Intelligent actuation in Home and Building Automation Systems

Rui Camacho[a,b,*]

[a]*Instituto Superior Técnico, Universidade de Lisboa*
Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal
{rui.camacho, paulo.carreira, silvia.resendes, ines.lynce}@tecnico.ulisboa.pt
[b]*INESC-ID Lisboa,*
Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal

Abstract

Ambient intelligent systems such as Home and Building Automation Systems (HBAS) are becoming evermore accepted and capable of actuating automatically on behalf of users to fulfil their requests or enable activities. However, when multiple users interact with such systems, the requirements of activities often interfere resulting in conflicting actuations which HBAS ought to automatically detect and resolve. Yet, despite recent advances in HBAS, no ambient intelligent solution has been reported that is adequately grounded on knowledge analysis.

The contributions of this article include a review on relevant literature on ambient intelligence, conflict detection, conflict resolution and knowledge representation in HBAS. It provides a practical approach to conflict detection and resolution backed by knowledge-based analysis. Effectively, the solution performs automatic environment actuations maximizing users comfort and energy efficiency.

Keywords: Home and Building Automation Systems, Ambient Intelligence, Conflict Detection, Conflict Resolution

1. Introduction

Home and Building Automation Systems (HBAS) are systems that control electronic devices and equipment inside home and buildings. Home Automation Systems (HAS) differ from Building Automation Systems (BAS) in the precincts that HAS focus on providing omnipresent and inobtrusive comfort to users. BAS, in contrast, aim at maximizing energetic efficiency towards economic benefits in large buildings, while ensuring an adequate level of service (Echelon, 2003; Siemens, 2012). A growing number of devices in our homes, from lamps (LIFX, 2012; Hue, 2013) to air conditioning units (Nest, 2012; Smart-Zone, 2012; Aros, 2014), although having a physical interface with buttons, are becoming increasingly autonomous and Internet-enabled (WeMo, 2013; Smartthings, 2012; Staples-Connect, 2013), thus forming a networked intelligent control system. Indeed, in the near future, HBAS will feature so many equipment and devices that it will be virtually impossible to use distinct buttons to handle every single function in each one of them. Therefore, Ambient Intelligence (AmI) systems will surface to play an important role in users everyday-life realizing Weiser's landmark vision (Weiser, 1991), developing an increasing reliability. (Weiser, 1993; Bohn et al., 2005).

Given the ever growing development of intelligent consumer electronics equipment and their capabilities, Home and Building Automation Systems have received increasing attention from the Computer Science community (Merz et al., 2009). Due to the high variability of user activities and building infrastructures, the implementations of Home and Building Automation Systems (HBAS) is largely ad-hoc. Implementation differ considerably depending on whether it is a small home, a large building, or commercial facility. Implementation might also change among equivalent building structures due to the constantly changing needs of the end-user or the architectural purpose of the building. Overall, infrastructure heterogeneity and the lack of established standards hamper HBAS take off.

The crucial aspect of Ambient Intelligence is the capability to react (or even anticipate) users needs and actuate in a way that is consistent with the users expectations. For this vision to become possible a large number of devices must coordinate themselves and adjust taking into account a large number of variables ranging from environment conditions, to space characteris-

*Corresponding author

tics or even user's emotional state. Creating these systems poses a number of challenges to computer science. Imagine a system that will adjust the light ambience of a room to match the type of movie being played on the TV that also takes into account the amount of natural light entering the room, thus closing the window blind if required. However, multiple contexts may coexist inside an automated environment triggering distinct actuations that may conflict with each other. Existing literature address conflict resolution in HBAS through simple resource management or priority rule mechanisms (Armac et al., 2006; Retkowitz and Kulle, 2009; Huerta-Canepa and Lee, 2008). Moreover, ontology-based approaches do not encompass query knowledge analysis for energy efficiency (Corno and Razzak, 2012; Wicaksono et al., 2010).

Consequently, it is still necessary to develop automation systems able to intelligently reason about relevant environment conditions extracted from models that represent buildings in terms of automated spaces and the services that they offer. This article presents an ontology-based solution that performs knowledge analysis by means of queries to infer context. The system models conflict and energy efficiency optimization as instances of the Constraint Satisfaction Problem (CSP) in order to perform environment actuations from the attained solutions.

2. Related Work

There is a great deal of work and literature concerning general conflict detection and resolution systems and HBAS separately. Automatic conflict resolution in HBAS is a relatively new topic in computer science and there are few solutions that address this issue directly. This section reviews the most relevant literature and work establishing adequate comparisons, extracting advantages and shortcomings.

Table 1 expresses an overview on the presented solutions in terms of approaches and the features surveyed. According to Table 1, many proposed systems rely on avoidance and resource management techniques for dealing with conflict. Others use policy-based analysis for detection and constraint satisfaction or complex algorithms as resolution methodology. The surveyed solutions are capable of solving most of the problems we considered to be major issues in AmI. However, this survey shows an heterogeneous tendency in conflict approaches. Furthermore, most of the presented solutions do not feature energy efficiency capabilities or apply any sort of knowledge-based analysis. In other words, the surveyed solutions do not present adequate approaches to conflict detection and resolution in HBAS.

