A distributed, collaborative development project of a Policy Engine for use in buildings

Global Software Development

Berntsen, R., raber@itu.dk
Hansen, K., kben@itu.dk
Kokholm, T., tkok@itu.dk
Stanciulescu, S., scas@itu.dk
Wainach, N., nicl@itu.dk

April 16, 2013

Abstract

background, however, ... what we did (the innovative aspects) contributions (design, evaluation) method results / what it means

Contents

Chapter 1

Introduction

Natural resources are a precious commodity. Constructing resource efficient buildings makes sense, both in a political and economical perspective. Modern buildings today might come equipped with a suite of sensors and actuators, opening up for a degree of customizable control. Our collective need is that buildings can adapt to the users and the sensor-perceived environment, either automatically or manually. This can be achieved by developing policies that controls the actuators. Policies can be based on semi-static data, like time and weekdays. However this can have unforeseen and unwanted consequences. For example, a policy governing lightning activated merely by a static time schedule, might entail problems for people attending a rarely occurring late-night party in the building. If the building? event calendar is accessible to the policy engine, a conditional event-checking statement might ensure continuous lighting. However, in order to achieve a more fine grained control, sensory input is needed. We define the interaction of these policies, as a task residing in Facility Management (FM).

Without policies centralized control of a building is highly complex and error prone task, and the building might not be managed in a resource efficient way. By employing a policy engine, with access to the building? sensors and actuators, both the building owner, the users and the administrators of the building benefits from the automation provided. If policies are correctly defined, building owners save energy and natural resources while providing extra comfort to their tenants. Building users can experience a building autonomously adjusting its internal environment to suit their comfort and needs. FM can achieve fine-grained control of the building, at a reduced workload.

In this paper we will; 1) document the collaborative project between the IT University of Copenhagen, Denmark and Strathmore University, Nairobi Kenya. 2) distill requirements from course provided material and a non-exhaustive literature search on policy engines, and 3) develop a software solution that implements these requirements.

Since this project was defined in the course Global Software Development at ITU, we have been provided a Building Simulator, making it up for a real building. The development focus of this project is therefore geared towards this simulator, and not for design and implementation challenges in doing a policy engine for a real building. The end product is a software based management console, that allow for centralized control of sensors and actuators in a building - by implementing a policy engine that allows for automated actuator responses based on sensor feedback, for example closing the blinds in excess sunlight or turning of the heater when windows are open.

1.1 Context

1.2 Problem

Chapter 2

Related Work

2.1 Related work

2.1.0.0.1 Article 1

2.1.0.0.2 Article 2

2.1.0.0.3 Article 3

2.1.0.0.4 Article 4

2.1.0.0.5 Article 5

Chapter 3

Method

3.1 Method

Before any actual work could start, one preliminary goal was to figure out how we could make our group work together as one. Actually this challenge is even more challenging in this project than in a normal work situation: No organization is in order, no predefined roles, no actual project goals and the likes. This chapter will focus on these challenges and how we tried to handle these. We will highlight different methods to create social interaction and understanding. We will focus on how one can rationalize collaboration. Afterwards we will discuss the different tools we used throughout the project life cycle with collaboration in mind. Finally, we will discuss how one can manage a virtual project.

3.1.1 Social Context

When we discuss the *Social Context*, we discuss the direct milieu in which the person is and how different factors can influence this person. Communication is also a part of the social context, which is not necessarily only between two persons but can be between one to many persons, in different time zones, different cultures etc.

3.1.1.0.6 Common ground Our first step to connect to our fellow group mates in Kenya was to introduce ourselves via an e-mail and just shortly highlight some common information about each person from ITU, like stating name, age etc. This method is known as creating "common ground", as introduced by Olson and Olson [**?**]. The term to create common ground "refers to that knowledge that the participants have in common, and they are aware that they have it in common"**REFERENCE**. Common ground is not only established through simple general knowledge about each participant. It is also created through a persons behaviour and appearance through meetings and conversations. We tried to use this method as a way of getting to know our team members, to create a level of understanding and finally to create a stepping stone from which the project could evolve from.

