.

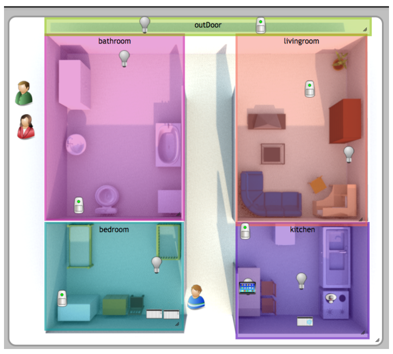
In the following, we show some simple application scenarios.

3 Application scenarios

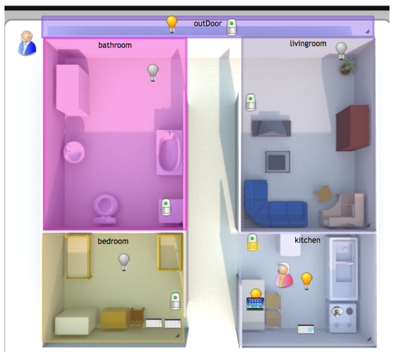
In this section we provide some examples of application scenarios. We simulate such scenarios in ICasa simulator [1]. The purpose is to show as the same smart environment behaves according to different user personalizations. Let us suppose to install the same smart home system in two different apartments. Thus, each apartment is furnished with sensors for the detection of individuals, with digital cooking stove, air conditioner, electronic shutters and so on. The first apartment is occupied by two adults, Sara and John, and a teenager, Luke. Sara and John establish some norms they want their smart apartment follows. In particular, they forbid the system to activate any cooking device if there are no adults at home. Thus, supposing the system owns an appropriate knowledge, such norm could be expressed in SBVR as follows:

N1: *It is prohibited turning on the cooking device if Sara is not at home or John is not at home.*

In the second apartment live Mike and his wife. Mike forbids the system to turn off the exterior lights of the apartment during the night if the house is empty or if only one person is at home. The corresponding norm could be:



(a) Apartment 1 without norm



(b) Apartment 2 with active norm

Fig. 3: Good Night Scenario

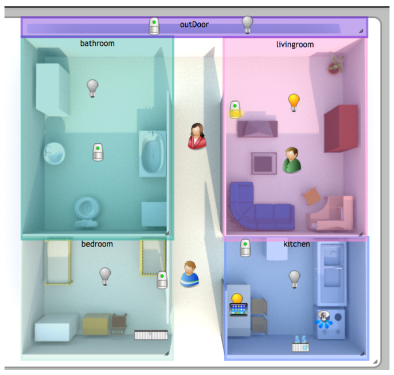
N2: *It is prohibited turning off exterior lights if it is night and house habitant are not at home.*

Let us suppose the smart system has two predefined scenarios: *good night* and *have a lunch*. In the *good night* scenario, the exterior lights of home turn off and the curtains and shutters of the bedroom will be automatically closed. In the *have a lunch* scenario, the TV switches on, the conditioner turns on at the desired temperature and the cooking stove turns on if there is a pot on. Such scenarios could be expressed in GoalSpec as follows:

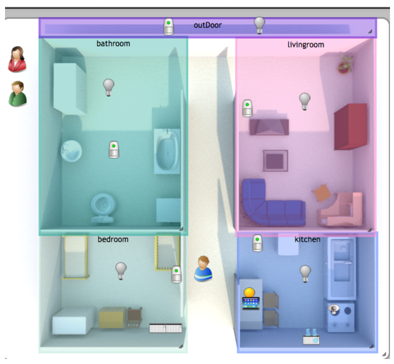
Good night:
WHEN on(10:00 pm) THE system SHALL ADDRESS closed(shutter) AND closed(curtains) AND exterior_lights(off).

Have a lunch:
WHEN on(1:00 pm) THE system SHALL ADDRESS television(on) AND conditioner(on, T)
WHEN on(1:00 pm) AND pot_on(cooking_stove) THE system SHALL ADDRESS cooking_stove(on)

Although in each apartment both the scenarios have been activated, the system behaves differently. In the first apartment the good night scenario will be executed as well as it is defined. In Figure 3(a), Sara and John are coming back at home later in the night. Luke is at home. The system at 10 o'clock closes shutters and curtains and turns off exterior lights. In the second apartment the *good night* scenario will be differently executed. In Figure 3(b), Mike went to walk after the dinner, his wife is at home. At 10 o'clock the system closes shutters and curtains. The system does not turn off exterior lights, until Mike comes back at home. Similarly, in the *have a lunch* scenario, if both families are at home the



(a) Apartment 1



(b) Apartment 1 with active norm

Fig. 4: Have a Lunch Scenario

system behaves in each apartment in the same way. TV and air conditioner are turned on and cooking device starts to cook the lunch (see Figure 4(a)). A day, Luke unusually comes back at home for lunch before of his parents. At 1 o'clock the TV and air conditioner turn on while cooking device continues to be off, although there is a pot on the stove (see Figure 4(b)). It is worth noting that norms defined by Mike and Sara are not directly related to *good night* or *have a lunch* scenario. They may also act in other scenarios that include the same actions the norms discipline.

These are simple examples we can simulate. But the proposed approach goes beyond the smart home systems. We can apply our approach also in other contexts. For example in smart traffic and transportation contexts, we can conceive a smart system that allows to automatically drive a vehicle. Such systems have to follow different traffic regulations, according to the country they will operate.