H. Wicaksono Wicaksono et al. (2010) perform knowledge based analysis using SPARQL Protocol and RDF Query Language (SPARQL) to acquire environment information. We believe that the ontology expressive power and flexibility is a major surplus regarding information extraction for conflict detection, which is why we are also using it in our framework. Information can be extracted and manipulated from the ontology model in order to keep track of system states that may lead to conflict. This is better than other options such as avoidance approaches in the sense that they can only be used as temporary measure or as a permanent means of disposing of a matter. Moreover, the Thomas and Kilmann grid Thomas (1992) considers avoidance strategies as a lose-lose proposition since it does not address the issue at hand. Other conflict detection approaches such as policy overlap or rule based detection might suffice to some frameworks but lacks the expressive semantic power and compatibility offered by ontologies.

Moreover, we believe that the proper way to perform conflict resolution operations in automated environments is one adequately modelled by ontologies, in which environment states are represented by a set of variables that must dynamically adapt in order to comply with user's preferences and energy efficient constraints. The most advantageous combination of variables should then be detected by applying constraint programming techniques. Yet, according to our survey, only F. Corno and HOMER solutions use CSP solvers to resolve conflict. Energy saving is an issue that can also be modelled as an instance of CSP, so that system actuations that aim at satisfying the majority of occupants can simultaneously increase energy efficiency. Our solution, like F. Corno's, models the problem in a way that the resolution module not only searches for solutions to resolve inter-personal conflict but also aims at finding the most energy efficient set-up within that solutions space.

3. A solution for automatic conflict resolution

Our proposed framework consists of three main modules responsible for aggregating context information, detecting conflict, resolving conflicts and deciding which action to take based on the current state of the environment. The system uses an ontology as knowledge base as input, and outputs commands to the actuators and notifications to users. The analyser module

Approach	Systems	Formal Model (Ontology)	Conflict Detection	Conflict Resolution	Energy Efficient
Multi-Agent Systems	HOPES & HECODES (Bell and Grimson, 1992)	NO	Avoidance	-	NO
Policy Based Systems	HOMER (Maternaghan and Turner, 2013)	NO	Policy Overlap	CSP - JaCoP	NO
	CARISMA (Capra et al., 2003)	NO	Policy Based	Micro Economic Mechanism	NO
Interest Based Systems	COMITY (Tuttlies et al., 2007)	NO	Avoidance	-	NO
	(Armac et al., 2006)	NO	Rule-Based Detection	Rule Mechanism & Priority Management	NO
	(Park et al., 2005)	NO	Intention Difference	Cost-Function	NO
	(Silva et al., 2011)	NO	Intention Divergence	Algorithm Selection	NO
Authorization Based Systems	(Masoumzadeh et al., 2007)	NO	Potential Conflict Graph	Precedence Based	NO
Resource Based Systems	(Retkowitz and Kulle, 2009)	NO	Avoidance/Prevention	Resource Management	NO
	(Huerta-Canepa and Lee, 2008)	NO	Avoidance (Device Control)	Resource Management	NO

Table 1: Overview and comparison of presented HBAS solutions in terms of approach, modelling, conflict detection/resolution functionalities and energy efficiency support.

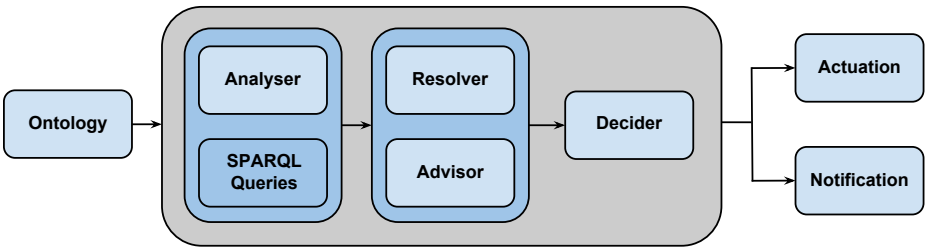

Figure 1: Prototype framework of the conflict detection and resolution system, in terms of its constituent modules: *(i)* analyser module, *(ii)* processor module composed by the resolver and advisor, and *(iii)* decider module. The system takes the ontology model and SPARQL queries as input and actuates upon the environment or returns information to occupants.

uses context information provided by the ontology making use of pre-defined SPARQL queries to extract information from the ontology about potential conflict situations. Finally, the information is translated into valid input data to feed the processor module responsible for performing mathematical operations to find solutions. Both the resolver and advisor apply constraint programming techniques over the elements returned by the analyser, modelling them as constraints. These constraints denote a set of solutions of which, the most context-adequate to the problem is to be chosen. The decider receives the solutions and data generated by the resolver and advisor and reads actuators data in order to verify whether device actuations are needed to make the environment adapt to the current situation.

Figure 1 represents the prototype framework architecture in terms of it's constituent modules and data flow.

3.1. Ontological representation of conflict

This section focuses on possible conflict situations that may occur in the system state.