3.1.1.0.7 Trust and First Impression This initial contact was already quite frustrating because of the fact that it was difficult to get a reply from some of the group members in Kenya, only two members was relatively easy to get in touch with. This leads directly to two different considerations in our group work: "Trust"textbfREFERENCE and "First impressions matters"textbfREFERENCE. Trust in group work is a value of high much the different team members trust in each other. How much does one believe that the other team members will deliver their part of the necessary work? How much does one believe that a mail we be answered? How well does the team work together? The trust is between the two subgroups, Denmark and Kenya, relatively low because of the amount -and lack of- replies and general communication. At the time being we only expect one member from Kenya to be online during our team sessions but at the same time we expect everybody from the ITU-group to be online at every session. This is also translatable from the first impressions that we received from the group from Kenya. It is not in any way rewarding for the group atmosphere not to join group conversations and not replying emails.

3.1.1.0.8 Collaboration Readiness The literature for these challenges seems to agree that these sort of problems generally arise from two different topics. One being "Collaboration- and Technology Readiness" and the other "Continuities/Discontinuities". The latter part will be discussed in the section below, **??**.

Collaboration readiness is a potential show stopper for the team work, if a given member is not ready to collaborate. This could be caused by having conflicts in interest, e.g. one is about to overtake another persons job or the likes. This could cause that the person, who is about to lose his job, would not be ready to collaborate. We have tried to identify these issues towards our fellow group members and we cannot find anything that should indicate that they would not be willing to collaborate. They should be just as interested in delivering a good product.

One thing that could cause their lack of interaction in our e-mail correspondences and Skype meetings are the technology readiness. We know that it is a challenge for some of the group members to get internet access because of the fact that not all of them have a connection in their home. Our approach to solve this issue was to have a meeting each Tuesday at 10:00. This way we know that they should have access to their University, which most of the time has an internet access they could use. We know they should have time for this meeting, thus it is planned as a course on their schedule.

3.1.1.0.9 Ethnocentrism A potential threat to the well-being and harmony of the group is known as Ethnocentrism. Ethnocentrism is a state of one subgroup, where the members sees that one group as the centre of everything, and every other group will be valued and ranked from this. A subgroup is a group inside a group, some of the group members are part of - but not the whole group is part of. This way it could create some sense of "Us versus Them", which is something you definitely want to avoid. One way to avoid this is to create multiple subgroups inside the group. One subgroup could be of everybody that likes football and know that they had this in common. If you are part of multiple subgroups the feeling of being part of just one group will dissolve, which should result in a more harmonious group. This way of creating subgroups would be something that you did early on during the initial communication, and something we tried to solve by writing small parts about each other member from Denmark. Unfortunately, we did not receive any feedback from Kenya. This has probably strengthened the feeling of us vs. them, because we do not know much about them. We definitely have a feeling that our group is the centre right now due to the fact that it is only the Danish group that develop and contribute to the project.

3.1.1.0.10 Coupling of work Coupling of work relates to the state of the current tasks and how loosely or tightly coupled they are. A completely loosely coupled work is one you can perform without the interaction and feedback from other persons, this could for instance be some of the work done at a large factory. A tightly coupled work task is, on the other hand, one you only can perform with other members of the group participating.

Our project has evolved from a very tightly coupled project to a more loosely coupled. This is done on purpose. In the beginning of the project everybody had to be at the meetings because of the fact that we had to define which way to move the project. The development life cycle was quite rigid and strict. After the initial phase, where we decided on the platform, chose an architecture etc., more and more tasks became slightly more loosely coupled. This means that one could start to work on his part of the project without any direct interaction with other team members. This would also allow such a highly distributed team as our to collaborate in an efficient way. It would just be to inefficient if everybody had to be together at the same time and place every time anything regarding the project should happen. As of right now we communicate through different Groupware Tools (see **??**) and only meet through one weekly meeting.

3.1.2 Collaborative work

Collaborative work across cultures is a challenge. In our case we had to work together 9 people with 4 being from Kenya, a culture and country that we before entering this project, didn't know much about. As mentioned earlier the social context and the process of creating 'common ground' with the collaborators is of high importance. The goal is to create a fundamental shared understanding of the task and build up the motivation and very much needed trust for the collaboration to succeed. Cooperative work is defined by Schmidt REFERENCE as "People engage in cooperative work when they are mutually dependent in their work and therefore are required to cooperate in order to get the work done," (Schmidt)

3.1.2.0.11 Articulation work The bigger the group the more 'articulation work', articulation work is the extra activities required for collaboration. The task at hand defines what is the actual work and what is articulation work. Articulation work is about who does what, when and where. There are mechanisms of interaction that supports the process when articulation work cannot be handled through every day social interaction. These mechanisms are for instance: Organizational structures (formal/informal), plans, schedules and conceptual schemes. What all these mechanisms have in common is that they all strive to reduce the effort in labor, resources, time, etc. required to handle articulation work. Our strategy for the articulation work was to define processes and choose the groupware technologies that supported our cause best possible.

