NTNU – Trondheim
Norwegian University of
Science and Technology

Appliance simulation models for the evaluation of energy management policies

Shie Chen

Master of Telematics - Communication Networks and Networked Services (2
Submission date: May 2014
Supervisor: Finn Arve Aagesen, ITEM

Norwegian University of Science and Technology
Department of Telematics

NTNU - Trondheim
Norwegian University of
Science and Technology

Appliance simulation models for the evaluation of energy management policies

Shie Chen

shiec@stud.ntnu.no

Submission date: May 29, 2014
Responsible professor: Finn Arve Aagesen, finnarve@item.ntnu.no
Supervisor: Kornschnok Dittawit, kornschd@item.ntnu.no

Norwegian University of Science and Technology
Department of Telematics

Problem Description

A home energy management system is a system in a house that assists consumers in monitoring and optimizing electricity usage in order to lower electricity costs while maintaining consumers' comfort. One approach to achieve this is by using energy management policies to schedule appliance activities at the appropriate time. A policy is a set of rules defining the events and the corresponding actions to reach goals. Simulations are needed to validate the policies. The SMASH (SiMulated Adaptable Smart Home) simulation platform has been developed at the Department of Telematics at NTNU. It is a model of the real adaptable smart home. It consists of entity energy models for various physical appliances. There are different types of simulation results such as cumulative energy consumption, cumulative electricity costs, and peak power. The feasibility and effectiveness of the policies can be validated.

The platform developed so far only contains two energy entity models. Additional models and policies are needed to simulate the use of many policies targeting different appliances simultaneously.

This project aims at developing at least two entity energy models and policies making use of the models. The models must include detailed energy consumption estimation during all the defined operations of the appliances. The tasks within this project includes 1) study how the platform works, 2) design entity energy models compatible with the platform, 3) create policies that use the designed models, 4) create case studies that combine the models and the policies, 5) implement the case studies and collect results, 6) analyze the results and evaluate the policies.

Abstract

With the increasing concerns in home energy conservation and user comfort assurance, researchers are developing methods and tools to simulate and reduce the energy consumptions in households while keeping the user comfort above certain level.

SMASH stands for SiMulated Adaptable Smart Home. It is a simulation platform developed at the Department of Telematics at NTNU that resembles the real adaptable smart home. It is used as a tool for validating and evaluating different energy management policies applied to home appliance models. The platform developed so far only contains a space heating energy model and a water heater energy model. Additional models and policies are demanded to further complete the platform.

This project aims at developing two entity energy models and policies that should be applied to the energy models. The models must include detailed energy consumption estimation during all the defined operations of the appliances.

The project consists of five tasks below:
RT1: Literature review of the platform under development, and syntax study of the policy.
RT2: Design of the lighting system and washing machine energy models compatible with the current platform.
RT3: Design of the high price response policies for the above two energy models.
RT4: Implementation of the energy models and the policies.
RT5: Test case study of the implemented energy models and policies, and evaluation of the results.

The main contributions of this project are:
C1: A lighting system energy model and a washing machine energy model compatible with the simulation platform under development.
C2: High price response policy for both energy models.

Acknowledgements

I would like to thank my supervisor Kornschnok Dittawit, for her patience when I came to her for help, and all the practical help she gave to me. Finally I am grateful for my friends Qi Wei and Ting Liu, for insightful discussions and their support.

Table of Contents

Problem Description ..... **i**
Abstract ..... **i**
Acknowledgements ..... **ii**
List of Pictures ..... **v**
List of Tables ..... **v**
List of Code Snippets ..... **v**
List of Acronyms ..... **vi**
1 Introduction **1**
1.1 Problem Outline 1
1.2 Objectives 2
1.3 Research Methodology 2
1.4 Contributions 3
1.5 Report Structure 3
2 Background **4**
2.1 Existing Simulation tools 4
2.1.1 OpenDSS 4
2.1.2 GridLAB-D 5
2.1.3 EnergyPlus 5
2.1.4 TRNSYS 6
3 SMASH simulation platform **7**
3.1 SMASH functional architecture 7
3.1.1 Energy management policy 8
3.1.2 Existing models and policies 9
4 Models and Policies Design **10**
4.1 Entity energy models 10
4.1.1 Lighting system energy model 10
4.1.2 Washing machine energy model 12
4.2 High price response policy 13
4.2.1 Lighting system 14
4.2.2 Washing machine 16
5 Models and Policies Implementation **18**

5.1 Entity energy models 18
5.2 High price response policy 21
5.2.1 Lighting system 21
5.2.2 Washing machine 21
6 Test Results 24
6.1 Lighting system testing 24
6.1.1 Test Case 1 25
6.1.2 Test Case 2 26
6.2 Washing machine testing 28
6.2.1 Test Case 1 28
6.2.2 Test Case 2 31
6.2.3 Test Case 3 32
7 Evaluation and Discussion 33
7.1 Evaluation of the process 33
7.2 Evaluation of contributions 33
7.3 Evaluation of limitations 34
7.4 Reflections 34
8 Conclusion 35
8.1 Contributions 35
8.2 Future work 35
References 37
Appendix I: Configuration of data illustration 38

List of Pictures

Figure 2-1: DSS structure [6] 4
Figure 2-2: Integration of elements of a building energy simulation [9] 5
Figure 2-3: Example project in simulation studio [11] 6
Figure 3-1: Functional architecture of the real system vs. SMASH 38 [12] 7
Figure 3-2: Policy and goal ontology model [14] 8
Figure 5-1: Class diagram for the implementation of energy models 18
Figure 5-2: Policy logic for the lighting system model 21
Figure 5-3: Policy logic for the washing machine model 22
Figure 6-1: Hourly price and price ceiling on 1st January 2010 25
Figure 6-2: Illuminance in the living room on 1st January 2010 25
Figure 6-3: Accumulated electricity cost in Jan. 2010 (Test case 1) 26
Figure 6-4: Accumulated electricity cost in Jan. 2010 (Test case 2) 27
Figure 6-5: Prices between target hour and latest finish time 29
Figure 6-6: Electricity cost for all working days in January 2010 (Test case 1) 30
Figure 6-7: Accumulated electricity cost of the washing machine (Test case 1) 30
Figure 6-8: Accumulated electricity cost of the washing machine (Test case 2) 31
Figure 6-9: Accumulated electricity cost of the washing machine (Test case 3) 32

List of Tables

Table 4-1: Terms/symbols used in (2) 11
Table 4-2: Light distribution in the house 11
Table 4-3: Program configurations 12
Table 4-4: Policy details for lighting system 14
Table 4-5: Policy details for washing machine 16
Table 5-1: Data calculation in both energy models 20
Table 6-1: Light plans in each room 24
Table 6-2: Illuminance and light statistics in the living room 26
Table 6-3: Illuminance and light statistics in the house at high price hours 27
Table 6-4: Program power settings of the washing machine 28
Table 6-5: Program and time configuration in test case 1 28
Table 6-6: Adjusted schedules for all working days in January 2010 29

List of Code Snippets

Code Snippet 3-1: XML rule structure 8
Code Snippet 4-1: Action structure in the policy 13
Code Snippet 5-1: Actions for the washing machine 23

List of Acronyms

COM: Component Object Model
CSS: Consumer Support System
CU: Coefficient utilization
DG: Distributed Generation
DLL: Dynamic Link Library
DSO: Distribution System Operators
DSS: Distribution System Simulator
ESP: Energy Service Providers
HEMS: Home Energy Management System
HVAC: Heating, Ventilation and Air Conditioning
LLF: Light Loss Factor
NxET: Native XML Equivalent Transformation
RMS: Root Mean Square
SMASH: SiMulated Adapatable Smart Home
XET: XML-based Equivalent Transformation
XML: Extensible Markup Language

1 Introduction

Both efficient energy consumption and preserved user comfort are goals of an adaptable smart home. A home energy management system (HEMS) is the system in a smart home that assists consumers in monitoring and optimizing electricity usage in order to lower electricity costs while maintaining consumers' comfort. One approach to achieve this is by using energy management policies for home appliances in the system to schedule appliance activities at appropriate time. Entity energy models simulate real home appliances such as a heater or a washing machine. A policy defines rules for an energy model about how to react upon certain events in order to save energy. The system can dynamically adjust the status of the home appliances when certain policy is triggered.

1.1 Problem Outline

The SMASH (SiMulated Adaptable Smart Home) simulation platform developed at the Department of Telematics at NTNU provides the capability to simulate energy consumptions of home appliances and the capability to validate and evaluate the energy management policies for home appliances. Different energy related data such as total energy consumptions and total cost can be collected and used to evaluate the feasibility of the energy models and the efficiency of the policies.

The platform currently supports simulations for a space heating energy model and a water heater energy model. A high price response policy and a power reduction request policy have been developed for the existing energy models. A high price response policy attempts to adjust the heating levels in all rooms in response to high prices. DSO can send power reduction request to reduce certain amount of power during control periods. The power reduction request policy then decides whether to comply with the request or reject it and generate actions accordingly. More models are required to expand the platform. As a result, more energy management policies suited to the energy models are also required.

The models must include detailed energy consumption estimation during all defined operations of the appliances. This is used to simulate the energy behaviors and to calculate the energy cost of all operations, so that the effectiveness of the policy can be evaluated by comparing the costs of different energy management actions. The energy models and the policies to be developed must be compatible with the existing platform, which requires the syntax of all the design to be consistent with the existing models and policies.

1.2 Objectives

In this project, the objective is to develop two entity energy models and the corresponding policies targeting these models. In order to generate realistic and effective models and policies to be added to the existing platform, there are several tasks to be done.

First of all, the existing platform should be fully studied. This gives a thorough understanding of how the platform works and how the models and policies should be structured.

Second, two home appliances must be chosen for the development of their corresponding energy models. Their operations and energy consumption estimation methods should be determined. Then the required energy models should be realized by following the structure and syntax of the existing models.

The energy management policies shall define rules needed to arrive at a final decision on how the schedules of the energy models should be adjusted to fulfill the goal of the policy. This project focuses on high price response policies for both energy models.

Finally, after the implementation of the above models and policies, test cases should be defined to collect data and evaluate the policies by analyzing and comparing the data.

1.3 Research Methodology

The development process of this project follows the agile software development methodology, which is a widely used methodology for software development. It is a rapid software development approach with a set of best practices, which iteratively completes the processes in the software development lifecycle (analysis, design, implementation, testing, deployment and maintenance). Agile development focuses on the source code rather than the analyses/design, and it is intended to deliver working software quickly within restricted time. So it is suited to this project, which is basically an implementation of two energy models and two policies, and which needs to be done within a five-month period.

The following chapters of this report illustrate three processes in the software development lifecycle (design, implementation and testing) in sequence for the ease of explanation, but they are actually completed in iteration.

1.4 Contributions

After a 5-month work on this project, all defined tasks have been completed. The contributions of this project can be summarized as the following:

- Two working entity energy models (lighting system energy model and washing machine energy model) were designed and implemented into the existing SMASH simulation platform.
- A high price response policy for each energy model was designed and implemented into the existing SMASH simulation platform. The feasibility and efficiency of the policies were successfully validated and evaluated through the test cases.

1.5 Report Structure

The remaining parts of this report are structured as follows.

Chapter 2 explains the background of this project. Chapter 3 illustrates the platform the project is based on in detail. Chapter 4, 5 and 6 provide details on the design, implementation, testing and results respectively. Chapter 7 provides the evaluation and discussion of the project. Finally, Chapter 8 concludes.

2 Background

There have been many research activities related to HEMS. AIM, for example, is a novel architecture for modelling, virtualizing and managing the energy consumption of household appliances [1], which is a project under the Seventh Framwork Programme (FP7). Beywatch [2] and Smart House/Smart Grid (SH/SG) [3] are two other projects under FP7 aiming at energy efficiency. [4] describes a real-time energy control approach for smart HEMS. Different energy management approaches were proposed as part of these projects. There are also other energy management approaches.

However, there is a need to evaluate energy management approaches before they can be used in practice. Thus a simulation platform is needed to estimate the results and validate the approaches. There exist many simulations tools such as OpenDSS and GridLAB-D. There are also simulation tools suitable for HVAC (Heating, Ventilation and Air Conditioning), such as EnergyPlus and TRNSYS.

2.1 Existing Simulation tools

2.1.1 OpenDSS

OpenDSS is an electric power Distribution System Simulator (DSS) for supporting distributed resource integration and grid modernization efforts [5].

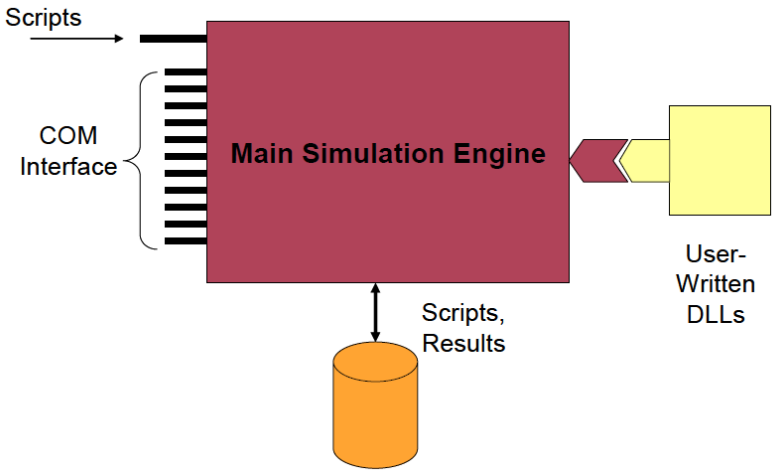


Figure 2-1: DSS structure [6]

It is an open source project implemented as both a standalone executable program and a COM (Component Object Model) DLL (Dynamic Link Library) designed to be executed on a variety of existing software platforms [6]. The DSS program supports all RMS (Root Mean Square) steady-state (i.e. power flow and harmonics in frequency domain). Many of the features in the program were intended to support

distributed generation (DG) analysis, but many new types of analyses are designed to meet future needs.