Consider a scenario of conflict described as follows: let 'Alice' and 'Bob' be the 'Living Room' performing the same activity 'Read', and have preferences over the same environment variable 'Light'. 'Charles' and 'David' are also inside the 'Living Room' but intend to perform different activities, 'Study' and listen to 'Music'. They share preferences over multiple environment variables related to 'Temperature' and 'Audio'. Potential conflict between Charles and David can be seen as a multi-dimensional variant of the one between Alice and Bob. In the 'BedRoom', however, 'Frank' and 'Eve' do not have preferences over any environment variable, but both perform different activities, 'Sleep' and watch a 'Movie' with pre-defined conditions over environment variables.

This scenario, illustrated in Figure 2, portrays preference divergence cases where users exhibit distinct inclinations regarding lighting, sound, temperature, air flow levels, as well as activity incompatibility situations where conditions to start one activity interfere with the conditions to keep another running, ultimately resulting in conflict. This sort of state must be automatically detected in order for the system to dynamically adapt the environment's conditions through device actuations in order to comply with the preferences of both occupants and activity's conditions, thus resolving conflict. To this aim, the system performs a knowledge-based analysis upon the model by means of queries that extract information to find states that represent conflicting situations like the ones represented in the example.

3.2. Model analysis using SPARQL

Shared spaces environments are dynamic and always changing. Thus, the ontology model must keep up with state changes to accurately reflect the environment. The model analysis is performed by a module responsible for extracting information about the environment. Such information allows us to know which occupants are installed in which rooms, which activities are being performed, and which environment variables are being used. It is this information that enables determining possible conflicting user preferences over the same environment variables or which activity conditions may interfere with each other. Put differently, model analysis allows the system to recognize conflicting situations and extract information in order to proceed with resolution.

Consider the system state depicted by Figure 2 that represents an environment state that depicts possible conflicts among activities and user preferences. In this case, it consists on preference mismatch situations between 'Alice', 'Bob', 'Charles' and 'David'. The activities 'Movie' and 'Sleep' may also interfere with each other due to environment conditioning conflicts. Therefore, analyzing the current state of the environment to find information about this sort of conflict, undergoes querying the model for:

1. Any environment variable preferred by more than one occupant inside the same shared space.
2. Any environment variable that is conditioned by more than one activity inside the same shared space.

Figure 3 presents the SPARQL queries that encode such logic. The query results are displayed in Table 2.

	room	**variable**	**occupant**
Query 1	LivingRoom	Light	Bob
			Alice
		Temperature	David
			Charles
	room	**variable**	**activity**
Query 2	BedRoom	AirFlow	Sleep
			Movie
		Sound	Sleep
			Movie

Table 2: Query results identifying that 'Bob' and 'Alice' may have conflicting preferences regarding 'Light' in the 'LivingRoom'. 'Charles' and 'David's' preferences over 'Temperature' levels may also interfere. 'Sleep' conditions 'AirFlow' and 'Sound', which interfere with the conditions imposed by 'Movie'.

For simplicity reasons, data properties that express the preference intervals regarding each environment variable levels are omitted. The extracted information provides relevant data about possible conflict situations, that are crucial for the resolution process. This data will be used as input of the *resolution* module.

3.3. Syntactic representation of the environment

The environment's conditions syntax can be such that variables $(X_1, ..., X_n)$ are restricted by a set of constraints $C_1, ..., C_n$. A constraint system is denoted as $P = C_1, ..., C_n$. Where θ is an assignment of values to all variables, $\theta : X_1 \cup ... \cup X_n$ in their respective domain. When this mapping of values is such that a constraint C holds, then θ is said to *satisfy* C, denoted $\theta \models C$. If there is an assignment θ of values, that satisfies all constraints in P at the same time, then θ is a *solution* for P, denoted $\theta \models P$. If no solution can be found it is said that P is *unsatisfiable*.

4. Validation

The validation process regarding the presented hypothesis consisted in implementing the prototype framework. As expressed in previous chapters, the development process entailed three main steps: *i)* development of the semantic model that would reproduce the environment-reflecting ontology; *ii)* performing an environment analysis by extracting information from the ontology using SPARQL queries; *iii)* generating environment solutions using constraint solving techniques.

