BuildingRules: A Trigger-Action Based System To Manage Complex Commercial Buildings

Alessandro A. Nacci
Politecnico di Milano
P.za Leonardo da Vinci
20133, Milano, Italy
alessandro.nacci@polimi.it

Rajesh Gupta
UCSD
9500 Gilman Dr.
92093, La Jolla, California
rgupta@cs.ucsd.edu

Bharathan Balaji
UCSD
9500 Gilman Dr.
92093, La Jolla, California
bbalaji@cs.ucsd.edu

Donatella Sciuto
Politecnico di Milano
P.za Leonardo da Vinci
20133, Milano, Italy
donatella.sciuto@polimi.it

Paola Spoletini
Kennesaw State University
1100 South Marietta Parkway
Marietta, GA, USA
pspoleti@kennesaw.edu

Yuvraj Agarwal
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, 15213, USA
yuvraj.agarwal@cs.cmu.edu

Abstract

Modern Building Management Systems (BMSs) provide limited amount of control to its occupants, and typically allow only the facility manager to set the building policies. BuildingRules let occupants to customise their office spaces using trigger-action programming. In order to accomplish this task, BuildingRules automatically detects conflicts among the policies expressed by the users using a SMT based logic. We tested our system with 23 users across 17 days in a virtual office building, and evaluate the effectiveness and scalability of the system.

Author Keywords

Guides, instructions, author's kit, conference publications

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous.

Introduction

Commercial buildings are complex entities where many elements like HVAC, lightning, fire safety, elevators and security systems have to be coordinated in order to satisfy the different requirements of present day enterprises. Such coordination is generally performed by means of Building Managment Systems (BMSes) [6, 1].

BMSes deployed today are designed for building managers and maintenance personnel. Occupants interact with buildings in a limited manner - using thermostats for HVAC control, switches for lights, keys cards for locks and outlets for plug loads. With existing BMSes, it is not possible for the occupants to automate and personalize their environment such as setting the temperature according to outside weather or automatically brewing coffee at 8am, etc. Recently, new web service based BMSes, are trying to involve occupants in the building settings, in order to improve their comfort and thus their productivity [5] as well as building energy efficiency [3].

Challenges

While giving occupants the ability to personalize their living environment is indeed promising, several challenges have to be addressed. First, building occupants do not understand the details of the building infrastructure, and are not necessarily programmers. As shown in prior work [7], occupants prefer not to interact with sensors and actuators directly; for example they relate better to "someone walked into a room" than "motion sensor was activated". Therefore, it is critical that the right level of abstraction and an intuitive user interface is provided by a building automation system to enable occupants with varying levels of expertise to express their preferences [7]. Second, there needs to be the appropriate access control mechanisms when the number of users – i.e. both occupants and building managers – increase to ensure proper building operation. Finally, with multiple users often customizing the same spaces, there needs to be a scalable mechanism to detect and resolve conflicts that will occur. Existing BMSes have limited or no support for such type of access control or conflict resolution.

Proposed System

In this context, we present BuildingRules, a system which provides an intuitive interface to the occupants of commercial buildings to customize their office spaces using trigger-action programming, under which occupants can express policies using the *"IF something happens THEN do something"* (IFTTT) pattern. BuildingRules automatically detects conflicts among the policies, expressed by the occupants, by using the Z3 SMT solver, and leverages an open source web service BMS (BuildingDepot) to provide access control and actuation services in a building. BuildingRules has been designed to scale for large commercial buildings, as it supports grouping of rooms for ease of policy expression, a scalable backend for resolving conflicts, and a simulator that shows the actuation of rules on a timeline. In a commercial building, typically facility managers set up automation policies using the existing BMS, such as the minimum allowable temperature or air flow. It is critical that occupants customizations don't violate these policies. Furthermore, occupants should not be able to control rooms to which they do not have access to. As automated applications such as Demand Response [2] become prevalent, BuildingRules needs to incorporate the policies expressed by them as well. In BuildingRules , we incorporate hierarchical levels of policy expression to address these challenges, and implement BR by extending an open source webservice-based BMS [1, 8]. Since BuildingRules is targeting commercial buildings, we needed it to scale to many hundreds of rooms and thousands of occupants. We achieve this scaling using several design choices. First, BuildingRules supports specifying a rule for the entire building, or a subset of rooms, using a grouping mechanism in combination with conflict resolution that may be required. Second, we have designed the conflict resolution mechanism to be