3.1.2.0.12 Coordination Increased ”reach” of a task changes the coordination. The more spread out between people and artifacts a task is the more the reach is increased. Segregation is a suitable strategy when a complex task is at hand. By dividing the complex task into smaller tasks you get to simultaneously solve them individually and thereby complete the larger complex task faster. It also allows for more specialised teams to investigate a task at a more detailed level. Therefor we segregated the tasks within our project into smaller more comprehensible tasks. Group members were assigned to these tasks that would work closer together until the task was completed.

To handle these tasks we created a project plan with deadlines and milestones to keep track of everything. Moreover we agreed on having a status meeting every week, where we would discuss progress, issues, ideas etc. Arranging a meeting where all is able to attend is not always easy. We did manage to all ’meet’ on Skype at times which took into consideration both the differences in time zones (temporal discontinuities) and the fact that people had entirely different classes and work schedules.

With computer supported cooperative work (CSCW), it’s impossible to anticipate every contingency which might occur, there will always be exception handling. The core challenges and dimensions of cooperative work includes articulation work, adaptation of technologies and awareness. The lack of trust and awareness when you never meet face to face with your collaborators is problematic and requires methods and training when using communication tools. We primarily used Skype to communicate with and made sure to document changes and commits to the code base very detailed. Individuals working together need to be able to gain some level of shared knowledge about each other’s activities.

3.1.3 Groupware Technologies

One way to describe *groupware technologies*, or collaborative software, is that it is a software designed to help people achieve a common work task within a group. One of the earliest definitions of collaborative software is found in the research paper “Rhytms, Boundaries, and Containers” by Peter and Trudy Johnson-Lenz. They describe the software as: “intentional group processes plus software to support them.” (http://nexus.awakentech.com:8080/at/awaken1.nsf/UN
April 1990, Rhythms, Boundaries, and Containers: Creative Dynamics of Asynchronous Group Life, Peter and Trudy Johnson-Lenz). The definition was first stated in 1974. A lot has happend since then, but the way a group

collaborates with and without software support has not. Some general terms are still vital to discuss. We find the most importantly used terms in our project to be critical mass, adaptation and adoption process. These three terms and a general overview of the used tools will be explained in the sections below.

3.1.3.0.13 Tools Generally we used four different tools to communicate our teamwork and current progress; one synchronous and three asynchronous. We used Skype for every team meeting and for general communication between the members of the group. Skype is synchronous communication because of the fact that while using voice chat you will normally get an instant reply. We furthermore used three asynchronous communication methods, that is GitHub, Email and Google Drive. GitHub was used to share the actual projects current status by communicating via tickets what needs to be done. Email was used for communicating general group announcements, that we want to make sure that all of the members from the group receive. Lastly we used Google Drive to share important documents.

3.1.3.0.14 Critical mass Critical mass is a sociodynamic that describes the necessary amount of people needed in order for a product to be a success. An obviously comparison can be made to the ongoing battle between the two social media platforms Facebook and Google+. Facebook has gained a lot of users throughout the last ten years, while Google+ struggles to compete with these numbers. The critical mass is best described by highlighting a real life example, which perfectly describes the term. The current situation of Google+ is that it is far behind Facebook, and we have no real use for it. Google are currently working on gaining a sufficient number of users that will make the product necessary for others to use. This is the goal of achieving the critical mass - to gain enough users for others to see the benefit and the reason for using it.

This description translates quite directly into our usage of groupware technologies, such as Skype and Google Drive. After selecting which tools to use to communicate between the different members, we have made a great deal out of actually using the tools. This has created the critical mass, in our group, for each tool, and every member knows that they can reach each other through these, and thus made it necessary for one to use.