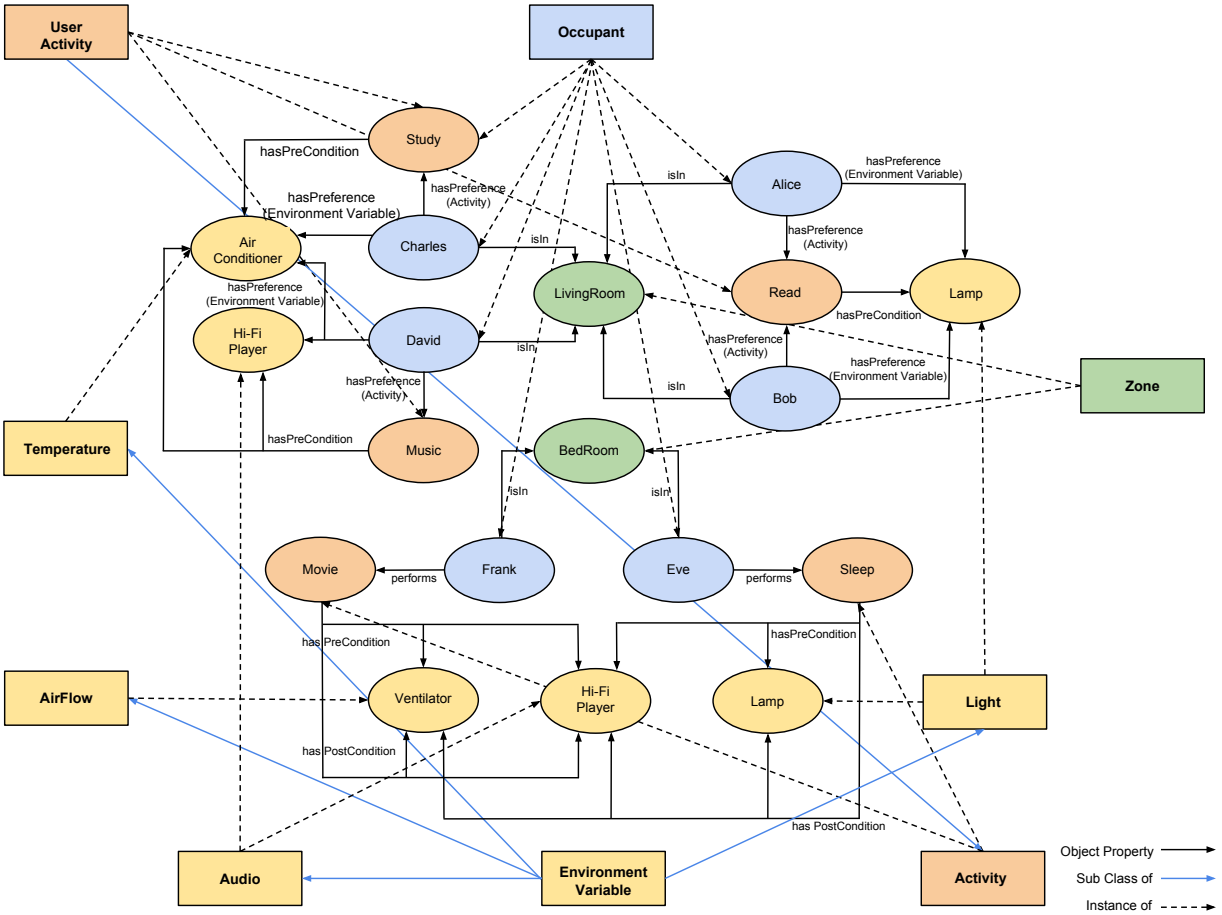

Figure 2: A graph representation of the ontology reflecting the current state of the environment as described in the scenario of section 3.1. Occupants have preferences over a set of environment variables depending on the activities they want to perform. Moreover, activities whose conditions may interfere with another's, are taking place inside the same room. Ellipse-shaped objects represent individuals sharing relationships, ultimately expressing the environment state. Rectangular-shaped objects represent classes or sub-classes.

The undertaken methodology for validating the proposed solution basically consisted of assuring the semantic model consistency using ontology reasoners and validators, detecting conflicting environment states, and finally producing valid solution values for well defined conflict scenarios in order to evaluate the system's response and attest its validity.

This section introduces an ontology validation procedure as well as several conflict oriented scenarios with respective generated output and a practical discussion regarding the obtained results.

4.1. Ontology validation

The ontology model was developed using Protegé[1], an open source knowledge-base framework that supports modeling ontologies in a variety of formats.

FacT++ is a Protegé built-in reasoner implemented in C++ that supports OWL DL and OWL 2 DL using optimized tableaux algorithms. The logic consistency of our ontology was validated by FacT++, producing the following output.

```
Initializing the reasoner by performing the following steps:
        class hierarchy
        object property hierarchy
        data property hierarchy
        class assertions
        object property assertions
        same individuals
FaCT++ classified in 127ms
```

Figure 4: Output produced by Protegé during execution of FacT++ built-in reasoner.

Furthermore, the ontology was validated as an OWL 2 DL profile by the University of Manchester OWL Val-