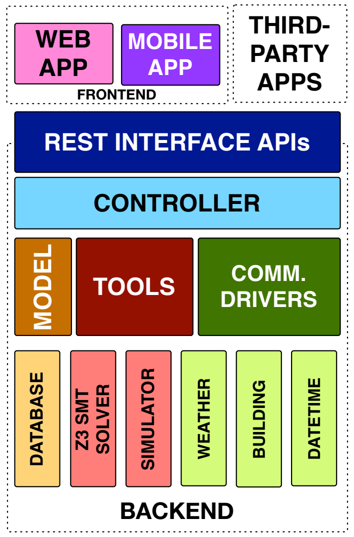

Figure 1: BuildingRules SW architecture

performed in parallel for each room, such that the latency does not increase with the number of rooms.

Automatic Conflict Detection

As said, in BuildingRules, users can express their own rules for rooms, some of which are shared by multiple users, conflicts can arise. We define two rules as conflicting when the rules can be in effect at the same time, but the action specified by the rules cannot be satisfied at the same time. If these conflicts are not resolved properly, it can lead to damage of equipment or compromise user comfort. To clarify, consider two users who independently specify the rules: "if time is between 9am and 6pm then turn the HVAC on" and "if time is between 5pm and 8am then turn the HVAC off". Between 5pm and 6pm, the system would be in an inconsistent state. This may cause discomfort to the occupants and could damage HVAC damper if not actuated properly. To identify the conflicts among rules, we formalize them as propositional formulae and analyze the formalization using the SMT Solver Z3 [4]. A rule is composed of two parts: a (conjunction of) trigger(s), and an action. Before adding a rule, it is verified against the set of rules already active in the room. We represent each rule as a propositional formula composed by an implication (the trigger implies the action) that is satisfied if the trigger is not satisfied, or if both the condition and the action are satisfied. In this context, the action is considered as a proposition that is true if the action can be executed, false otherwise. The new rule together with the existing ones are seen as a specification and automatically verified to check their satisfiability. If the specification is satisfiable, the rules are not in conflict with each other. If not, two or more rules are in conflict and need to be resolved. If a user tries to insert a conflicting rule, the list of the conflicting rules is displayed to the user.

It must be considered also that such a conflict detection mechanism is not able to detect all the possible conflicts: let us imagine to have a rule A:*it is Sunday then turn off the light* and a rule B:*if it is after 6PM turn on the light.* These two rules are not conflicting *per se*, but what is going to happen on Sunday after 6PM? In this case, the user probably wants say something like *generally turn on the lights after 6PM, but keep it off on Sunday*. For this reason we let the users to assign a *priority* to each rule: in this way, then two rules that have to actuate on the same object are triggered at the same time, we select the rule with the higher priority.

Implementation

We have designed BuildingRules as a RESTful HTTP/JSON web service, with a **frontend** for the user interface, and a **backend** which communicates with the BMS, stores information about rules, runs the conflict resolution algorithm and provides RESTful APIs for native mobile applications or building management applications. Figure 1 shows the software architecture of the system. We have implemented BuildingRules in Python 2.7 using the Flask framework.