Figure 2-1 demonstrates the DSS structure. The main simulation engine is responsible for creating electric power distribution systems models and performing related analysis. Other external solution modes can be added and make use of the simulator through the COM interface. Users can use the program by developing user-written DLLs and plugging it into the container. The program is script-driven, and it allows the program to solve the limited-dialog issue in some distribution system analysis tools.

2.1.2 GridLAB-D

GridLAB-D is a power distribution system simulation and analysis tool [7]. The core of GridLAB-D provides an advanced algorithm that simultaneously coordinates the state of many independent devices. It also examines the interaction between different parts of a distributed system. It can be used to design more effective and efficient load control programs.

2.1.3 EnergyPlus

EnergyPlus is an energy analysis and thermal load simulation program [8]. The user describes the building's physical make-up and associated mechanical systems, and the program calculates the energy necessary for heating and cooling a building using a variety of systems and energy sources. It assists HVAC and building designers in estimating real-world effects from a particular building parameter configuration, heating and cooling system size and placement, etc.

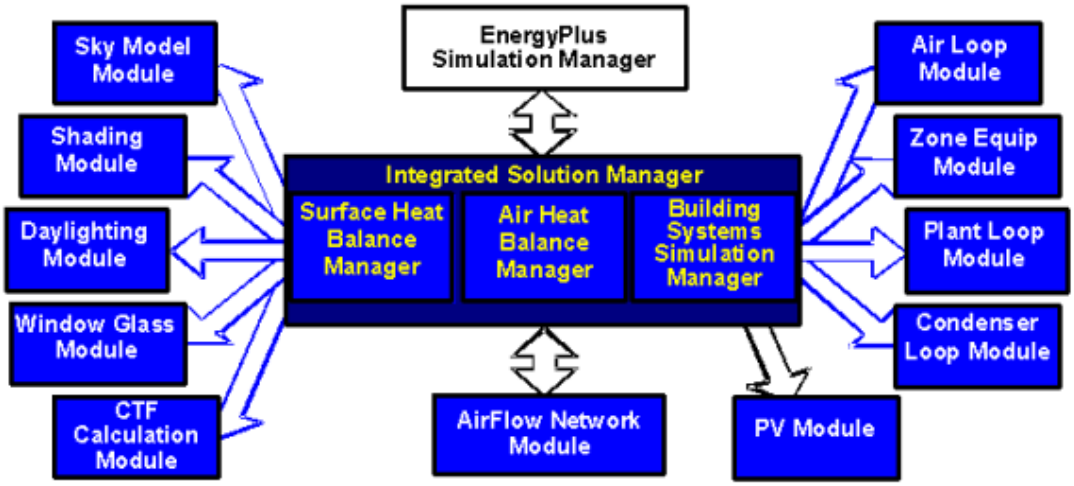


Figure 2-2: Integration of elements of a building energy simulation [9]

Figure 2-2 shows a basic overview of the integration of important elements of a building energy simulation. The modularity of EnergyPlus makes it easier to add other

component simulation modules. EnergyPlus also integrates the three aspects of a simulation, which are loads, systems and plants.

2.1.4 TRNSYS

TRNSYS is a TRaNsient Systems Simulation program with a modular structure [10]. Similar to EnergyPlus, the user specifies the components in the system and in addition specifies the manner in which they are connected by using a system description language recognized by TRNSYS. The TRNSYS program has a library which includes the commonly used components in thermal and electrical energy system. Typical applications can be solar systems, HVAC systems and renewable energy system. The TRNSYS simulation studio helps the user to connect the components in a simple and easy fashion. An example project in the studio looks like the following figure:

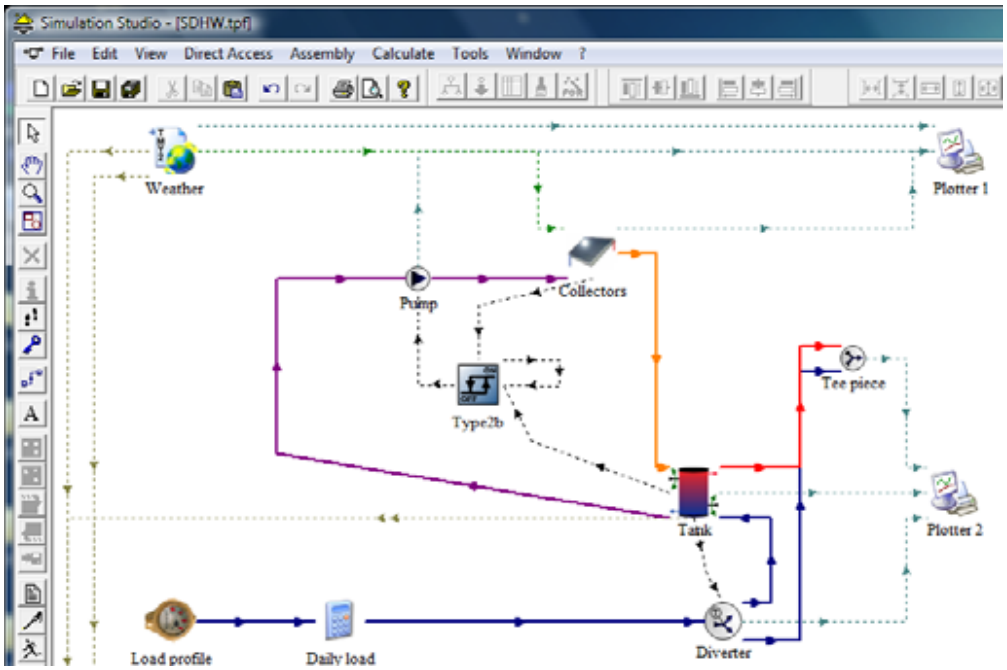


Figure 2-3: Example project in simulation studio [11]

Both OpenDSS and GridLAB-D, together with some other simulation tools, are intended for the simulation of the complex power system and does not focus on what happens inside a house and how consumers react to events. The EnergyPlus and TrnSys tools are well suited for simulation of the load on a system, but they do not provide a means to test energy management approaches. Thus a simulation platform called SMASH was developed at NTNU to address these drawbacks. The platform can simulate the energy consumption both with and without energy management policies. Policy designers can use the platform to assist in the design of energy management policies by analyzing simulation results on cost reduction, peak load reduction and comfort reduction, which provides an indication on the effectiveness of the policies. The next chapter describes the platform in more details.

3 SMASH simulation platform

This project is designed to expand the functionality of the SMASH simulation platform and is mainly based on the existing platform.

3.1 SMASH functional architecture

As mentioned earlier, this platform is used to validate and evaluating the energy management policies for home appliances in a smart home. Figure 3-1 shows the 3-layer architecture of both the real smart home system and the SMASH platform.

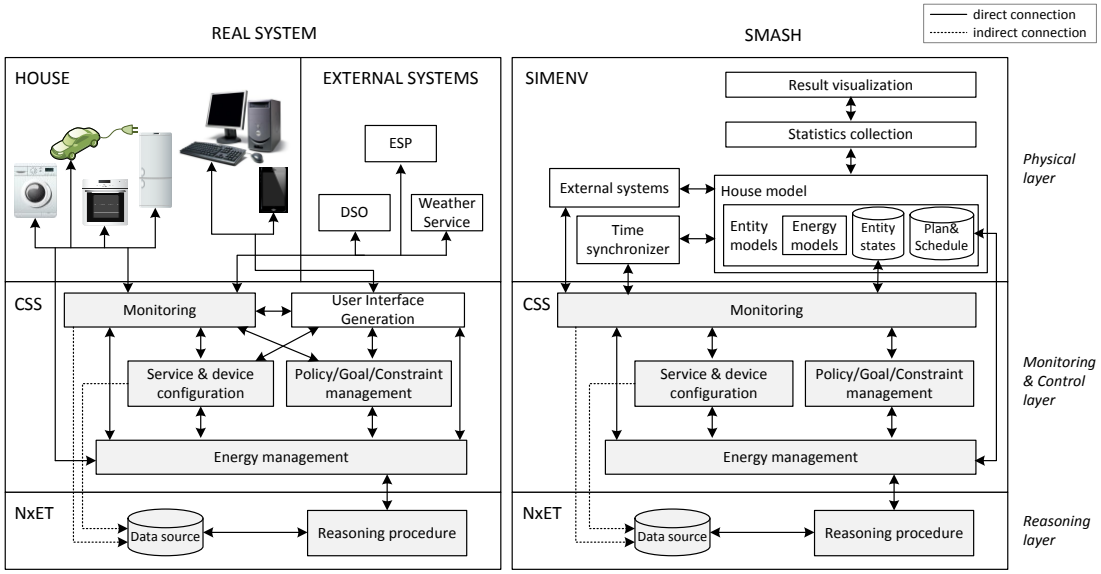


Figure 3-1: Functional architecture of the real system vs. SMASH 37 [12]

The following part gives an overview of every layer in SMASH functional architecture and the content of the existing platform. More detailed description can be found in [12].

SimEnv is a simulation environment that simulates the house and the home appliances in the real system. It is the place where the entity models are developed and where the simulation results are collected.

CSS stands for Consumer Support System, and its main responsibility is to generate actions used to decrease electricity cost while maintaining user comfort by running energy management policies. CSS retrieves entity data from SimEnv and generates actions that are applied to the entity models to adjust the entity schedule. As seen in the figure, User Interface Generation functional component was not included in the SMASH compared to in the real system.

NxET stands for Native XML Equivalent Transformation. It is an XML-based implementation of the Equivalent Transformation paradigm written in Java [13]. It is

a policy-based reasoning machine that processes energy management policies to reach a control decision used by CSS. [13] explains the NxET system in detail.

3.1.1 Energy management policy

An energy management policy is a set of rules defined and processed to generate actions aiming at reducing electricity cost. Rules in a policy consist of constraints and variables. Constraints set limit to the possible actions and they are set by users. Variables are relevant data used in the process of choosing actions.

The variables in the platform can be categorized into two types, dynamic variables and decision variables. A decision variable is what needs to be determined by the reasoning machine, and a dynamic variable is data that changes as a function of time and may be beyond the control scope of the reasoning machine [14]. The electricity price is an example dynamic variable, and the new status of an entity is an example decision variable. Dynamic variables have influence on decision variables.

A goal is associated with a policy and it defines the target to achieve. Actions are operational instructions which are executed on the entity models. Figure 3-2 illustrates an ontology model for the above concepts. Refer to [14] for more details.

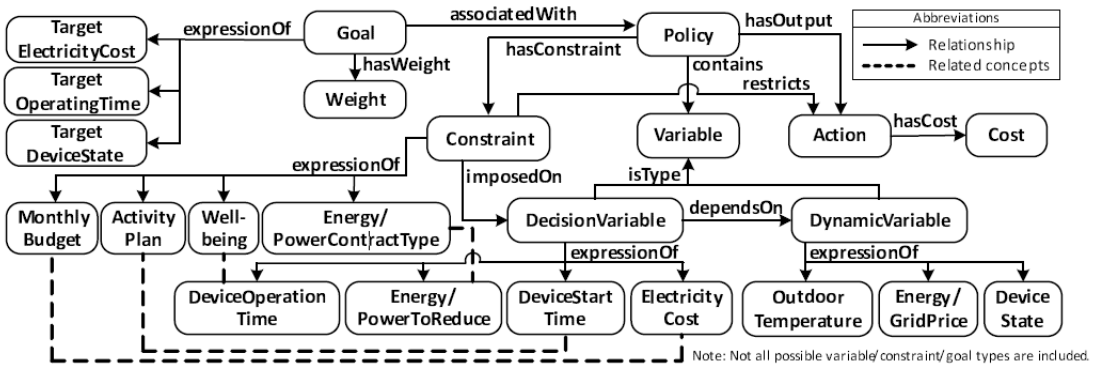


Figure 3-2: Policy and goal ontology model [14]

The policy is XML-based, and it complies with the XML-based data model in the NxET system (see [13] for more details). Every rule in the policy has a head, an optional condition and a body. Code Snippet 3-1 shows the abstract structure of a XET rule in the policy.

Code Snippet 3-1: XML rule structure

```
<xet:Rule xet:name='…' xet:priority='…' xet:class='…'>
    <xet:Meta>...</xet:Meta>
    <xet:Head>...</xet:Head>
    <xet:Condition>...</xet:Condition>
    <xet:Body>...</xet:Body>
</xet:Rule>
```

The head is used to match either the query triggered by an event or a sub-rule required when a certain rule is being processed. The condition determines when the body of the rule is processed. It is comprised of one or more XML expressions. The body contains XML expressions to be processed to reach a decision. The expression with the “xet” prefix is recognized by the NxET reasoning machine as an ET transformation clause that will be processed according to the usage of the system.

3.1.2 Existing models and policies

There are currently two entity energy models in the SMASH platform, the space heating energy model and the water heater (with tank) energy model. Both are normal home appliances in Norway.

The space heating energy model is both an energy and thermal model. It estimates the energy consumption used to heat up the room and it simulates the fluctuated temperature in the room at the same time. The model makes use of a set of parameters whose values are dependent on the layout of the rooms in the house and the materials used to build the house. This model is used in the SimEnv layer to simulate the effect of a space heater and is also used by CSS to get energy estimation data during the process of the policy.

The water heater energy model estimates the energy consumption used to heat up water in the tank and also simulates the fluctuation of water temperature in the tank. This model is only used in SimEnv to simulate the effect of the heater. It is not used as part of the policy processing because consumer plan on the water demand is usually not known.

A high price response policy and a power reduction request response policy are present in the platform.

The high price response policy generates actions that adjust the heating levels (room temperature) in the house in response to high prices. The policy is triggered at the start of the day and uses day-ahead prices to estimate the electricity cost.

The power reduction request response policy is triggered when a power reduction request is received. The Distribution System Operators (DSO) sends this request one hour before the target hour. The request can be either accepted or rejected by the reasoning machine. In case it is accepted, the power reduction requirement will be fulfilled by turning off the heating system.

The energy models and energy management policies described above have included the two of the most common used home appliances in Norway. The results of the policies reduced the electricity cost while maintaining the user comfort.

4 Models and Policies Design

This chapter describes the design of the two chosen energy models and the corresponding high price response policies.