[1] http://protege.stanford.edu/

```
PREFIX ont: <http://www.semanticweb.org/rui/ontologies/2013/11/automation#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT ?room ?variable ?occupant
FROM <http://web.ist.utl.pt/rui.camacho/Ontologies/automation.rdf>
WHERE {
    ?occupant ont:hasPreference ?something; ont:isIn ?room.
    ?occupantB ont:hasPreference ?somethingB; ont:isIn ?roomB.
    ?variable rdfs:subClassOf ont:EnvironmentVariable.
    ?variableB rdfs:subClassOf ont:EnvironmentVariable.
    ?something a ?variable.
    ?somethingB a ?variableB.
    ?property rdfs:subPropertyOf ?anyproperty.
    ?something ?property ?value.
    FILTER (?occupant != ?occupantB && ?variable = ?variableB && ?room = ?roomB)
}

ORDER BY ?occupant

SELECT DISTINCT  ?room ?variable ?activity
FROM <http://web.ist.utl.pt/rui.camacho/Ontologies/automation.rdf>
WHERE {
    ?activity ont:hasCondition ?ev. MINUS{ ?activity a ont:UserActivity}
    ?activityB ont:hasCondition ?evB. MINUS{ ?activityB a ont:UserActivity}
    ?activity ont:isIn ?room.
    ?activityB ont:isIn ?roomB.
    ?variable rdfs:subClassOf ont:EnvironmentVariable.
    ?variableB rdfs:subClassOf ont:EnvironmentVariable.
    ?ev a ?variable.
    ?evB a ?variableB
    ?property rdfs:subPropertyOf ?anyproperty.
    ?ev ?property ?value.
    FILTER (?activity != ?activityB &&?room = ?room_b && ?variable = variableB)
}
ORDER BY ?activity
```

Figure 3: Example SPARQL queries that return concurrency information upon environment variables for each shared space and instances of activities whose conditions may interfere with the conditions of another activity. The queries select the tuples that comply with a set of constraints: *(a)* occupants must be inside the same room and have preferences over the same sub-class of environment variable (i.e. Light, Temperature, Sound or Air Flow); *(b)* activities are in the same room and condition the same environmnet variables.

idator [2].

4.2. Test Cases

Herein, we present several practical test scenarios that aim at verifying the system performance regarding the most diverse environment states. In other words, the test cases approach both custom and default activity conflict as well as the space advising module decisions regarding different well defined situations.

4.2.1. Services Testing

The services offered by each space determine the purpose of which a given space shall be used for. Services specify conditions over environment variables and activities to be executed in a space must comply with those services. As such, it is necessary to verify whether the system correctly fits candidate activities with the services available inside the space they are to be performed.

Consider that a system managed refrigerator room, which sole purpose only applies to conserving food.

Default Activity / Service				
Space	**Environment Variable**	**Activity/ Service**	**Condition**	**Satisfied**
Refrigerator Room	Temperature	(Service) Refrigerate	0 - 5	True
		Conserve Meat	1 - 4	True
		Read	21 - 23	False
		Movie	20 - 22	False
		Sleep	22 - 24	False
			Solution:	**2**

Table 3: System's produced results to the ontology that reflects the test case where four distinct activities compete for the resources of a space that offers a service that only complies with one of them.

Furthermore, there are four distinct activities, **Read**, Movie, **Sleep** and **Conserve Meat**, three of which are not meant to be carried out inside a refrigerator. All of them, specify preferences over the same environment **Temperature**. Figure 5 represents the ontology that reflects the environment state described above. Normally, the system satisfies the majority of the competing activities. However in this case, only one activity complies with the offered service, so it is to be expected that the system overrides that logic in order to avoid the space

[2] http://mowl-power.cs.man.ac.uk:8080/validator/

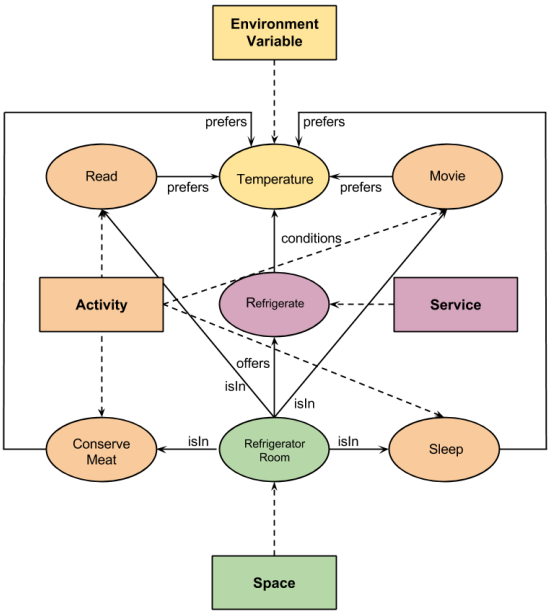

Figure 5: Diagram representing the ontology that reflects the test case where four distinct activities compete for the resources of a space that offers a service that only complies with one of them.