3.1.3.0.15 Adaptation & Adaption Process Adaption and the adaption process are two closely related topics that we briefly touched through our project. These terms describes how well an organization adapts a new technology and how they do it. Some of the problems with new technologies are that human beings tend to have rutines that are not that easy to steer away from. This results in a thought of “Why should I use this new technology, when it works perfect the way I use to do it”. One of the ways we tried to solve these problems was that we chose our tools as a group, and not by enforcing it. This way the majority chose their favourite tool. The adaption process describes how an organization adapts a new tool: The first couple of weeks are extremely important for a succesful adaption. If the adaption does not succeed through these weeks it is very likely that the tool will fade out and never be used again. Our approach to this challenge has been that we pushed it to the people who did not know it, and tried to help them with the setup etc. An example of this could be the installation of Eclipse. One of the members from Kenya had some problems, and we all stayed online throughout the process and helped him install it. This way he could push it to his fellow team members in Kenya.

3.1.4 Virtual Project Management

Virtual Project management is about handling the entire project virtually where the users can access the project from any place and contribute seamlessly in developing the solution. We coordinated activities online through GitHub - handling the code base and reportingupdating issues as we moved forward. The mode of interaction has mostly been through chat and voice via Skype, and not through face to face meetings. This presented various discontinuities.

3.1.4.0.16 Discontinuities Watson-Manheim et al. (2002) (Distance Matters, Except When It Doesn’t: Discontinuities in Virtual Work) examine virtuality in terms of boundaries and discontinuities. They define discontinuities as “a break or gap in the work context,” or a “lack of continuity.” They proposed the concept of discontinuities as a general notion to permit a more comprehensive understanding of the many ways in which virtuality can be perceived. Distance is the most obvious boundary that is encountered in virtual work but it is clear that there are more boundaries such as time, organization, and nationality, which are not usually present in more conventional work settings to the same extent. It is only when those working in virtual settings perceive a boundary to be a discontinuity that it hinders work processes.

General properties of discontinuities are that they can emerge and change over time as people adapt in the teams. Discontinuities may only affect parts of the work. The typical discontinuities are temporal (working across time zones), geographic work location, work group membership (e.g. who you work with), organizational affiliation and cultural backgrounds. However discontinuities can also be expertise related (novice vs experts), historical (different version of a product), different professions (e.g. developers and researchers) or different technologies.

In our group we experienced how the different cultural backgrounds can become a discontinuity. In Kenya if you show up half and hour or an hour later than the time agreed upon, it is seen as okay. While in the danish culture it is problematic. Also when asking Kenyans for feedback or criticism they are often fine with how it is, this might be because they don't want to cause problems and don't want to appear impolite. The concept of teams varies across cultures and organizations, and how teams are perceived will differ based on the organizational and national cultural attributes of its members (Gibson & Zellmer-Bruhn, 2001). It is also critical to note that individuals from different national cultures vary in terms of their group behaviors and communication styles (Gudykunst, 1997).

3.1.4.0.17 Continuities Continuities are the opposite of discontinuities. Continuities are the stable factors in the collaboration that the participants are aware of and consciously act on, or they may be implicit and unrecognized. REFERENCE (Watson-Manheim et al., 2002:200). Continuities can appear to routine or be invisible in order to overcome discontinuities. Often continuities can be described as strategies or factors to overcome discontinuities.

Chapter 4

Design

4.1 Design

During our brainstorm sessions it quickly became clear that we needed a policy engine that was flexible. With inspiration from other building management systems (see related work) and looking at the web service we had to communicate with we came up with the concept of policies that consist of rules.

4.1.0.0.18 Policy A policy in our project is defined as a set of rules that can be activated in a given setting.

4.1.0.0.19 Rules It's in the rules that the power of our engine lies. We have striven to create a model for rules that are flexible and yet simple. A rule is defined as a "'IF THEN ELSE"' statement, that can contain expressions that supports the logical operators of AND, OR and the relational operators LESS THAN, GREATER THAN, EQUALS, NOT.