4.1 Entity energy models

The SMASH platform should resemble a real house as much as possible, so the chosen home appliances should be common devices that are used in almost every house. Under this consideration, the lighting system and the washing machine have been chosen.

During the simulation process, several categories of data are collected and stored in the SimEnv, which are illustrated later to show the behavior of the entity models.

4.1.1 Lighting system energy model

Every house has a lighting system, and different rooms may use different lights. The typical types of light used in a house are incandescent and fluorescent lights. Depending on the users' choice, different lights can be used and it is hard to determine which should be used in this project. However, a light is characterized by its physical specifications, and only these specifications are relevant in this project. The specifications that are used in this project are lumens and rated power. The term light and lamp are used alternately in the rest of this report.

The calculation of energy consumed by all the lights is necessary to calculate the electricity cost. The energy consumption of one light is calculated by using the following formula:

$$E = P \times t \tag{1}$$

where E represents energy in kWh, P represents the power specification of the light in kW and t represents the total time in hour that the light is on.

This energy model is also used for calculating the illuminance in each room. Lumen method is a commonly used technique to calculate the average maintained illuminance across a working plane. In photometry (optics), Illuminance is term representing the amount of luminous flux per unit area measured in lux. Lumen is a SI derived unit representing the amount of light emitted in a unit solid angle of one steradian from a uniform source of one candela. Usually the lumen value will be provided as a physical specification of the light. The lumen method uses the following formula to calculate the average maintained illuminance of a room.

$$\bar{E}_{maintained} = \frac{total\ lamp\ lumens \times CU \times LLF}{workplane\ area} \quad [15] \qquad (2)$$

Table 4-1: Terms/symbols used in (2)

Terms/Symbols	Meanings
$\bar{E}_{maintained}$	The average maintained illuminance of the room
Total lamp lumens	The sum of lumen values of all the lamps in a specific room
CU	Coefficient of utilization. The ratio of lumens received on the work plane to the total lumens emitted by the lamps alone
LLF	Light loss factor. The ratio of illuminance for a given area to the value that would occur if lamps operated at their (initial) rated lumens and if no system variation or depreciation had occurred [15]
Workplane area	A workplane is assumed as the horizontal plane 0.76 meter above the floor, where a visual task is usually done. The area is calculated as the multiplication of the length and width of the plane

Both CU and LLF are affected by several factors, such as the reflectance of room surface overtime, the lifetime of the lamps and the atmospheric condition in the room. There are formulas to determine the values of these two parameters. However, in order to simplify the configuration in this project, fixed values are used.

The lumen method assumes that the lamps in the room are uniformly distributed to obtain reasonable results. This assumption will also be used in this project.

Table 4-2: Light distribution in the house

Room (length × width)	No. of lights	Lumen/ *lm*	CU	LLF	Average illuminance/ *lux*	Recommended illuminance/*lux*
Living room 4×4 m^2	4	400	0.7	0.8	56	50
Kitchen 4×3 m^2	4	1000	0.7	0.8	186.7	100-300
Bathroom 1 5×2.75 m^2	2	2000	0.7	0.8	162.9	150
Bathroom 2 4×2.5 m^2	2	1500	0.7	0.8	168	
Bedroom 1 6×3 m^2	4	500	0.7	0.8	62.2	50
Bedroom 2 4.5×3 m^2	4	400	0.7	0.8	66.4	
Remarks	Length and width of the rooms are measured in meters (*m*). Lumen uses the symbol *lm*. Illuminance uses the unit symbol *lux*.					

One energy model takes care of one light, so in the same room, there are several energy models being simulated. The distribution of the lights in the house is shown in Table 4-2. The average illuminance is calculated according to (2).

In order to make the design consistent with the existing platform, the number of rooms and the length and width in every room are retrieved from the entity specification for the house.

Similar to the existing energy models in the platform, the lighting system uses an hourly plan for the lights in a day. The reason is that the lighting system is used regularly every day in a house. So its plan is predictable. All lights in the same room use the same plan initially, and different rooms can use different plans. Later when the policy is triggered, the individual light plan may be changed. The hourly value in the plan indicates the status of the light, either on or off.

4.1.2 Washing machine energy model

A washing machine is a common used home appliance. It can often be programmed to wash different types of clothes, which may result in different washing duration and water temperatures.

In this project, a Whirlpool AWO/D 8001 washing machine is chosen as the prototype of the washing machine energy model. Since the user manual does not specify the processing time and rated power required for each process in a program, which are main parameters in the energy model, a power meter was used to manually measure the time and power consumed. Two programs were measured as sample programs in the energy model being developed. The settings of the programs and the time and power specification are illustrated in Table 4-3.

Table 4-3: Program configurations

Programs	Settings	Time and power specifications					
			Water intake	Heating	Rinse 1	Rinse 2	Spin
Synthetics	T: 40℃ Vs: 1400*rpm* Dirtiness: medium	Time/*min*	3	12	32	23	10
		Power/*W*	4-48	48-2035	3-170	3-167	4-472
Wool	T: 40℃ Vs: 1000*rpm* Dirtiness: minimum	Time/*min*	3	20	7	7	3
		Power/*W*	10-48	387-1798	3-48	3-29	2-455
Remarks	T: temperature, measured in Celsius(℃) V_s: spin speed, measure in round per minute(*rpm*) Time: measured in minutes(*min*) Power: measured in watt(*W*)						

In reality, one process may contain several sub-processes. For example, there may be a ‘drain water’ sub-process and a ‘water intake’ sub-process during the rinse process. And the sub-processes are executed in an iterative order. The separation of processes shown in Table 4-3 is simplified. As a consequence of this simplification, the fluctuations of power for different sub-processes are combined into the range shown in the above table. The limitation is that the distribution of power for a process is assumed to be uniform, which means the values within the power range have the same possibility to be used in the simulation process. But in reality the sub-processes make it non-uniform, the power changes dynamically.

Unlike the plan for the existing energy models, the plan for the washing machine is designed to be a weekly plan. It is unlikely that a washing machine is used regularly every day in a residential house. Moreover, the operation time for one wash is limited, normally less than two hours, so an hourly plan is unnecessary. The seven values in the plan indicate whether the washing machine will be working or not. If yes, then it specifies the start time. It is also assumed that a washing machine will only work once per day.

For both of the energy models being developed in this project, there is both a plan and an actual schedule. The plan is set by the user before the simulation starts, and the schedule is assumed to be the same as the plan. But when the policy is applied and executed, the schedule can be changed to adjust the behavior of the entity during the simulation, so that the goal of reducing electricity cost can be achieved.

4.2 High price response policy

This policy was designed to adjust entity schedules in response to high prices. The expected result of a policy is a set of actions that sets the values of the decision variables related to the schedule such as a new start and end time of the activity. A skeleton of one action set is shown as following:

Code Snippet 4-1: Action structure in the policy

```
<smash:ActionSet smash:set="…" smash:key1="…" smash:key2="…">
    <smash:Action smash:type="…">
      <smash:variable1>...</smash:variable1>
      <smash:variable2>...</smash:variable2>
      <smash:variable3>...</smash:variable3>
      ...
    </smash:Action>
    ...
  </smash:ActionSet>
```

The decision variable types depend on the nature of the appliance. For example, in case of a lighting system, the variables are the ID of the target light, the new status of the light and the duration of the adjustment.

As illustrated in Figure 3-2, a policy contains goals, variables and constraints. Same as in the existing high price response policy, the policies being developed in this project will use cost and comfort as the two goals. For both goals, a 'comfort' cost and a 'cost' cost will be calculated and evaluated during the processing of the policy to determine the resulting actions. A 'cost' cost is the reduction of electricity cost, and a 'comfort' cost is dependent on the type of entity.

4.2.1 Lighting system

The policy details for the lighting system are shown in Table 4-4. The policy is triggered at the start of the day, and the schedules of lights to turn off are adjusted one time a day (if needed).

Table 4-4: Policy details for lighting system

Triggering condition	At the start of the day
Dynamic variables	Hourly ESP prices ($P_{ESP}(H)$), hourly DSO prices ($P_{DSO}(H)$); H = [0...23]
Decision variables	Set of lights in a set of room to schedule (L), set of target hours to schedule (H_S), new status for the lights being scheduled at target hours ($S(L, H_S)$)
Constraints	Comfort: 1) When the goal is cost, room comfort levels σ(R), at target hours must be above unacceptable levels ($\sigma(R) > \sigma_U(R)$); $\sigma_U(R)$ represents unacceptable comfort level for room R. 2) When the goal is comfort, room comfort levels σ(R), at target hours must be above reduced levels ($\sigma(R) > \sigma_R(R)$); $\sigma_R(R)$ represents reduced comfort level for room R. Cost: 1) The estimated 'cost' cost must be less than 0. The 'comfort' cost, which equals the reduction of illuminance in the room, must be smaller than the reduction limits (difference between the initial illuminance and the unacceptable/reduced illuminance level)

Action set	Let:		
	r	represents	room; r ∈ all rooms in the house
	h	represents	target hour; h ∈ H_S
	n	represents	n^{th} light; n ≥ 1
	Φ(r, n)	represents	lumens of n^{th} light in room r
	N(r)	represents	total number of lights in room r
	$N_o(r)$	represents	number of lights to turn off in room r
	E(r)	represents	initial illuminance in room r
	$E_{\sigma R}(r)$	represents	reduced illuminance in room r
	$E_{\sigma U}(r)$	represents	unacceptable illuminance in room r
	S(r, n, h)	represents	new status of n^{th} light in room r
	$P_E(h)$	represents	electricity price ($P_{ESP}(h) + P_{DSO}(h)$) at hour h
	$P_L(r, n)$	represents	rated power of n^{th} light in room r
	∀rh Set S(r, n, h) to *off*; $1 \le n \le N_o(r)$ Costs 'cost': $-calElecCost^1(\{P_L(r, 1),..., P_L(r, N_o(r))\}, P_E(h))$ 'comfort': $reductionOfIllum^2(\{\Phi(r, 1),..., \Phi(r, N(r))\}, N_o(r))$		
	Goal: cost		
	$N_o(r) = numLightOff^3(E(r), E_{\sigma U}(r), \{\Phi(r, 1),..., \Phi(r, N(r))\})$		
	Goal: comfort		
	$N_o(r) = numLightOff^3(E(r), E_{\sigma R}(r), \{\Phi(r, 1),..., \Phi(r, N(r))\})$		

[1] *calElecCost(powerArray, targetPrice)* is a function that calculates the reduction of electricity cost in a room at a target hour. It first adds up the powers of lights to turn off (*powerArray*), and then multiplies it with the duration to turn off (which is the target hour) and the target hour price *targetPrice*. Finally adjust it to the *kWh* magnitude.

[2] *reductionOfIllum(lumensArray, numOff)* is a function that calculates the reduction of illuminance in a room at a target hour by using formula (2). The number of lights to turn off *numOff* and the lumens of every single light *lumensArray* gives the total reduction of lumens, thus gives the total reduction of illuminance.

[3] *numLightOff(initIllum, illumLimit, lumensArray)* is a function that calculates the number of lights in a room to turn off at a target hour by using formula (2). The difference between initial illuminance *initIllum* and unacceptable/reduced illuminance *illumLimit* determines the reduction of total lumens in the room which can be used to determine the number of lights to be turned off.

4.2.2 Washing machine

The policy details for the lighting system are shown in Table 4-5. The policy is triggered one hour ahead of the start time. Since the program will only run for at most two hours a day, it is more appropriate to adjust the program some time before it starts to operate, instead of adjusting one day ahead. This will also reduce the 'comfort' cost by limiting the range to adjust. The 'comfort' cost is here defined as the deviation of the new total wash time from the planned total wash time (program length plus possible pause time).

Table 4-5: Policy details for washing machine

Triggering condition	One hour before the planned start hour
Dynamic variables	Hourly ESP prices ($P_{ESP}(H)$), hourly DSO prices ($P_{DSO}(H)$); H = [0...23]
Decision variables	Washing machine to schedule (M), new start time (T_N), decision on whether to pause the program (P) (optional)
Constraints	Comfort: 1) When the goal is cost, pause action is allowed for only one hour 2) When the goal is comfort, no pause action is allowed 3) The program must finish before certain hour (H_L) on the same day, and the start time and the finish time T_F must fulfill the condition: $T_N \geq H_M$ & $T_F \leq H_L$; H_M represents the triggering hour Cost: 1) The estimated 'cost' cost must be less than 0. When the goal is comfort, the deviation from the planned start time, must not be larger than two hours
Action set	Let: r represents room; r ∈ all rooms in the house p represents decision on whether to pause the program M(r) represents washing machine in room r $t_s(r)$ represents original start time of the machine in room r $t_f(r)$ represents original finish time of the machine in room r $t_n(r)$ represents new start time of the machine in room r $h_l(r)$ represents latest finish time of the machine in room r $h_m(r)$ represents the triggering hour for the machine in room r $A_t(r)$ represents process times of the machine in room r $A_p(r)$ represents process powers of the machine in room r $P_E(H)$ represents electricity prices for hours in the working day

	∀M(r) $(t_n(r), p) = calNewStartTime^1(t_s(r), t_f(r), h_m(r), h_l(r), A_t(r), A_p(r), P_E(H))$ Set new start time to $t_n(r)$; pause the program for one hour if p indicates so Costs 'cost': $calElecCost^2(t_n(r), A_t(r), A_p(r), P_E(H), p) - calElecCost^2(t_s(r), A_t(r), A_p(r), P_E(H), p)$ 'comfort': $\|t_n(r) - t_s(r)\|$ (if paused, + 1) The possible adjustment options depend on the time span of program.
	Goal: cost
	Within the limited time range, when the span of total program length is: 1) one hour Find the hour with lowest price 2) two/three hours Check the set of every two hours (i.e. hour 1&2, hour 2&3...) and the set of every two other hours (i.e. hour 1&3, hour 3&5...), find the combination with lowest electricity cost
	Goal: comfort
	Within the limited time range, either advance or postpone the start time/finish time to integer time point (i.e. postpone start time to 18:00 or finish time to 19:00), find the start time with lowest comfort cost

[1]*calNewStartTime(oriStartTime, oriFinishTime, minHour, finishLimit, timeArray, powerArray, priceArray)* is a function returns the adjusted new start time and the pause indicator. Inside this function, it uses the [2]*calElecCost()* function to calculate electricity cost for every possible adjustment. Then it compares the costs and returns the start time and pause indicator which leads to the least cost. The triggering hour *minHour* and the latest finish time *finishLimit* set the time range to adjust. The planned start time *oriStartTime* and the planned finish time *oriFinishTime* decides the span of program length.