4 Related Works

Personalization is a key issue to be addressed in order to widely diffuse the use of smart environments. The approach we propose is quite simple, but at the same time effective. It works to a higher abstraction level by introducing norms for personalizing user requirements, thus not modifying the lower execution layer of a smart environment system.

To the best of our knowledge, there are no works that use norms for personalising smart environments. Such an issue is commonly addressed as scenario definition problems [3, 6, 9, 14, 8]. Such kind of systems do not allow to manage non functional requirements during a predefined scenario execution.

Other specific works cope with this problem by endowing the system with ad-hoc devices or specific system infrastructures. Only to cite a few, in [5] Giroux *et.al* present a multi-agent infrastructure where agents collaborate to personalize cognitive assistance by adapting their behaviour to the need of the people living in the smart home. In [10], Loseto *et.al* present a distributed multi-agent framework for home and building automation, based on a semantic enhancement of EIB/KNX domotic standard by exploiting knowledge representation and reasoning technologies. The proposed approach supports the semantic characterization of user profiles and device functionalities to discovery and orchestrate resources in HBA (Home and Building Automation). In [12], Russell *et.al* propose a general methodology that defines how to use unobtrusive sensors within smart environments to provide physical context and metadata for system personalization. In [2], Bergesio *et.al* propose a method for personalizing the behaviour of smart environments by means of the configuration of smart objects through mobile devices. With respect to the previous ones, our approach introduces more flexibility in the scenario definition by means of a Normative Layer that is independent from the Execution Layer.

5 Conclusions

The paper proposes an approach for personalising user requirements by expressing non functional requirements (i.e: variable user personalizations) by means of norms (i.e: permissions, obligations or prohibitions). The proposed norm-based approach is able to support the specification of the variable parts of a scenario execution in a smart environment. The approach we propose facilitates the customization of a smart environment to a particular usage context by decoupling the variable parts (i.e: user personalization) from standard user requirements. Then an algorithm implemented in the Normative Layer combines system requirements with user personalizations. Norms allow to specify user personalizations in a way that is understandable by the user, but also executable by the Execution Layer, thus bridging the gap between user and technology. At the moment we are working on the integration in the Normative Layer of an algorithm to allow runtime modification of the system. We are also developing a norm editor that allows to easily compose norms according to the domain the system will operate. The work proposed in this paper is part of a larger one that aims at creating normative smart environments that can be compliant with the legislative environments they are plugged in.

References

1. Icasa a dynamic pervasive environment simulator, http://adele.imag.fr/icasa-a-dynamic-pervasive-environment-simulator/.
2. Luca Bergesio, Igo Marquinez, Ana M Bernardos, Juan A Besada, and José R Casar. Perseo: A system to personalize the environment response through smart phones and objects. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 640–645. IEEE, 2013.

3. André Bottaro, Anne Gérodolle, and Philippe Lalanda. Pervasive service composition in the home network. In *Advanced Information Networking and Applications, 2007. AINA'07. 21st International Conference on*, pages 596–603. IEEE, 2007.
4. M Cossentino, C Lodato, S Lopes, and L Sabatucci. Musa: a middleware for user-driven service adaptation. *in proc. of XVI WORKSHOP "DAGLI OGGETTI AGLI AGENTI", Napoli, June, 17-19, 2015*, 1382, 2015.
5. Sylvain Giroux, Matthieu Castebrunet, Olivier Boissier, and Vincent Rialle. A multiagent approach to personalization and assistance to multiple persons in a smart home. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, pages pp–11, 2014.
6. Guillaume Grondin, Noury Bouraqadi, and Laurent Vercouter. Madcar: an abstract model for dynamic and automatic (re-) assembling of component-based applications. In *Component-Based Software Engineering*, pages 360–367. Springer, 2006.
7. Object Management Group. Semantics of business vocabulary and business rules (sbvr). version 1.3. may 2015.
8. Fady Hamoui, Marianne Huchard, Christelle Urtado, and Sylvain Vauttier. Specification of a component-based domotic system to support user-defined scenarios. In *SEKE 2009: 21st International Conference on Software Engineering and Knowledge Engineering*, pages 597–602. Knowledge Systems Institute Graduate School, 2009.
9. Hiroo Ishikawa, Yuuki Ogata, Kazuto Adachi, and Tatsuo Nakajima. Building smart appliance integration middleware on the osgi framework. In *Object-Oriented Real-Time Distributed Computing, 2004. Proceedings. Seventh IEEE International Symposium on*, pages 139–146. IEEE, 2004.
10. Giuseppe Loseto, Floriano Scioscia, Michele Ruta, and Eugenio Di Sciascio. Semantic-based smart homes: a multi-agent approach. In *13th Workshop on objects and agents (WOA 2012)*, volume 892, pages 49–55, 2012.
11. A.S. Rao and M.P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pages 312–319. San Francisco, 1995.
12. Luke Russell, Rafik Goubran, and Felix Kwamena. Personalization using sensors for preliminary human detection in an iot environment. In *Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on*, pages 236–241. IEEE, 2015.
13. Luca Sabatucci, Patrizia Ribino, Carmelo Lodato, Salvatore Lopes, and Massimo Cossentino. Goalspec: A goal specification language supporting adaptivity and evolution. In *Engineering Multi-Agent Systems*, pages 235–254. Springer, 2013.
14. Chao-Lin Wu, Chun-Feng Liao, and Li-Chen Fu. Service-oriented smart-home architecture based on osgi and mobile-agent technology. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(2):193–205, 2007.