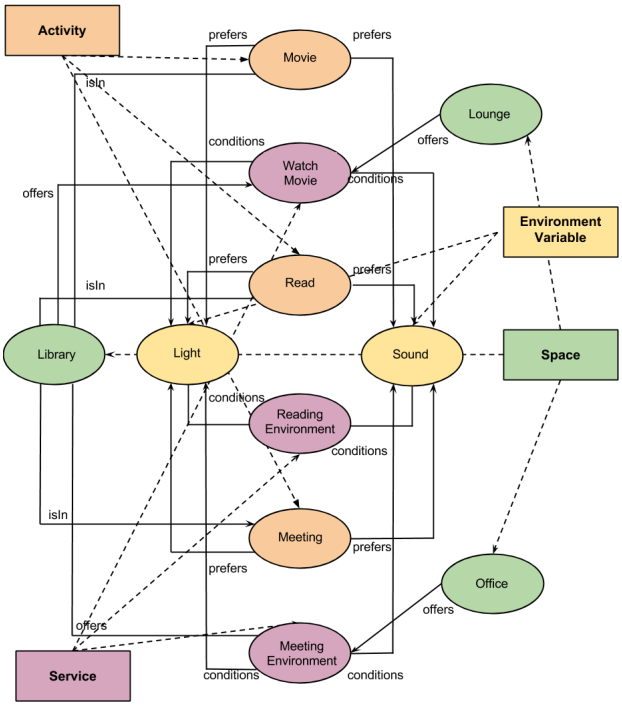

Figure 6: Diagram representing the ontology that reflects the test case where three distinct activities compete for the resources of the same room and the discarded are suggested to be executed inside other proper spaces.

being used for the wrong purposes.

Table 3 express the system's generated ouput solution to this particular test scenario. Results show that although there is a majority of activities that can be satisfied, the system effectively choses to satisfy the one that complies with the service. That is possible because the system immediately excludes the activities that do not fit with the services offered by the space in which they are trying to be executed.

4.2.2. Advisor functionality testing

Consider the scenario where three distinct activities, **Movie**, **Read** and **Meeting**, are to take place inside the same space **Library** and **Lounge** and **Office** spaces are empty. In this case, each space offers one service, **Read Environment**, **Watch Movie** and **Meeting Environment** respectively, conditioning two environment variables, **Sound** and **Light**, both also offered by all three spaces. Figure 6 depicts the described environment state. Furthermore, consider that the set of preferences regarding sound and lighting is such that **Read** is the only activity that fits into the service offered inside the library, in this case Reading Environment. It is expected that the system is not able to satisfy both Movie and Meeting. Moreover, the two remaining rooms, Lounge and Office offer services that are able to accommodate the activities that are not satisfied inside the Library. The system is also expected to suggest whoever is trying to watch a movie and have a work meeting to relocate to the Lounge and Office respectively.

Table 4 represents the system output to the ontology associated to the described environment state. Results show that the Meeting was immediately set aside because it was the less energy efficient activity when competing against Read. Activity Movie was also discarded because the service offered by the Library does not allow the execution of that particular activity, letting Read be the only activity occurring inside that space at that time. However, the advisor module kicks in and analyses whether the discarded activities may still be executed in different spaces. The ontology representing this test scenario still indicates that there are two more spaces where activities can occur. Since activities Movie and Meeting fit perfectly with the services offered in the Lounge and Office respectively, the advisor module suggests users performing each activity to relocate, as shown in the results, to the right spaces.

4.2.3. Default vs custom activities

The environment state of the managed spaces is dynamic and ever changing. This environment state is rep-

Default Activity / Service				
Space	**Environment Variable**	**Activity/ Service**	**Condition**	**Satisfied**
Living Room	Sound	Reading	0 - 20 **Solution:**	True **10**
Living Room	Light	Reading	200 - 300 **Solution:**	True **250**
Living Room	Sound	Movie	58 - 65 **Solution:**	False **-**
Living Room	Light	Movie	5 - 10 **Solution:**	False **-**
Living Room	Sound	Meeting	60 - 65 **Solution:**	False **-**
Living Room	Light	Movie	250 - 350 **Solution:**	False **-**
Lounge	Sound	Watch Movie	0 - 65 **Solution:**	True **32**
Lounge	Light	Watch Movie	0 - 600 **Solution:**	True **300**
Office	Sound	Meeting Environment	0 - 65 **Solution:**	True **32**
Office	Light	Meeting Environment	200 - 600 **Solution:**	True 400
Advisor Module				
Movie Meeting		goes to goes to		**Lounge Office**

Table 4: System's produced results to the ontology reflecting the test scenario where an automated space tries to host three default activities at the same time, but does not offer all the necessary services to accommodate all of them.

resented by the ontology individuals (instances) and relations among each other. Furthermore, occupants may opt to perform system default (i.e. predefined) activities or perform a personalized version of those activities which specify the individual preferences of users. As such, it is not only possible but inevitable that default and custom activities come to compete for the resources of a given space at the same time.

Consider the case where occupants **Alice** and **Bob** are inside the **Office** where the default activity **Study** is already taking place by a number of users, establishing preferences over the environment variable **Temperature**. Moreover, both Alice and Bob also establish their own preferences over the same environment variable. Figure 7 depicts the ontology that represents the environment state at that time.