[2]*calElecCost(startTime, timeArray, powerArray, priceArray, pause)* is a function that calculates the electricity cost for the program that starts at *startTime* (and pauses for one hour if *pause* indicates so). As the start time changes, the electricity cost need to be recalculated as a result of changing prices. When the total program time steps across several hours, the electricity cost is divided into several parts accordingly. Thus the sub-process times *timeArray* and powers *powerArray* are passed in as parameters.

5 Models and Policies Implementation

This chapter describes the implementation details of the logics and algorithms in the entity energy models and in the policies. The entities and energy models are implemented in Java, and the policies are structured by the supported data representations in the NxET data model.

5.1 Entity energy models

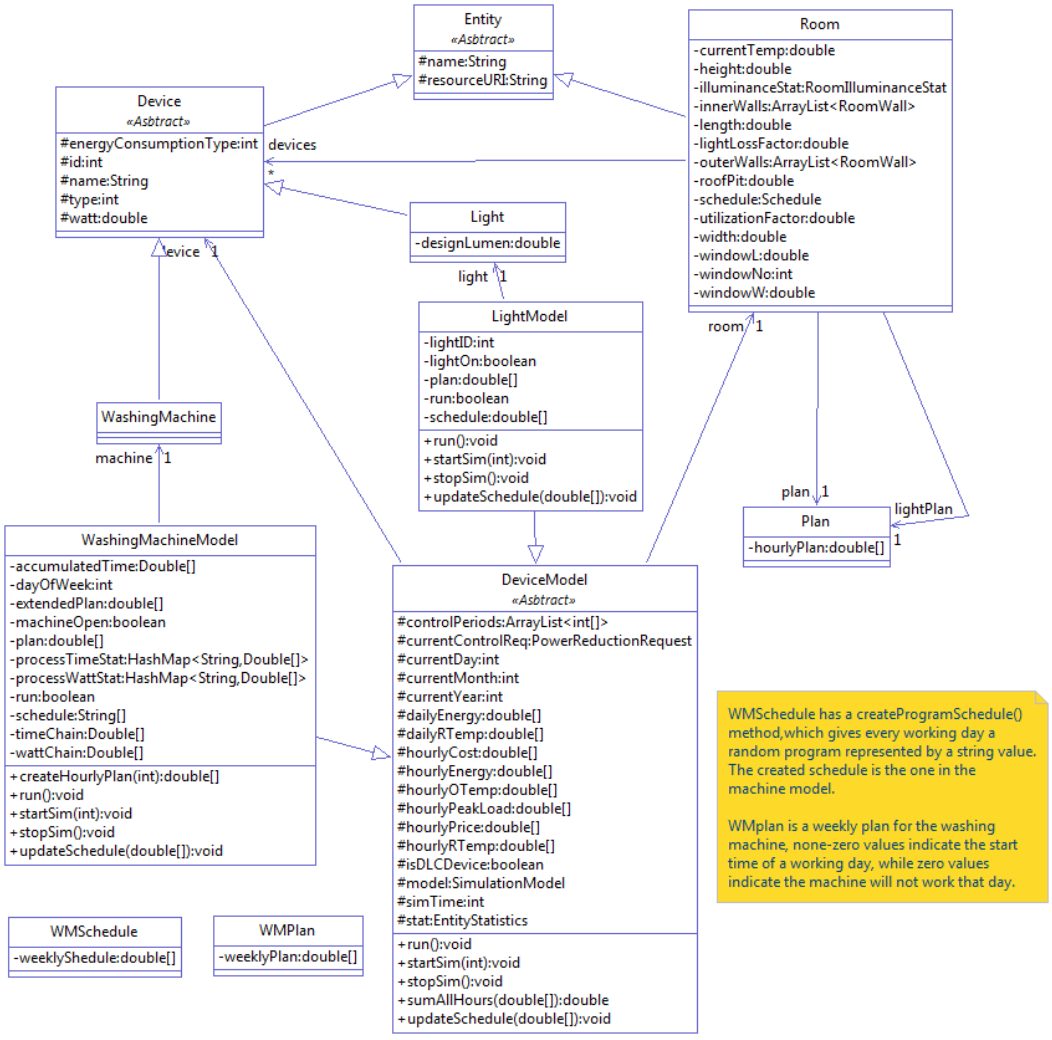


Figure 5-1: Class diagram for the implementation of energy models

Figure 5-1 is a class diagram presenting the relationship among entity energy models and the required entities. The Entity, Device, Room, Plan and DeviceModel classes already exist in the platform. The *utilizationFactor* (CU) and the *lightLossFactor* (LLF) were added into the Room entity as they are dependent on the physical layout

of the room and the lights in that room (although they are set to fixed values in this project). The room has a 24-hour *plan* for the space heater and a 24-hour *lightPlan* for all the lights in the room initially. It also has a set of *devices*.

The Entity class is the parent for Room and Device, and all the concrete device entities extend the Device class. It contains the identity (i.e. *id*) and physical specifications (i.e. *watt*) for the device. In addition to these, the Light class has only a *designLumen* attribute indicating the lumens specification for a light. The washing machine needs to record the time and power consumption for all the sub-processes of each program. This is implemented into a time HashMap [16] and a power HashMap. The key indicates the program, and the values are sub-times and sub-powers respectively.

The DeviceModel is the parent for all energy models. It contains the abstract attributes and methods which record the device status, store energy-related data and maintain the simulation process. It can start and stop the simulation, and it can update the entity schedule, when a policy is applied.

The LightModel class and the WashingMachineModel class are the implementations for the lighting system and the washing machine respectively. Both extend the basic functionality from the DeviceModel, and both override some of the methods to fit into the concrete models.

The plan and schedule used for the washing machine model are different from the ones for the lighting system model (which is a 24-hour plan). The plan is a 7-day plan with its values indicating either not working or the start time on a working day. The schedule is a 7-day schedule with its values indicating either not working or the program on a working day. Depending on the program, the time and power series for sub-processes are different. However, for both models, an extended 288-value plan is created based on the above plans and schedules. The values indicate the status of the device in 24 hours with a resolution of 5 minutes (24*(60/5) = 288). So for the light, if it is on at hour 20, then the value from 241^{th} to 252^{th} in this plan is 1. For the washing machine, if the start time is hour 17 and the total program length is 1.5 hour, then the value from 205^{th} to 222^{th} in this plan is 1.

Every concrete entity energy model holds a reference to one device entity, because it needs to retrieve device specifications for data collection. Data collection is an important function of every energy model. The data collected illustrates the estimated electricity cost of every device with or without policy, and the results are the basis for evaluating the usability and efficiency of the policy.

For both energy models being developed, there are several types of data need to be collected. On the room level, it is the energy consumption, average peak load, plan and schedule. On the household level, it is the energy consumption, average peak load, electricity cost and hourly price. In addition to these, the light system model also records the energy consumption, electricity cost and average peak load on a device

level, because there may be more than one light in a room, and it is reasonable to distinguish among them. For the data calculation, it is done every 5 minutes. The following table shows the algorithms for both models to calculate the data.

Table 5-1: Data calculation in both energy models

Energy models	Algorithms
Lighting system	*energy*[*index*] = *light.getWatt()***lightOnTime*/*3600***0.001*; *peakload*[*index*] = (*lightOnTime* > *300***0.2*?*light.getWatt()*:*0*); *light.getWatt()*: it obtains the Watt of the light referred to by the model;
Washing machine	If the machine is operating during the first sub-process: if(*totalTime* <= *accumTime*[*0*]*60 && *totalTime* > *0*) { *energy*[*index*] = *wattChain*[*0*]**machineOnTime*/*3600***0.001*; *peakload*[*index*] = (*machineOnTime*>*300***0.2*?*wattChain*[*0*]:*0*); } If the machine is operating during the other sub-processes: for(int *i* = *1*; *i* < *timeChain.length*; *i*++) { if(*totalTime* <= *accumTime*[*i*]**60* && *totalTime* > *accumTime*[*i-1*]**60*) { (if the 5-minute period contains 2 sub-processes) if((*totalTime* - *machineOnTime*) < *accumTime*[*i-1*]**60*) { *energy*[*index*] = *wattChain*[*i-1*]*(*accumTime*[*i-1*]**60* - (*totalTime* - *machineOnTime*))/*3600***0.001* + *wattChain*[*i*] * (*totalTime* - *accumTime*[*i-1*]**60*)/*3600***0.001*; *peakload*[*index*] = (*machineOnTime*>*300***0.2*?*Math.max*(*wattChain*[*i-1*], *wattChain*[*i*]):*0*); } (if the 5-minute period only contains one sub-process) else{ *energy*[*index*] = *wattChain*[*i*]**machineOnTime*/*3600***0.001*; *peakload*[*index*] = (*machineOnTime*>*300***0.2*?*wattChain*[*i*]:*0*); }}} *totalTime*: the total time that the program has operated for; *accumTime*: the accumulated time of sub-processes. The current value is the sum of the previous value and the current sub-process time; *timeChain*/*powerChain*: time/power array for sub-processes; *energy*: depending on the time distribution of sub-processes within 5 minutes, it is the energy sum of either one or two sub-processes;
Remarks	0 ≤ *index* ≤ 287; energy measured in kWh, and peak load measured in W; *lightOnTime*/*machineOnTime*: total time that the light/machine is on for within 5 minutes; *peakload*: if the *lightOnTime*/*machineOnTime* is larger than one minute, then the peak load is 1)for the light: the rated power; 2)for the machine: the larger sub-process power;

5.2 High price response policy

The policy is structured by using rules similar to the one in Code Snippet 3-1. Inside the rule, different built-in functions are used to manipulate data variables.

5.2.1 Lighting system

The logic of the policy for a lighting system is shown Figure 5-2.

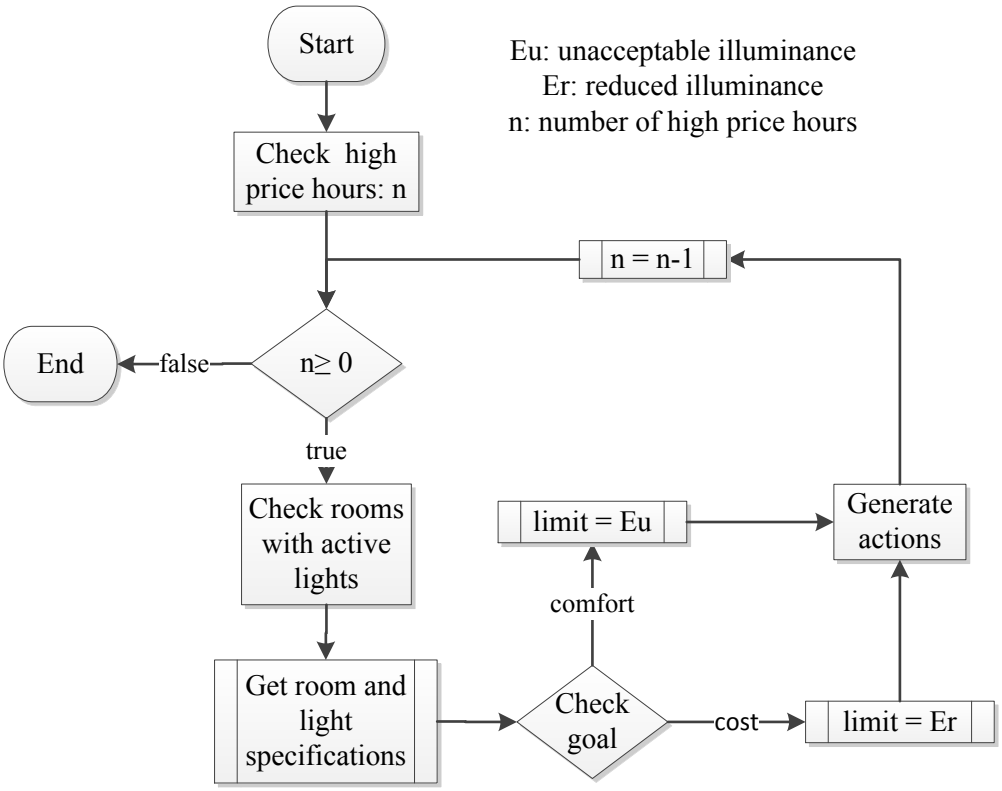


Figure 5-2: Policy logic for the lighting system model

First of all, the high price hours together with their values are retrieved from the DSO. Then for every high price hour, check all rooms with active lights. Active means the lights are added into the simulation environment. In the entity specification of the house, all lights are included, but in the simulation environment, it is possible that not all lights are added. A sub-process is to get the room and light specifications from the entity specification file, which are the illuminance limits of the room, designed lumens and rated power of the lights. After this, for each room, the goal is checked. In case of a cost goal, the unacceptable illuminance is used for the calculation of lights to turn off. Otherwise, the reduced illuminance is used.

5.2.2 Washing machine

The logic of the policy for the washing machine is shown in Figure 5-3.