System evaluation

BuildingRules has been designed for office building occupants to express their preferences using custom automation policies. To evaluate BuildingRules we ran a large user study to analyse the rules expressed, the conflicts detected, and finally, how our system can affect the office environment. We created a virtual office environment with 30 rooms, and testing it on 23 users spread across 17 days. Each participant was assigned to a random set of rooms, for example, an office space, kitchen, and meeting room. The participants were told to use BuildingRules for at least 10 minutes and complete a set of actions, i.e., add, remove, edit rules, each day (10 actions the first day, then decreasing each day. Average of 5 actions per day). A final survey was taken at the end of the week to understand the usability of the system. We obtained a total of 636 rules from this study, and there is an average of 15 rules per room, and 16 rules per user. We show that the conflict detection latency is 251 ms in the worst case, and 102 ms in the average case. 636 rules were specified during the experiment, and we detected up to 50 conflicts in a day. We asked our participants of virtual building study to fill a usability survey to better understand their needs and to evaluate BuildingRules from an occupant's view point. Of the 23 participants, 6 users filled the final survey. Figure 2 shows the results of our survey. The participants liked the idea behind BuildingRules (8.8), thought that it would be useful to have such a system in their office (8.2) and liked the system overall (7.8). However, as the number of rules in a room increased, our UI was not adequate to give an overview of the rules in place. Users found it difficult to create, edit and understand the rules.

Survey Question	
Overall impression score	7.8
System usability	7.6
Would BuildingRules be useful in your office?	8.2
How difficult was it to insert new rules?	8.3
How difficult was it to edit existing rules?	5.9
Do you like the philosophy behind the system?	8.8
How easy was it to resolve conflicts within the rules?	5
Was it easy to understand how the combination of rules will affect the office?	7.2
How useful was the *rule editor* to identify individual rules?	7.2
Was it possible to grasp all rules using *rule navigator*?	7.1

Figure 2: Experimental results

Conclusions

We have presented the preliminary design and the implementation of BuildingRules, a system that enables expression of personalized automation rules in commercial buildings using trigger-action programming paradigm, which can then be integrated with existing Building Management Systems (BMSes) to actuate buildings. We show that when multiple users express different policies for the same physical space conflicts can occur. To resolve these conflicts, we detecting them as rules are inserted using the Z3 SMT solver. We tested our system with 23 users across 17 days in a virtual office building, and evaluate the effectiveness and scalability of the system. The current implementation of BuildingRules is still an experimental tool and further work on usability is required. With this paper we demonstrate that a TriggerAction based mechanism can really help users in managing their offices.

Additional Material

We have uploaded a couple of videos showing how to the system works: `https://youtu.be/oRs51oq8LvQ` and `https://youtu.be/p_J9PcgyATc`. A detailed technical report is available here: `https://goo.gl/wQBRsA`.

Acknowledgment: Part of this work has been supported by Telecom Italia - Joint Open Lab S-Cube. Thanks to Jacopo Fiorenza and Andrea Mariani for their effort.

References

[1] Agarwal, Y., Gupta, R., Komaki, D., and Weng, T. Buildingdepot: an extensible and distributed architecture for building data storage, access and sharing. In *Proc. of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, ACM (2012).

[2] Alliance, O. Openadr 2.0 profile specification, a profile.

[3] Balaji, B., Teraoka, H., Gupta, R., and Agarwal, Y. Zonepac: Zonal power estimation and control via hvac metering and occupant feedback. In *Proc. of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, ACM (2013), 1–8.

[4] De Moura, L., and Bjørner, N. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, 337–340.

[5] Haynes, B. P. The impact of office comfort on productivity. *Journal of Facilities Management 6*, 1 (2008), 37–51.

[6] Johnson Controls. http://www.johnsoncontrols.com/content/us/en/products/building_efficiency/building_management.html.

[7] Ur, B., McManus, E., Ho, M. P. Y., and Littman, M. L. Practical trigger-action programming in the smart home. *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems* (2014).

[8] Weng, T., Nwokafor, A., and Agarwal, Y. Buildingdepot 2.0: An integrated management system for building analysis and control. In *Proc. of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013).