It would be sensible that the system took default activities in high priority over custom activities. Consequently, it is expected that, due to the presence of people trying to perform default activities inside the Office, the

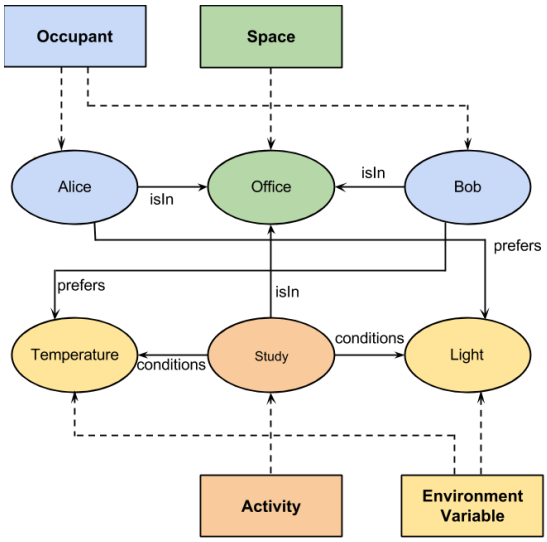

Figure 7: Diagram representing the test scenario where default and custom activities are taking place inside the same space.

system satisfies custom activities as long as they do not interfere with the solutions found to settle the default activities. In other words, default activities are solved first and custom activity preferences must fit inside those solutions in order to be considered satisfied as well.

Table 5 expresses the system's generated results to this particular scenario. Results show that both preferences imposed by Alice and Bob are indeed satisfied and the produced solutions comply with the service conditions.

4.3. Energy efficiency

Energy efficiency operations are carried out in the solution computation phase of the execution flow. Ideally, the system would have an energy monitoring component, or module, in order for the system to analyze energy consumption and engender a cost function in terms of environment variable values. Given a generated set of possible solutions, the system would chose the one that minimized the cost function, thus finding the most energy efficient environment variable value. However, no energy monitoring component was integrated in the system, making this approach unfeasible. On the other hand, most ambient devices have an energy consumption proportional to the levels they are set to. As such, the system choses the minimum value from the solution set in order to avoid unnecessary energy expenditure.

4.4. Discussion

The previous sections presented ontology consistency and validation methodologies, as well as several conflict

Default Activity / Service				
Space	**Environment Variable**	**Activity/ Service**	**Condition**	**Satisfied**
Office	Temperature	Study	20 - 24	True
Office	Light	Study	500 - 505	True
Custom Activity				
Space	**Environment Variable**	**User**	**Preference**	**Satisfied**
Office	Temperature	Alice	23 - 25	False
Office	Temperature	Bob	20 - 22	True
			Solution:	21
Office	Light	Alice	450 - 500	True
Office	Light	Bob	510 - 550	False
			Solution:	500

Table 5: System's produced results to the ontology reflecting the test scenario where an automated space hosts both services and user preferences conditioning the same environment variable.

scenarios to evidence the system's efficiency and accuracy.

The analyzed system output showed to be valid solutions for the theoretical CSP formal expressions for all presented scenarios and all experiments carried out during the implementation phase, making the possibility of incorrect output almost negligible. Furthermore, there was no evidence non-deterministic output for a static ontology, in other words, the system always produces the same solution for a constant environment state. As such, the system's output accuracy and reliability is attested.

The most time-consuming phase takes place in the resolution module during the execution of constraint-solving operations. The time it takes to compute CSP operations varies in terms of variables and constraints. However, in the scope of HBAS it is possible to perform such operations in sensible time. In this case, none of the presented conflict scenarios exceeded the 200 ms threshold, thus evidencing the system's efficiency.

As mentioned before, the time it takes for the system to generate solutions varies in function of constraints and variables. More specifically, with the number of automated spaces and users/activities. Basically, the system scalability fails only if the number of managed spaces is such that the time consumed for generating an iteration output is greater than the time interval between iterations. It is inferred that the system is able to manage a large set of buildings, depending in size or number of automated spaces, before starting to reveal performance issues.

As for real-world applicability, it would be necessary to deploy the system inside an automated laboratory and test it with real users. On the other hand, there were some implications that challenged this approach. A network of sensors and a context-aware module would have to be provided and developed respectively to feed the ontology. Unfortunately, those resources were not available. Nonetheless, it is expected a good level of real-world practicality in future works, given the promising results presented by this article.

5. Conclusions

Developing highly intelligent and adaptive Home and Building Automation Systems (HBAS) is a complex multi-domain problem that is knowing increasing relevance. However, with respect to solving conflict arising in multi-user scenarios, HBAS are still to reach a maturity level that enables them to automatically handle highly subjective contextual information regarding users intentions and interfering actuations, namely with respect to solving conflicts arising in multi-user scenarios.

The prototype solution proposed herein for conflict detection and resolution is able to perform context analysis based on an ontology that formally represents environment's conditions. Conflict detection capabilities are powered by a knowledge-analysis module that enables the recognition of potential conflicts. The conflict resolution operations are carried out by means of constraint solving enabling to respond automatically to a diversity of decision-demanding scenarios. Moreover, we presented a set of relevant frameworks and libraries to be used in the prototype's implementation.

In order to evidence the potential value addition and main contributions of our work, a real world scenario is presented. We observe that our framework performs automatic environment adaptations on behalf of users according to their comfort preferences while, at the same time, maximizes energy efficiency by setting actuators to the least energy demanding configuration.