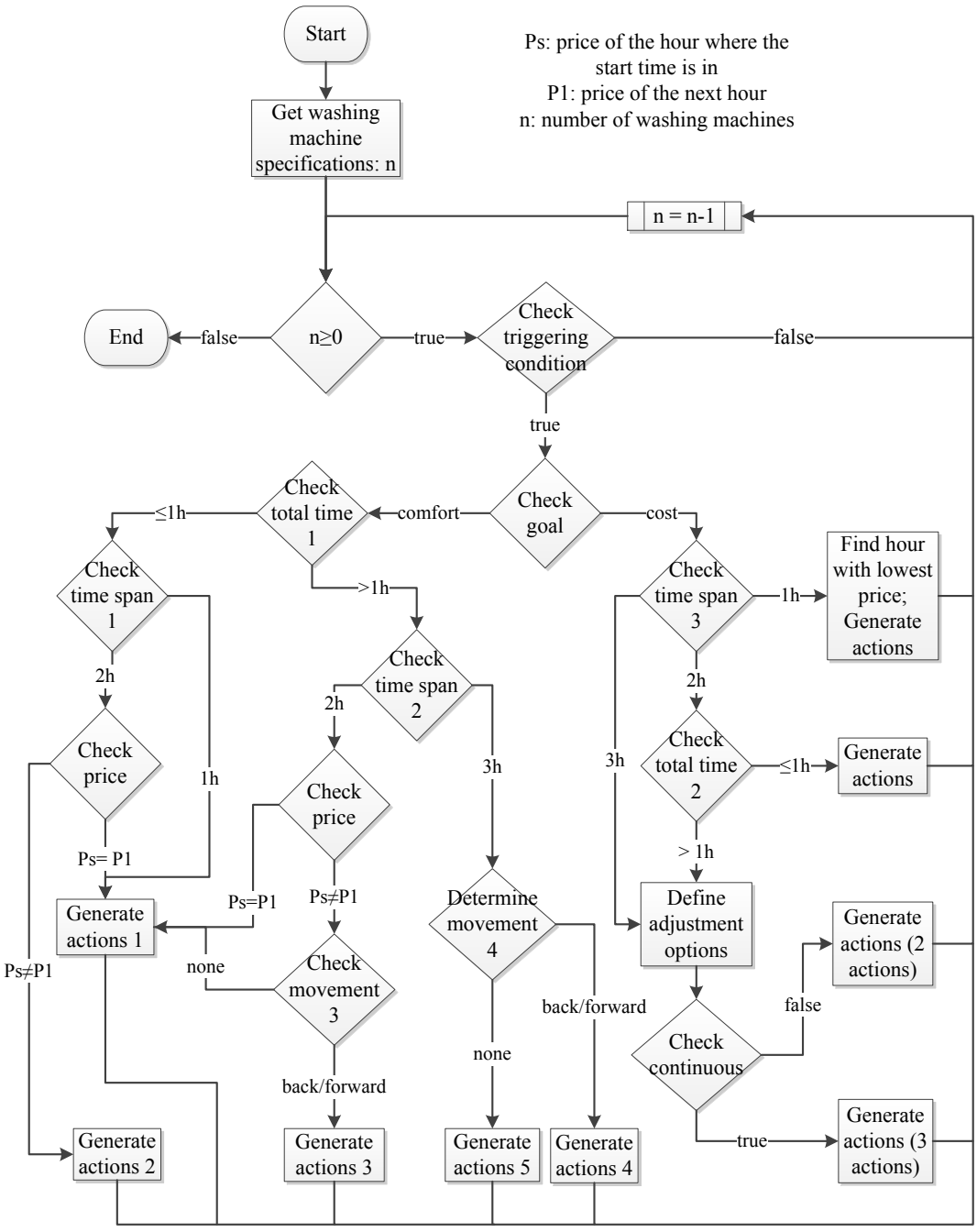


Figure 5-3: Policy logic for the washing machine model

The functions with the same name but different numbers do the same task, but the parameters they need may be different due to the different decision making processes. Thus it is better to differ from these functions.

Due to the implementation of the time trigger and the changing value of start time, it is hard to assign the triggering time to one hour before the start time. So the policy is implemented to take the responsibility for checking the triggering condition for every washing machine, which is one hour before the hour that the start time is within. In the rest of this report, the hour that the start time is within will be called the start hour.

When the triggering condition is fulfilled, the goal is checked. In case of a cost goal, the span of program total time is checked. This time span indicates the number of hours the program goes through and it affects the calculation of electricity cost because the price for each hour may be different.

When the span is one hour, the hour with lowest price in the limited range is found. The limited range is from the triggering hour to the latest finish time of the machine. If the lowest price hour is not the start hour, actions are generated to change the start time to this hour. The most important decision variables in the actions are shown in the following snippet:

Code Snippet 5-1: Actions for the washing machine

```
<smash:ActionSet smash:set="1" smash:key1="Svar_EntityID">
  <smash:Action smash:type="WashingSchedulingAction">
    ...
    <smash:start>Svar_dateString1</smash:start>
    <smash:duration>Svar_duration1</smash:duration>
    <smash:newStatus>Svar_status</smash:newStatus>
    <smash:cost smash:cost_cost="Svar_cost_cost1"
smash:cost_comfort="Svar_cost_comfort1"/>
  </smash:Action>
    ...
  </smash:ActionSet>
```

If the span is two hours, the generated actions are different depending on the total program time. If the total time is less than or equal to one hour, then the actions are similar to the above, except the calculation of original electricity cost is different, because it involves two hours with different prices. If the total time is longer than one hour, then every two hours and every other hour are checked. If the resulting two hours with lowest cost are continuous, then only two actions are generated, one to set the original plan to zero, and the other sets the new plan. Otherwise, the program needs to be paused for one hour, which introduces a third action.

If the goal is comfort, the generated deviation from the original start time will either be no larger than one hour back or less than two hours forward. The deviation is dependent on the estimated cost, which is further dependent on the prices. The total program time and the time span of the program are checked to determine how many hours are involved in the calculation of estimated cost. All the other decision making processes shown in Figure 5-3 will determine whether to move the start time back or forward and how much to move.

6 Test Results

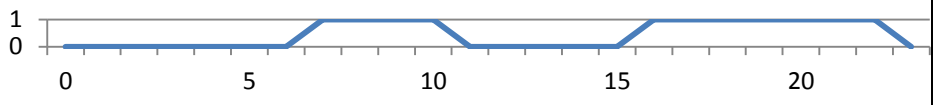
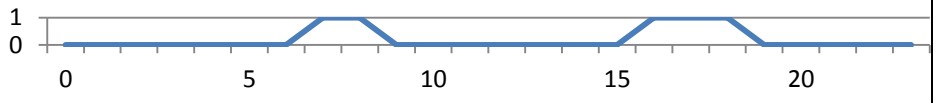
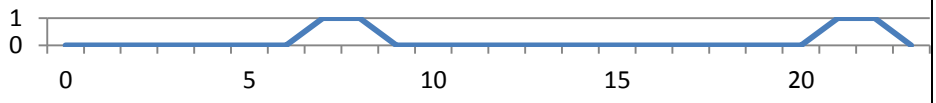
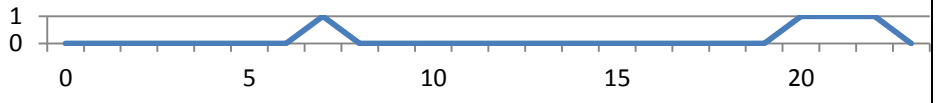
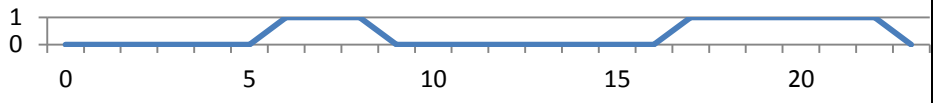
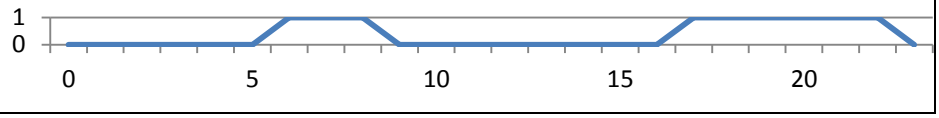
The simulation test cases were run for the duration of one month. The month selected was January 2010. The selection was based on the real ESP prices in Norway where year 2010 was considered a high priced year. The electricity price data used in all simulated cases was constructed using real ESP prices and synthesized DSO prices due to the lack of variable DSO prices in Norway.

For each test case, three simulations were executed. The first one was run without policy and the other two were run with policy and with different goals (cost or comfort).

6.1 Lighting system testing

Each room in the house has an initial light plan, which is applied to all the lights in the same room. The plans used in the simulation are shown in Table 6-1.

Table 6-1: Light plans in each room

Room	Light plan (24 hours)
Living room	
Kitchen	
Bathroom 1	
Bathroom 2	
Bedroom 1	
Bedroom 2	

All lights in the house were designed to have the same Watt specification, which was 30W. So for each hour when the light was on, the energy consumed by one light was 0.03kWh.

6.1.1 Test Case 1

This test case contained the four lights in the living room. For a specific day (1st Jan. as an example), the collected data is shown in the following table and figures. Figure 6-1 illustrates the hourly price on 1st January 2010. As shown in the figure, hour 17, 18 and 19 were high price hours.

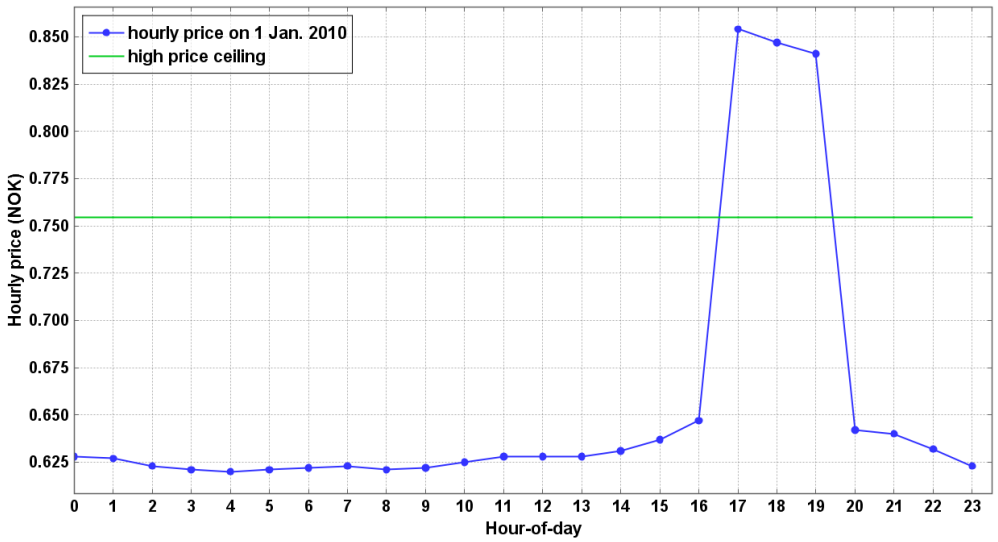


Figure 6-1: Hourly price and price ceiling on 1st January 2010

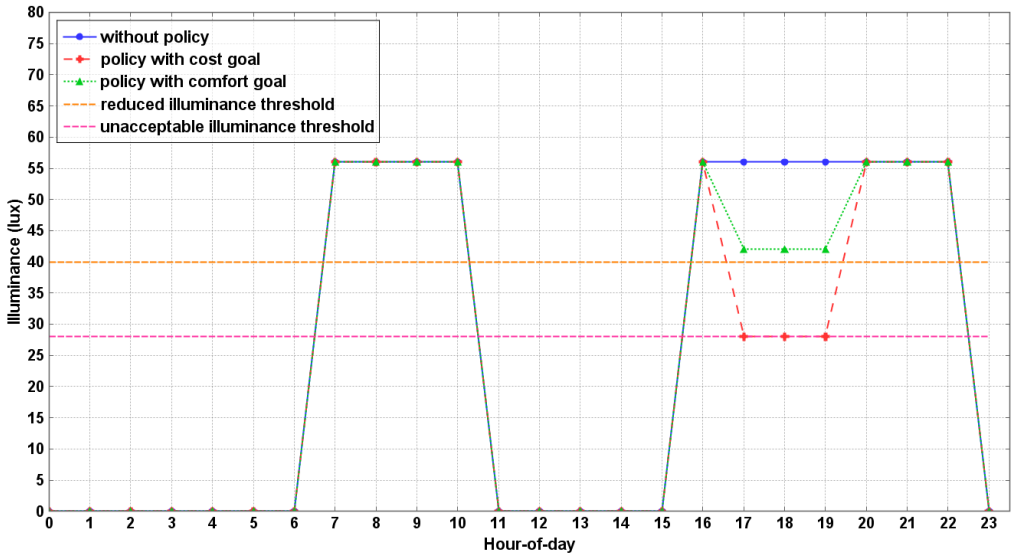


Figure 6-2: Illuminance in the living room on 1st January 2010

When policy was applied, certain lights were turned off at those high price hours. As a result, the illuminance in the living room decreased at high price hours. The decreased illuminance and the lights turned off are shown in Figure 6-2 and Table 6-2.

Table 6-2: Illuminance and light statistics in the living room

Parameters	Without policy	Policy with cost goal	Policy with comfort goal
Illuminance/*lux*	56	28	42
Illuminance limit/*lux*	—	28	40
Lights turned off	—	light 1 & 2	light 1
Lights left	light 1, 2, 3 & 4	light 3 & 4	light 2, 3 & 4

When the goal was cost, half of the lights were turned off at high price hours because the illuminance limit was half of the original. When the goal was comfort, only one light was turned off to maintain user comfort while reduce the electricity cost.

For the whole month, the accumulated electricity cost in NOK is shown in Figure 6-3.