Using this setup allows us to construct rules in a matter similar to other DSL languages such as SQL. Below is an example from a rule we can construct with our engine:

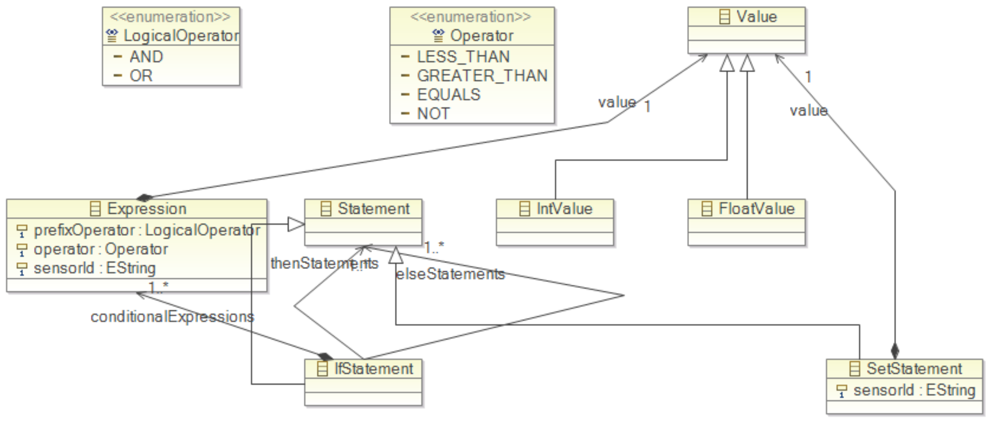

Figure 4.1: Model

numbers

```
{"statements":[

// Define IF statement
{"type":"IfStatement","data": {"conditionalExpressions":[{"prefixOperator":"
    AND","aValue":
        {"type":"IntValue","data":{"theValue":10}},"operator":"EQUALS","
            sensorId":"ROOM1.TEMPERATURE"}],

// Define THEN statement
        "thenStatements":[{"type":"SetStatement","data":{"aValue":
        {"type":"BooleanValue","data":{"theValue":true}},"sensorID":"ROOM1.
            HEATER"}}],

// Define ELSE statement
        "elseStatements":[{"type":"SetStatement","data":
        {"aValue":{"type":"BooleanValue","data":{"theValue":true}},"sensorID":
            "ROOM1.BLINDS"}}]}}]}
```

What happens in this example is ...

4.1.0.0.20 High level design

4.1.0.0.21 Limitations

... Alternatives?

Chapter 5

Implementation

5.1 Implementation

This chapter, **??**, is dedicated to the actual low-level implementation of the presented system. In this chapter we will explain in detail what has been implemented and how. We will give an overview of the whole project, explain how the different components communicate with each other and finally describe the flow - from creating a policy to applying it.

5.1.1 System

In this section, **??**, we will elaborate on the implementation for this project. We will delve into the system that the project is running on. We will shortly discuss the technological choices we have made and explain our reasons so.

5.1.2 Interface Component

5.1.3 Abstraction Component

5.1.4 Storage Component

5.1.5 Request Flow

5.2 Front-end

5.2.1 Initial Ideas

Initial ideas. User-interaction

The main idea for the user interface was to keep is simple and user-friendly without loosing advance functionally[1]. To better understand what we were dealing with we initial started with some rough drawings, planing ahead how the visuals would look like, and the functionally of the interactions with the user.

We knew that the project could easily expand and become far too complex to we set up implementing a simple skeleton with the most basic elements needed.

The primary parts of the front-end include:

Listing already existing and active policies:

- Adding, deleting and modifying policies
- A visual representation of the building with floor and rooms (like a map)
- Visual representation of sensor values group towards the individual rooms they represent.

5.2.2 Sketch-up-s drawings of original ideas

Sketch-up’s drawings of original ideas. Final visual interface

5.2.3 Usability

Something about the choices made towards the usability of the user interface, or at least ideas how we would improve the user interface to be more user-friendly.

5.2.4 Technology Choices

Why Java, benefits? due to policy engine with object classes etc. Original plan to use PHP expected it to be simple to use Java classed to handle the policies.

Chapter 6

Evaluation

6.1 Evaluation

Evaluate the project.

Chapter 7

Discussion

7.1 Discussion

This is where we will discuss our process, collaboration and overall project!

7.1.1 Collaboration

Discuss the collaboration. What was good? What was bad? What could one do to make it better?

7.1.2 Project

Discuss the project. What was good? What was bad? What could one do to make it better?

Chapter 8

Conclusion

conclusion...

Bibliography

[1] Steve Krug. Don’t Make Me Think: A Common Sense Approach to the Web (2nd Edition). New Riders Publishing, Thousand Oaks, CA, USA, 2005.

[2] Gary M Olson and Judith S Olson. Distance matters. Human-Computer Interaction, 15(2):139–178, 2000.

Appendix A

Appendix

A.1 A.1. Requirements

A.2 A.2. Tests