Future work will deal with a number of issues which must be tackled to allow the development of intelligent automation systems that act predictively on behalf of users. One research direction worth pursuing is to investigate how knowledge-based automation systems can be effectively integrated with context-aware components. Another direction is to investigate whether it is possible to perform knowledge-based analysis and constraint solving operations in a centralized architecture where actuations are sent to clients deployed in remote buildings. Finally, it will be explored the possibility of optimizing the system with learning capabilities to automatically infer user preferences based on observation.

In addition, we plan to integrate the proposed framework in a real-world building setting in the scope of the smart campus project[3].

Acknowledgements

This work was supported by national funds through FCT Fundação para a Ciência e Tecnologia, under project PEst-OE/EEI/LA0021/2013.

References

Armac, I., Kirchhof, M., and Manolescu, L. (2006). Modeling and Analysis of Functionality in eHome Systems: Dynamic Rule-based Conflict Detection. In *13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06)*.

Aros (2014). Aros: A truly brilliant air conditioner. https://www.quirky.com/aros. (Last accessed on April, 2014).

Bell, D. A. and Grimson, J. (1992). *Distributed Database Systems*. Addison-Wesley, Boston, MA, USA.

Bohn, J., Coroamă, V., Langheinrich, M., Mattern, F., and Rohs, M. (2005). Social, economic, and ethical implications of ambient intelligence and ubiquitous computing. In *Ambient intelligence*, pages 5–29. Springer.

Capra, L., Emmerich, W., and Mascolo, C. (2003). Carisma: Context-aware reflective middleware system for mobile applications. *Software Engineering, IEEE Transactions on*, 29(10):929–945.

Corno, F. and Razzak, F. (2012). Intelligent energy optimization for user intelligible goals in smart home environments. *Smart Grid, IEEE Transactions on*, 3(4):2128–2135.

Echelon (2003). Echelon smart buildings. http://www.echelon.com/applications/smart-buildings/. (Last accessed on April, 2014).

Hue (2013). Phillips hue connected light bulb: Personal wireless lighting. http://meethue.com/. (Last accessed on April, 2014).

Huerta-Canepa, G. and Lee, D. (2008). A multi-user ad-hoc resource manager for smart spaces. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–6. IEEE.

LIFX (2012). Lifx: The lightbulb re-invented. http://lifx.co/. (Last accessed on April, 2014).

Masoumzadeh, A., Amini, M., and Jalili, R. (2007). Conflict detection and resolution in context-aware authorization. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 1, pages 505–511. IEEE.

Maternaghan, C. and Turner, K. (2013). Policy Conflicts in Home Automation. *Computer Networks*, 57(12):2429–2441.

Merz, H., Hansemann, T., and Hübner, C. (2009). *Building Automation: Communication Systems with EIB/KNX, LON und BACnet*. Signals and communication technology. Springer.

Nest (2012). Nest: Learning thermostat. https://nest.com/. (Last accessed on April, 2014).

Park, I., Lee, K., Lee, D., Hyun, S. J., and Yoon, H. Y. (2005). A dynamic context conflict resolution scheme for group-aware ubiquitous computing environments. In *Proceedings of the 1st International Workshop on Personalized Context Modeling and Management for UbiComp Applications (ubiPCMM 2005)*, pages 42–47.

Retkowitz, D. and Kulle, S. (2009). Dependency management in smart homes. In *Distributed Applications and Interoperable Systems*, pages 143–156. Springer.

Siemens (2012). Building automation systems maximum confort and perfect functionality a minimum cost. http://www.buildingtechnologies.siemens.com/bt/global/en/buildingautomation-hvac/building-automation/Pages/building-automation-system.aspx. (Last accessed on April, 2014).

Silva, T. R. B., Ruiz, L. B., and Loureiro, A. A. (2011). Conflicts treatment for ubiquitous collective and context-aware applications. *Journal of Applied Computing Research*, 1(1):33–47.

Smart-Zone (2012). Samsung smart-zone: Air conditioning. http://www.samsung.com/au/air-conditioning/smart-zone/. (Last accessed on April, 2014).

Smartthings (2012). Smartthings: Hello, smart home. http://www.smartthings.com/. (Last accessed on April, 2014).

Staples-Connect (2013). Staples connect: Connected home made easy. http://www.staples.com/sbd/cre/marketing/staples-connect/. (Last accessed on April, 2014).

Thomas, K. W. (1992). Conflict and conflict management: Reflections and update. *Journal of Organizational Behavior*, 13(3):265–274.

Tuttlies, V., Schiele, G., and Becker, C. (2007). Comity-conflict avoidance in pervasive computing environments. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, pages 763–772. Springer.

Weiser, M. (1991). The Computer for the Twenty-First Century. *Scientific American*, 3.

Weiser, M. (1993). Hot Topics: Ubiquitous Computing. *IEEE Computer*, 26(10):71–72.

WeMo (2013). Belkin wemo home automation. http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/. (Last accessed on April, 2014).

Wicaksono, H., Rogalski, S., and Kusnady, E. (2010). Knowledge-based intelligent energy management using building automation system. In *IPEC, 2010 Conference Proceedings*, pages 1140–1145. IEEE.

[3] http://greensmartcampus.eu/