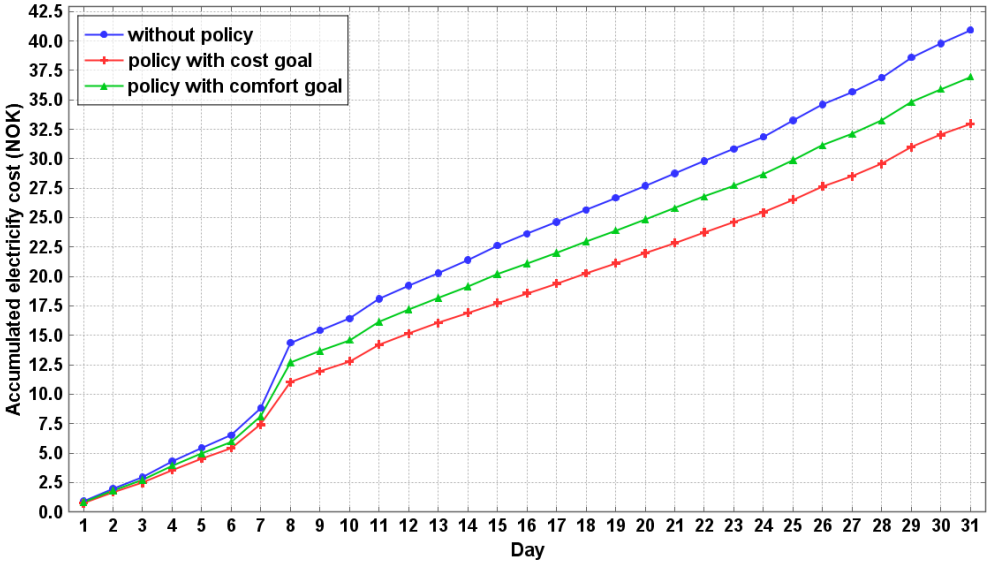


Figure 6-3: Accumulated electricity cost in Jan. 2010 (Test case 1)

As shown in the above figure, the high price response policy reduced the electricity cost in January 2010. When the goal was cost, 7.9486NOK was saved, which was 19.42% of the original cost 40.9319NOK. The 'comfort' cost in this case was a 50% (28*lux*) reduction of illuminance, but it was still acceptable.

When the goal is comfort, 3.9739NOK was saved, which was 9.71% of the original cost. The comfort cost in this case was only a 25% (14*lux*) reduction of illuminance, which was slightly above the reduced illuminance.

6.1.2 Test Case 2

This test case contained all the lights in the house. For every single room, the results were similar to those in the test case 2. At the high price hours, the illuminance and lights statistics are shown in Table 6-3.

Table 6-3: Illuminance and light statistics in the house at high price hours

Room	Light No.	Illuminance limit/*lux*	Illuminance /*lux*			Lights turned off	
			Normal	G1	G2	G1	G2
Living room	1-4	40/28	56	28	42	light 1 & 2	light 1
Kitchen	5-8	131/93	186.7	93.3	140	light 5 & 6	light 5
Bathroom 1	9-10	105/75	162.9	81.5	162.9	light 9	—
Bathroom 2	11-12	105/75	168	84	168	light 11	—
Bedroom 1	13-16	35/25	62.2	31.1	46.7	light 13 & 14	light 13
Bedroom 2	17-20	35/25	66.4	33.2	49.8	light 17 & 18	light 17
Remarks	G1: policy with cost goal; G2: policy with comfort goal The illuminance limit shows both the reduced illuminance and the unacceptable illuminance						

When the policy goal was cost, each room turned off some lights. The 'comfort' cost was 50% reduction of illuminance. When the policy goal was comfort, each room turned off one light except the bathrooms. This meant that the light schedules in the bathrooms were not changed. For the other rooms, the 'comfort' cost was 25% reduction of illuminance. For both cases, the actual illuminance was above (or equal to) the illuminance limit.

The accumulated electricity cost in the whole month is displayed in Figure 6-4.

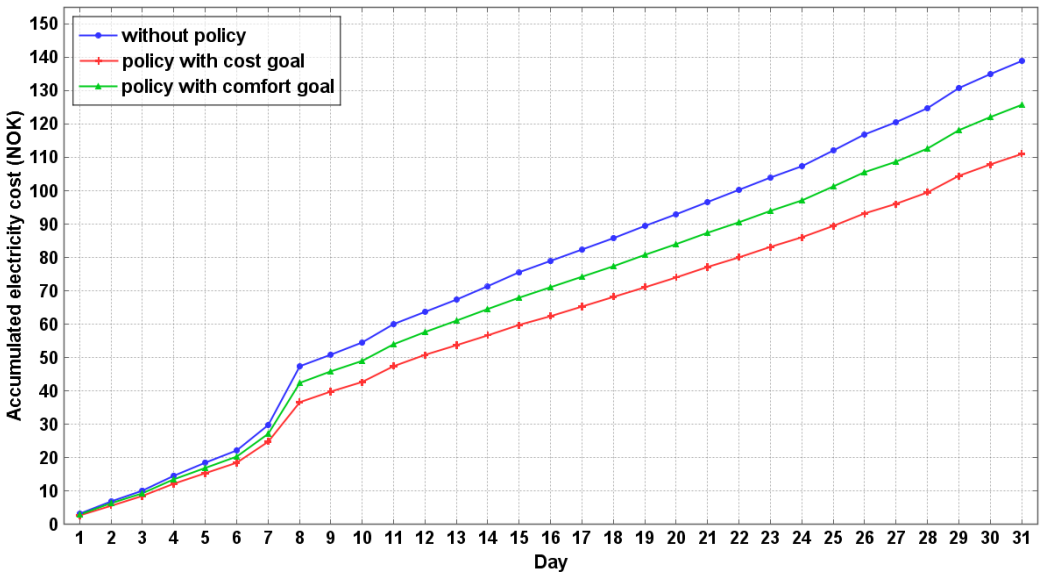


Figure 6-4: Accumulated electricity cost in Jan. 2010 (Test case 2)

The original cost was 138.9497NOK. When policy with cost goal was applied, 27.8204NOK (20.02%) was saved. When policy with comfort goal was applied, 13.2601NOK (9.54%) was saved.

Both test case 1 and test case 2 showed that the high price response policy for the lighting system not only efficiently reduced the electricity cost (by about 20%), but also kept the illuminance above the limit, which was also referred to user comfort.

6.2 Washing machine testing

The washing machine operations were more complicated than the lighting system, and more test cases were constructed to fully test the model and the policy. For all test cases, the washing machine was put in bathroom 1. In the design of the washing machine model, the powers of sub-processes were not fixed values due to the nature of the actual device. However, in order to make the simulation environment identical for all simulations in the same test case, the powers must be fixed. The values used in all the simulations are listed as below:

Table 6-4: Program power settings of the washing machine

Programs	Sub-process powers				
	Water intake	Heating	Rinse 1	Rinse 2	Spin
Synthetics	39.58	1856.243	31.54	45.593	371.958
Wool	11.375	1151.237	23.247	5.762	285.504

The values were chosen by creating a function in Java to randomly generate numbers. The functions made use of the Random class [17] in Java API, and it assumed that the numbers were uniformly distributed.

Due to the implementation of the time trigger, the start time for each program were set to the same value in order to trigger the policy just one hour before the target hour.

6.2.1 Test Case 1

The program setting and time plan is shown in Table 6-5. The washing machine was used twice a week with different programs.

Table 6-5: Program and time configuration in test case 1

Day-of-week	Programs	Start time	Latest time	Triggering time
Wednesday	Synthetics	17:00	21:00	16:00
Saturday	Wool			

Based on the time plan, the possible scheduling hours are during 16:00-21:00. The price fluctuation for these hours is illustrated in Figure 6-5.

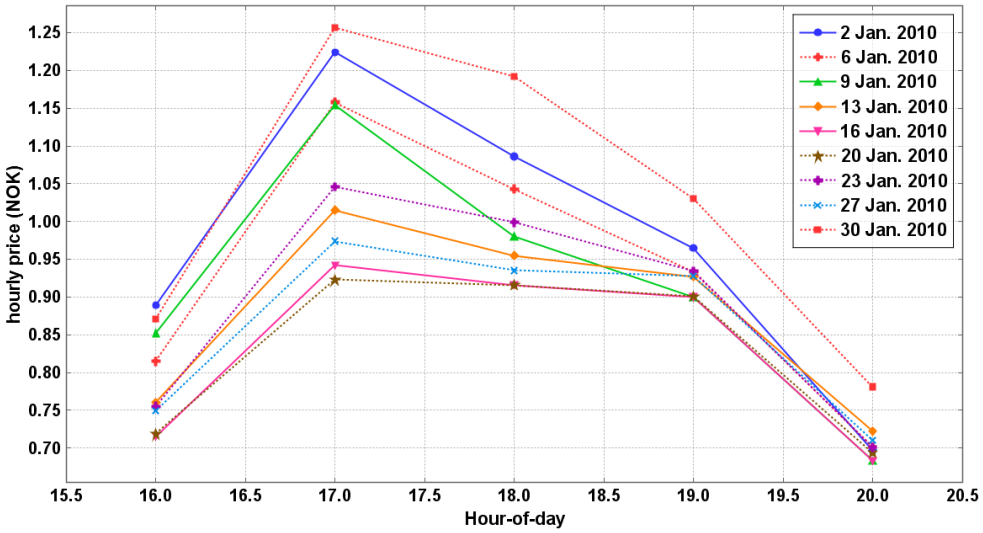


Figure 6-5: Prices between target hour and latest finish time

As shown in the above figure, hour 17 was the highest price hour, and the price went down after that. The price data used in this project indicated that hour 17 and 18 were high price hours on the working days shown in the above figure. The start time (17:00) was chosen to see if the policy can effectively adjust the program schedule and reduce the electricity cost. If the policy was successfully executed, the schedules for all programs should be adjusted, because the price of the planned start time was the highest. The adjusted schedules for all the working days in January 2010 are displayed in Table 6-6.

Table 6-6: Adjusted schedules for all working days in January 2010

Day	Program total time/*min*	Adjusted schedule	
		Policy with cost goal	Policy with comfort goal
2	40	20:00-20:40	16:20-17:00
6	80	16:00-17:00 & 18:00-18:20	17:40-19:00
9	40	20:00-20:40	16:20-17:00
13	80	16:00-17:00 & 18:00-18:20	17:40-19:00
16	40	20:00-20:40	16:20-17:00
20	80	16:00-17:00 & 18:00-18:20	16:40-18:00
23	40	20:00-20:40	16:20-17:00
27	80	16:00-17:00 & 18:00-18:20	17:40-19:00
30	40	20:00-20:40	16:20-17:00

When the policy with comfort goal was applied, the 'comfort' cost for all the programs except 20th was a 40-minute shift of the start time. The 'comfort' cost for 20th January 2010 was a 20-minute shift. All these 'comfort' costs were within the comfort cost limit, which in this case was one hour.

When the goal was cost, all 40-minute programs were postponed to hour 20 because it was the lowest price hour. All 80-minute programs were scheduled to start one hour earlier, and they were paused for one hour. Considering that the sub-process with largest energy consumption (the heating process) was within the first 30 minutes, starting the washing machine at the lowest price hour 16 can greatly reduce the electricity cost.

Figure 6-6 illustrates the electricity costs for all working days with and without policy.

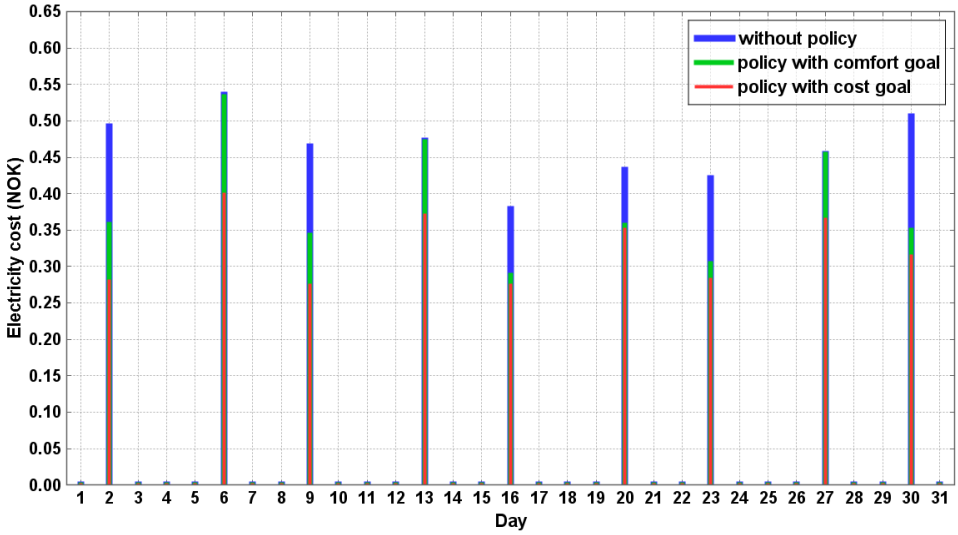


Figure 6-6: Electricity cost for all working days in January 2010 (Test case 1)

The policy reduced the electricity cost of every program in the whole month. When the policy goal was comfort, the reduction was not so obvious for the 80-minute programs (day 6, 13 and 27) due to the comfort cost restriction and the small energy consumptions of several sub-processes.

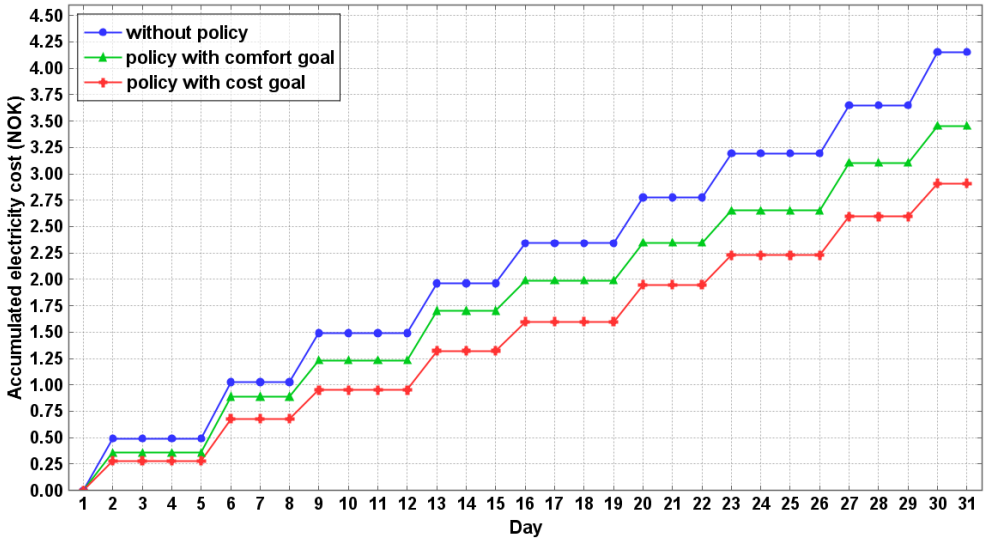


Figure 6-7: Accumulated electricity cost of the washing machine (Test case 1)

Figure 6-7 illustrates the accumulated electricity cost for the whole month, which included nine programs in total. In January 2010, the policy with cost goal reduced the electricity cost by 29.99% (=(4.1559-2.9096)/4.1559), and the policy with comfort goal reduced the electricity cost by 16.86% (=(4.1559-3.4551)/4.1559).

6.2.2 Test Case 2

In the second test case, only the start time for all programs was changed. It was changed to 17:30. This was done to test more time span cases in the policy with different goals. When the goal was cost, the resulted adjustment was identical to that in test case 1. The reason is that the possible options of action were only dependent on the total time of the program. The time span of the program only affected the calculation of original cost. But when the goal was comfort, the time span also affected the possible options because the 'comfort' cost (deviation of start time from the original) must be considered.

When the policy with comfort goal was applied, the adjusted schedules were:

40-minute program:	18:00-18:40
80-minute program:	17:40-19:00 except 20th January 2010 (18:00-19:20)

The 'comfort' cost for the 40-minute programs was a 30-minute shift (from 17:30 to 18:00), and it was a 10-minute shift for the 80-minute programs except 20th January 2010, which had a 30-minute shift. However, all these 'comfort' costs were within the comfort cost limitation two hours. Figure 6-8 illustrates the accumulated electricity cost with and without policy.

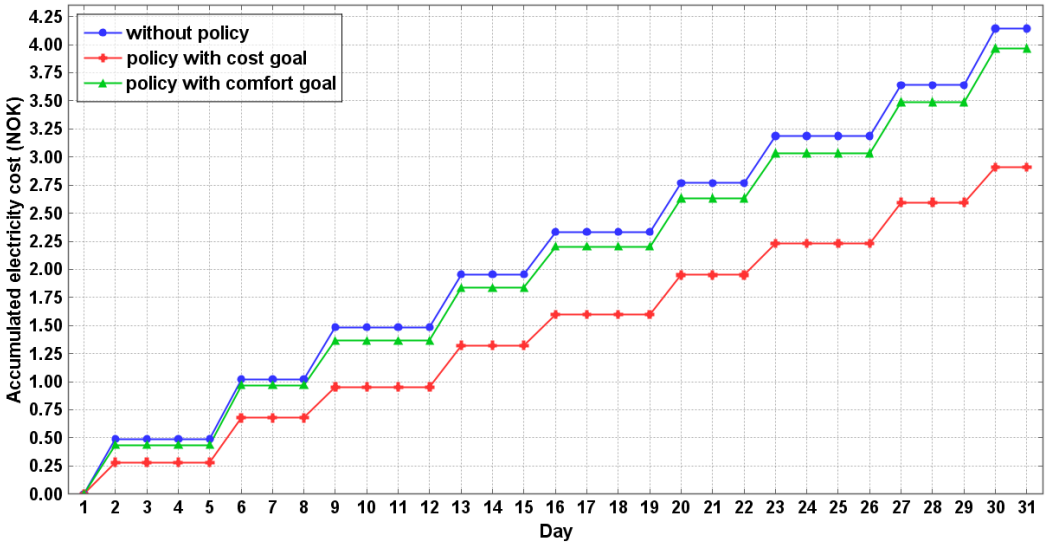


Figure 6-8: Accumulated electricity cost of the washing machine (Test case 2)

As shown in Figure 6-8, for the whole month, the electricity cost was reduced by 29.80% (=(4.1448-2.9096)/4.1448) when the policy with cost goal was applied, and

4.32% (=(4.1448-3.9659)/4.1448) when the policy with comfort goal was applied. The latter was much lower than that in test case 1, and it was due to a smaller time shift for the 80-minute programs. When the time shift was smaller, less energy was multiplied with a lower price, which led to a larger electricity cost.

6.2.3 Test Case 3

In this third case, the start time was further postponed to 17:50 in order to test the 80-minute programs with three-hour time span. All other configurations were identical to those in the first two test cases.

Again, when the policy with cost goal was applied, the changed schedules were the same as in the first two cases, although it just reduced the electricity cost by 27% (=(3.9859-2.9096)/3.9859, as shown in Figure 6-9). When the policy with comfort goal was applied, the resulted schedules were different. The adjusted schedules are:

40-minute program:	18:00-18:40
80-minute program:	18:00-19:20

All programs were postponed for 10 minutes, and the reason was simple. The prices from 17:00 decreased gradually, so postponing the program would certainly reduce the cost and the 'comfort' cost was small. The reduction of electricity cost was totally 2.77% (=(3.9859-3.8753)/3.9859). As the time shift became even smaller than that in test case 2, the reduction was smaller. The accumulated electricity cost in this test case was shown in Figure 6-9.

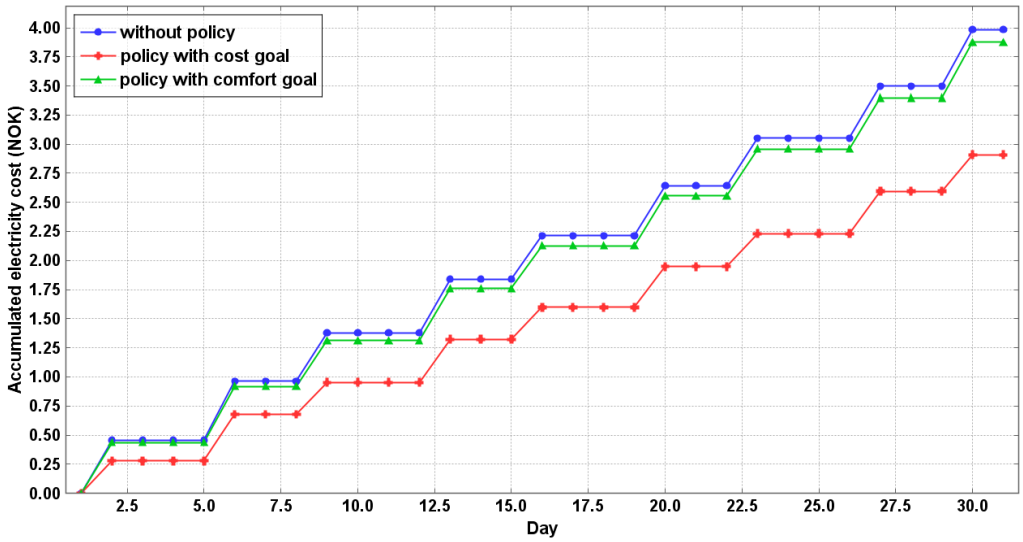


Figure 6-9: Accumulated electricity cost of the washing machine (Test case 3)

All the statistics from the above three test cases had illustrated that the high price response policy for the washing machine reduced the electricity cost. The policy with cost goal had efficiently reduced the electricity cost by 27%-30%. Although the

reduction of electricity cost resulted by the policy with comfort goal was relatively small (2.77%-16.86%), it did reduce the electricity cost. User comfort was also maintained by limiting the time shift within two hours when the goal was comfort.

7 Evaluation and Discussion

This chapter evaluates the development process of this project and discusses the limitations. The reflection on the process by the author is also included.

7.1 Evaluation of the process

By following the agile development methodology, different phases of the development process were carried out in an iterative manner. The implementation phase was more complex than the design phase, especially the implementation of the high price response policies. Take the policy with comfort goal for the washing machine as an example, the design was just to either postpone or advance the start time according to the prices and program total time. But during the implementation phase, the prices, program total time and program time span led to several checking conditions, which made the algorithm in Figure 5-3 very complex.

Apart from the manual inspection of the algorithm and semantics, an inspection of the detailed execution results of the policy was needed. The large amount of execution results made the inspection complex.

The feedback from the implementation phase refined the design details of the energy models and the high price response policies. The testing cases discovered several defects of the implementation and they had been corrected.

7.2 Evaluation of contributions

A lighting system energy model and a washing machine energy model were developed and added to the existing SMASH platform. Several assumptions were used in this project for simplicity, such as the uniform distribution of washing machine power specification and the usage frequency of a washing machine on a working day (one time per working day). Further research can be done and these assumptions can be changed to be more realistic.

The high price response policies developed in this project can efficiently reduce the electricity cost. The cost reduction of the lighting system can be up to 50%, and the cost reduction of the washing machine can reach 29.80%. User comfort was maintained by setting appropriate constraints. Refactoring of the algorithm is possible to make it simpler and easier to understand.

7.3 Evaluation of limitations

The washing machine power specification was obtained by measuring the real-time power consumption of an actual device. The estimated energy consumption in this project turned out to be too low to begin with. The energy rating for the washing machine used in this project indicated that the annual energy consumption was approximately 217kWh, which was about 2kWh per wash if it was suited to the configuration in this project. However, the measured power consumption of the actual device would only give a maximum 0.64kWh per wash, which resulted in a low electricity cost of 40NOK in January 2010. This had made the energy management for the washing machine less meaningful because it seemed unnecessary to reduce the cost which was already low. The energy consumption of the washing machine is dependent on several parameters such as the water temperature and the weight of clothes. More measurements of the power consumption can be carried out, or different devices can be used to get more precise data for the simulation.

The triggering conditions supported by the time trigger developed in the platform were limited. The time trigger was registered before the simulation starts, and it cannot be changed during the simulation. In case of the washing machine triggering condition, it was not possible to define the triggering time as one hour before the planned start time because the start time could be different on every working day. There was one way to check the triggering condition resulted by different start time, but this approach was time-consuming because the policy would be triggered very frequently. This time trigger can be refined in the future to support more triggering situations.

7.4 Reflections

During the design and implementation phases, several assumptions and constraints were defined in the author's perspective, or to say in a potential user's perspective. For example, the washing machine working frequency was defined as two times a week, and at most once per day. This was based on the author's experience. No survey or questionnaire was conducted to gather more information which would give more accurate statistics.

The policy algorithms for determining the final actions were on the other hand, as the author considered, comprehensive enough. The simulation results indicated that the policies could efficiently reduce the electricity cost, which accomplished the main objective of developing the policies.

8 Conclusion

The SMASH simulation platform developed at NTNU resembles a real adaptable smart home. It aims at simulating the operations of common home appliances and managing the energy consumption by applying energy management policies. These policies provide a means to reduce the electricity cost in a house while maintaining the user comfort above certain level.

This project was proposed to develop more home appliance energy models and the corresponding energy management policies, which could be added to the existing platform. This report described the process of developing two working energy models and the corresponding high price response policies which are consistent with the existing SMASH simulation platform.

8.1 Contributions

A lighting system energy model and a washing machine energy model were developed in this project. The energy models included the power requirements of different operation cycles of the devices and estimated the energy consumption of the devices within a certain period (one month). Different time schedules and device programs were used in the test cases to thoroughly include the functionalities of the devices and to test the effect of the energy management policies.

Two high price response policies were developed, one for the lighting system and the other for the washing machine. The policies defined the algorithms to arrive at a set of actions which adjusted the time plan for the devices and as a result, reduced the estimated electricity cost of the devices. Both policies had two goals, cost and comfort. The policies were tested under several different environment configurations and the results indicated that these policies had effectively reduced the electricity cost.

Both the electricity cost reduction and the user comfort are important criteria for evaluating the policies. In the policy with either goal, the electricity cost was reduced. The user comfort was kept above certain levels, which were defined as the constraints.

To conclude, this project had accomplished the tasks in the above manner.

8.2 Future work

First of all, several assumptions were used to simplify the case. In the future, some assumptions can be removed by using more realistic data. Part of the data can be collected by conducting a survey or questionnaire. More real devices can also be tested to gather data since the platform is a simulation of the real system.

Secondly, the limitations had reduced the flexibility of the platform and restrict the functionalities of the platform. Thus the limitations described in the last chapter must be eliminated in the future to expand the platform.

Finally, more energy models and energy management policies can be developed and added to the platform. A smart home contains different kinds of home appliances, and it takes time to include more such appliances to the platform. The more devices are included, the more precise and realistic statistics we can get.

References

[1] *AIM: A novel architecture for modelling, virtualizing and managing the energy consumption of household appliances*, http://www.ict-aim.eu/home.html

[2] Beywatch: Beywatch Home, http://www.beywatch.eu/index.php

[3] SmartHouse-SmartGrid Consortium: Smart House/Smart Grid, http://www.smarthouse-smartgrid.eu

[4] Suyang Zhou , Zhi Wu , Jianing Li, Xiao-ping Zhang, *Real-time Energy Control Approach for Smart Home Energy Management System*, Electric Power Components and Systems, 42:3-4, 315-326, 2014.

[5] OpenDSS, http://electricdss.sourceforge.net/

[6] Electric Power Research Institute, *Introduction to the OpenDSS*, April 2009.

[7] GridLAB-D: GridLAB-D Home, http://www.gridlabd.org/

[8] EnergyPlus Energy Simulation Software, Energy Efficiency & Renewable Energy, http://apps1.eere.energy.gov/buildings/energyplus/energyplus_about.cfm

[9] ENERGYPLUS™, *Getting Started with EnergyPlus, Basic Concepts Manual-Essential Information You Need about Running EnergyPlus*, October 2013.

[10] *TRNSYS: A TRaNsient Systems Simulation Program*, http://sel.me.wisc.edu/trnsys/

[11] Matthew J. Duffy, Marion Hiller, David E. Bradley, Werner Keilholz, Jeff W. Thornton, *TRNSYS-features and functionality for building simulation 2009 conference*, Eleventh International IBPSA Conference, July 2009.

[12] Kornschnok Dittawit, *Technical Report V2.0, SMASH Simulation Platform*, 2013. (unpublished)

[13] Paramai Supadulchai, *NxET Reasoning Machine*. Plug-and-plat technical report, ISSN:1500-3868, 2007.

[14] Kornschnok Dittawit, Finn Arve Aagesen, *Architecture and Functional Framework for Home Energy Management Systems*. T. Bauschert (Ed.): EUNICE 2013, LNCS 8115, pp. 173–184, 2013.

[15] *The IESNA Lighting Handbook*, ninth edition. ISBN 0-87995-150-8.

[16] Java™ Platform, Standard Edition 7 API Specification, http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html

[17] Java™ Platform, Standard Edition 7 API Specification, http://docs.oracle.com/javase/7/docs/api/java/util/Random.html

Appendix I: Configuration of data illustration

Data collection is an important procedure during the simulation process, and it is the main source for policy validation and evaluation. The SMASH simulation platform generates data files for different types of data and creates a way to demonstrate the data in a web browser (an .html file is created along with the related data files). The following part will explain the steps for configuring the environment in sequence. The demonstration is done on a desktop computer with Windows 7 operation system.

I. Development environment setup

The SMASH simulation platform is Java based, thus a Java runtime environment is required in order to run the programs. A JVM (Java Virtual Machine) is needed regardless of the operating system. Oracle JDK 7 (Java Development Kit 7) was used for this purpose. Different installation instructions of the JDK on different operating systems are found at http://docs.oracle.com/javase/7/docs/webnotes/install/index.html.

Eclipse (Kepler package) was used as the IDE (Integrated Development Environment) for the development of this project. Installation details of Eclipse can be found at http://wiki.eclipse.org/Eclipse/Installation. To open Eclipse, simply run the Eclipse executable file *eclipse.exe* located in the installation sub-directory *eclipse*.

Usually the JAVA_HOME, PATH and CLASS_PATH environment variables are configured as a convenience. The detailed explanation and settings of these variables are found at http://docs.oracle.com/javase/tutorial/essential/environment/paths.html.

The SMASH platform contains three Java projects: NxET, SMASHCSS and SMASHsim. They refer to the three layers in the functional structure of the platform. To develop the platform, just import the existing projects into Eclipse. http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2Ftasks%2Ftasks-importproject.htm gives the step-by-step instructions for importing existing projects into Eclipse.

The newly developed policies are under SMASHCSS/policies_add. The files names are *pls_price2.xet.xml* (for the lighting system) and *pwm_price.xet.xml* (for the washing machine).

The newly developed energy models and relevant entities are under the SMASHsim/src/no.ntnu.item.smash.sim.model package and the SMASHsim/src/no.ntnu.item.smash.sim.structure package respectively. The test cases are under SMASHsim/src/no.ntnu.item.smash.sim.experiments.light (for the lighting system) and SMASHsim/src/no.ntnu.item.smash.sim.experiments.washing (for the washing machine). The following figure illustrates the file directory for the above files.

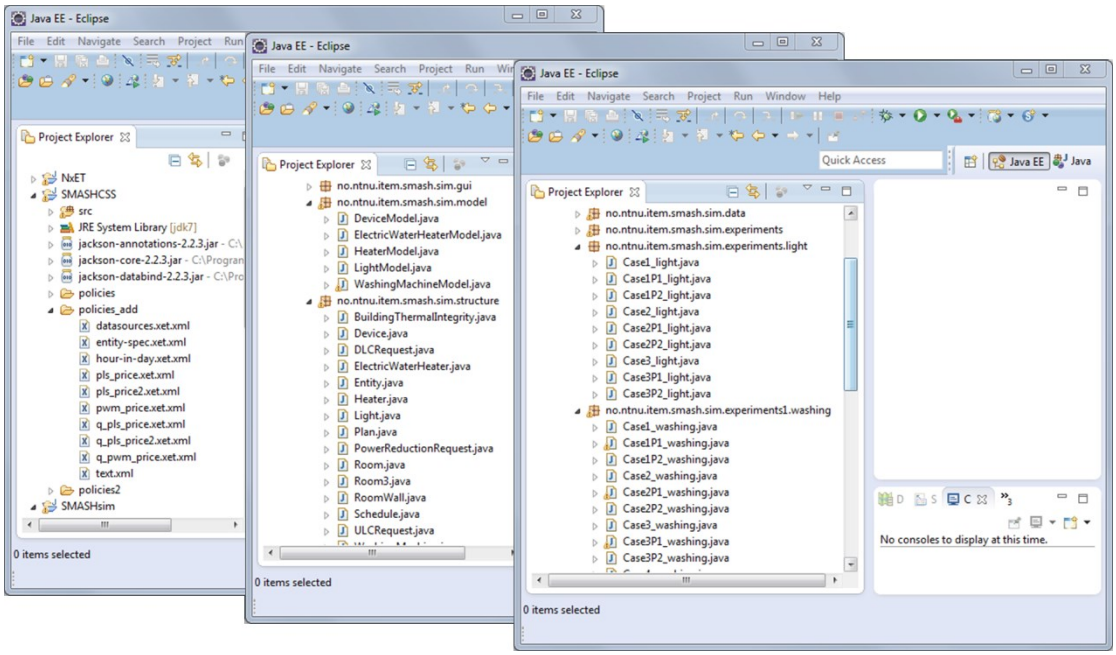


Figure A-1: Directories for policies, energy models and test cases

The generated data files for all the simulations are under SMASHsim/experiments1. The naming format for the folders under this directory is:

Case[test case no.][policy no. (P1: cost; P2: comfort; optional)]_[device identifier];

For example, Case1_light indicates the files for Test Case 1 without policy for the lighting system; Case1P1_light indicates the files for Test Case 1 with policy with cost goal for the lighting system.

The data files for the washing machine are under SMASHsim/experiments1/washing.

II. Data illustration setup

Required software:

A web browser (Internet Explorer 11 used in the demonstration);
A web server (Tomcat 7 used in the demonstration).

The installation and configuration of Tomcat 7 can be found on the official website http://tomcat.apache.org/.

To illustrate the statistics in form of graphs in a web browser, the following steps should be followed.

1. Put all the related simulation data files in the Eclipse workspace under the Tomcat installation sub-directory called *webapps*. In case of this demonstration, the directory is at C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\resultsviewer\smash\Experiments.

2. Start Tomcat. Go to Start menu > All programs > Apache Tomcat 7.0 Tomcat7 > Configure Tomcat, and the following wizard will show up. Click 'Start' to start Tomcat.

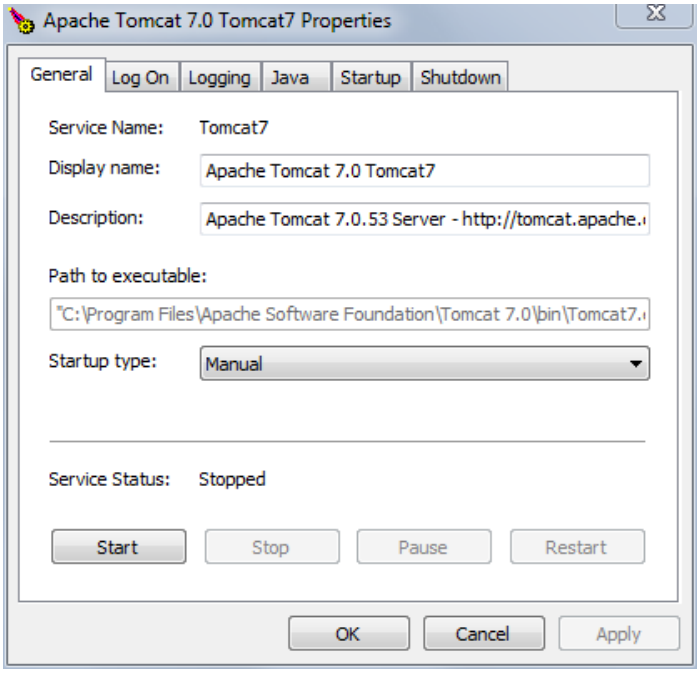


Figure A-2: Apache Tomcat configuration wizard

3. Open Internet Explorer. Take Test Case 1 without policy for the lighting system (the Case1_light folder) as an example. Enter the following address: http://localhost:8080/resultsviewer/smash/Experiments/Case1_light/results.html.

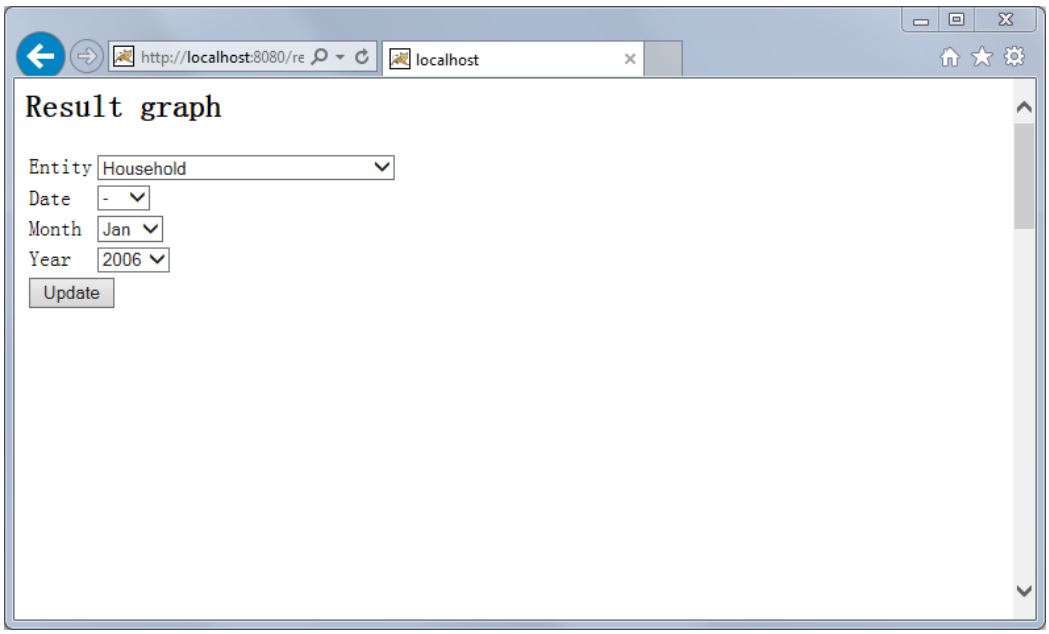


Figure A-3: Front page for the data illustration

The Entity includes all rooms and all the devices been developed so far in the house (both active and passive). Date, Month and Year indicates the simulation date. In this demonstration, only days in January 2010 will contain valid graphs, since it was set as the simulation period in this project. Choose the entity and the date you want to see, and then click on 'Update'. The relevant statistics will be shown in graphs. Here the Household statistics on 1st January 2010 is taken as an example demonstration. The graphs are shown in the following figure.

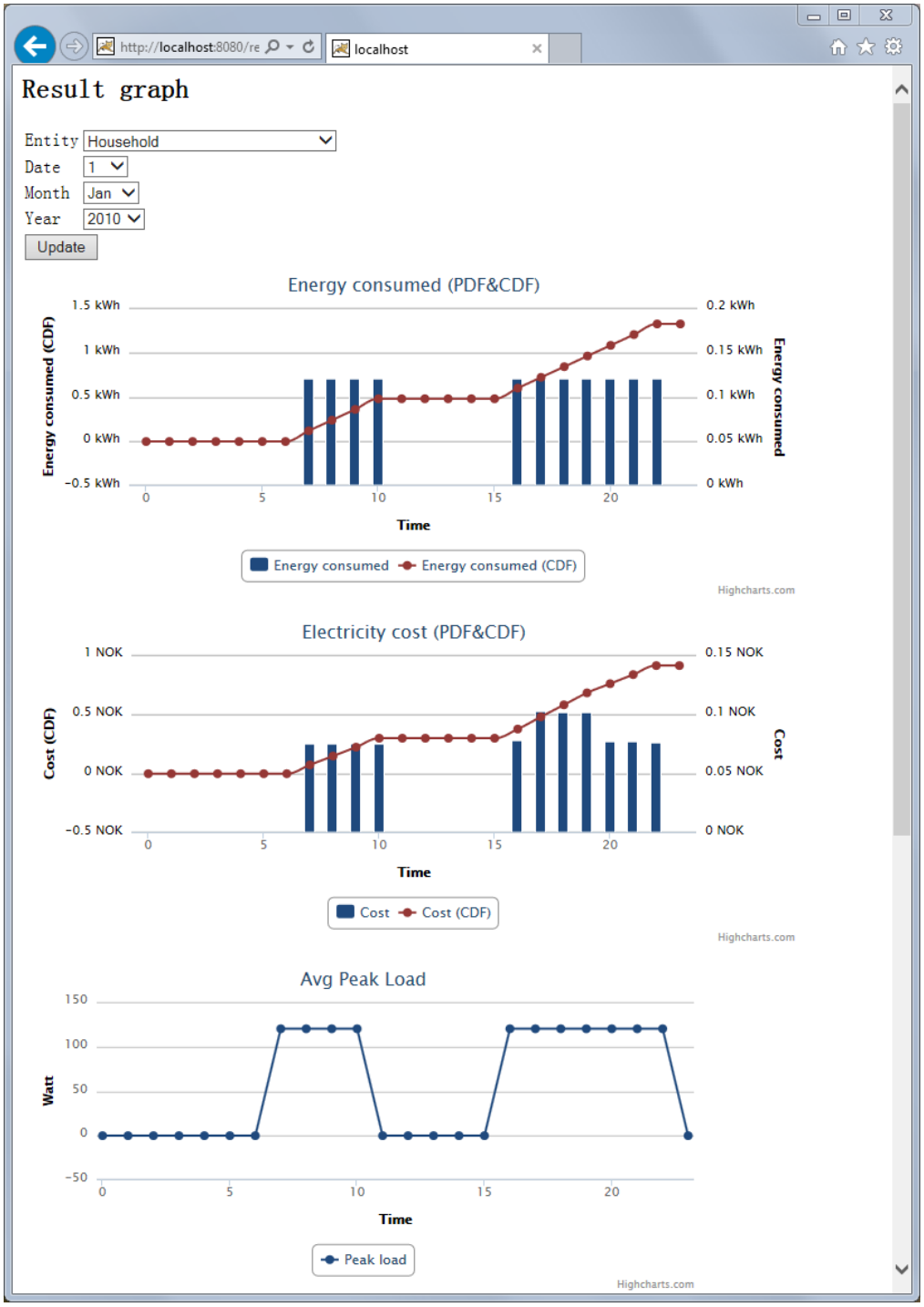


Figure A-4: Result graphs for Case1_light folder