Integrated Simulation Platform for Optimized Building Operations

A. Mady, M. Boubekeur and G. Provan
CCSL, Computer Science Department, University College Cork,
Lancaster Hall, 6 Little Hanover street,
Cork, Ireland.
{mae1, m.boubekeur, g.provan}@cs.ucc.ie

Abstract—In order to develop an integrated system that can optimise building operations, all facets of monitoring, control and reconfiguration need to be integrated. To achieve this, we need to develop an integrated system model, together with associated sub-system interfaces. This links the low-level wireless/wired systems monitoring with multi-level control, building information management and user-preference inputs. In this paper we describe an integrated simulation platform that partially fulfils such objectives. We detail through a case study an integrated simulation platform allowing testing, refinement and optimization of the design at an early stage. The paper provides a detailed simulation results and outlines the future works.

Keywords- optimized building operations; hybrid systems, integrated simulation, user preference, lighting system

I. INTRODUCTION

Intelligent buildings also referred to as smart buildings, are buildings that incorporate a Building Management System (BMS). The main objective of a BMS is to maximise the productivity of its occupants, to allow an efficient management of resources and to minimise costs. A typical BMS would provide a core functionality that keeps the building climate within a specified range, automates the lighting based on an occupancy schedule, and monitors system performance and device failures.

One of the main objectives of building control is to maintain a comfortable environment in an energy-efficient manner. To this end, the most important parameter to control is temperature; however, other parameters are also important, including lighting, humidity, CO_2 levels, air flow etc.

A system integration framework that integrates all the services provided by a building is key to fulfilling these goals. Among the advantages of an integrated environment are the improvement of safety, system control reliability and the reduction of operating costs.

The aim of our research is to define a methodology for modelling and integrating building management system services. Our approach is built around two main aspects of building automation: requirements and system integration. The requirements guide the whole development process, while the integration guarantees system interoperability and increases the optimization capabilities. In order to maximise the use of the requirements in the development process, we capture them using "use-case diagrams" when possible. The objects used in these diagrams conform with the Industry Foundation Classes (IFC) standard [1]. This will ensure portability and extensibility of the approach, since the IFC is an object-oriented formalism. Therefore the building description, the sensor features and the sub-system specifications are all captured using IFC format.

Our approach makes use of learned data, a very important feature which is not usually integrated in the BMS. Indeed, one of the objectives is to provide a machine learning functionality allowing further efficient optimization. Based on this, the BMS can predict the future behaviour of the building in different conditions, and would then implement the appropriate optimizations in a reconfigurable fashion. The building is operated for a certain time while the data is continuously collected via the sensors in a data warehouse module. The predicted behaviour is then computed from the information in the data warehouse.

In this paper we focus on the simulation phase. Three steps are involved in the design flow, as explained later: modelling, simulation and implementation. We describe, through a case study, an integrated simulation platform that allows testing, refinement and optimization of the design at an early stage. In particular, we illustrate the developed methodology for modelling and simulation of control strategies. The case study we consider here is a lighting system for an office area. This example illustrates the combination of discrete events behaviour (presence detection, light switch on/off) and hybrid properties for the luminosity control, i.e. where both discrete and continuous aspects are considered.

The reminder of paper is organized as fellows: Section 2 presents the preliminary architecture and outlines its main components. Section 3 describes the design flow; it focuses on the integrated simulation approach. Section 4 illustrates the integrated simulation phase through an intelligent lighting system. In Section 5, we outline the simulation results. We end in Section 6 by giving a discussion of our work and outlining future perspectives

II. PRELIMINARY SYSTEM ARCHITECTURE

The global architecture we propose to implement is a service/model-driven one. It is composed of several components operating at different levels. Here we outline the main components. At the specification level, we distinguish the requirements definition, sensor/actuator network specification and the building information model. According to the specification, a first modelling of the required sensor network platform is provided. This will interact (sense and actuate) with the different services that the building should provide. The architecture also includes a global control device consisting of a virtual software entity that guarantees the consistency of the system and the interfaces between the architecture components. We now summarise the main components:

- A *Building Information Model* (BIM) provides a description of the building, including location of rooms, zones, floors and windows, lights, the Heating, Ventilation and Air Conditioning (HVAC) equipment, the physical characteristics of the building, and data from multiple sensors. The building information model also contains data on the state of the building systems and components (e.g., components not working properly, systems due for maintenance, etc).
- A *requirements module* stores all the requirements regarding the building's functionality and infrastructure. It consists of a data base indexed in a convenient layout in order to provide an easy access. It also efficiently interfaces with the different modules of the architecture. In our research we choose to work with the commercial tool DOORS.
- The *Global System Control* (GSC) is a software entity for systems integration. The GSC is responsible, among other tasks, for exchanging information between the different sub-systems, such as HVAC, lighting, fire control, etc.
- A *Human-interface device* is a software interface with the building management system. It provides different levels of access, depending on the degree of responsibility of the user. For example, a simple user might only consult information regarding building operations; however, a security manger can trigger a fire alarm signal, or switch it off.
- A *modelling tool for building services*. This component involves the modelling process of the different services provided within the building, in accordance with the requirements. Different kinds of modelling techniques are considered, depending on the nature of the services. For purely discrete-event-driven systems, we would simply use finite state machines or statechart diagrams [2]; however, for systems that involve continuous parameters, hybrid automata are needed. These models constitute the foundations for the maintenance and diagnosis components.

Finally, the global architecture has the following properties:

- Implements an integrated platform that enables an efficient optimization process;
- Stores models developed from requirements/objectives;
- Guarantees interface specification to achieve consistent data exchange, in order to ensure the consistency of any future modifications;
- Enables parameterized views; for example, according to his prerogatives, a user can access a specific view, such as a control view, energy consumption view, etc.

Fig. 1 shows an extended diagram of the system architecture. In addition to the components we mentioned earlier, the diagram introduces other components in relation the ITOBO project [3]. ITOBO is a strategic research cluster for Sustainable and Optimised Building Operation, funded by Science Foundation of Ireland (SFI). For example, it introduces a performance framework module responsible for the specification of the BIM, and a Data warehouse module in charge of data collection and management.

III. System Design Flow

The system design flow is described in Fig. 2. We start by defining relevant scenarios to be conducted within the building. These scenarios are defined using a high level specification language, the Unified Modelling Language (UML) [4]. The UML models are interpreted using specific models for simulation and analysis purposes. At this level we use an optimization loop to optimize the model at an early stage of the development. The development process finishes with the implementation phase. The integration process is performed through the implementation of a model-/service-based middleware [5] platform, which enables component connection and data exchange. All the different components of the architecture collaborate with the requirements module and the sensor network platform.

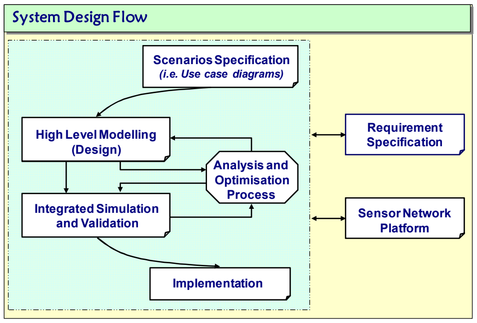

Figure 2. System Design Flow

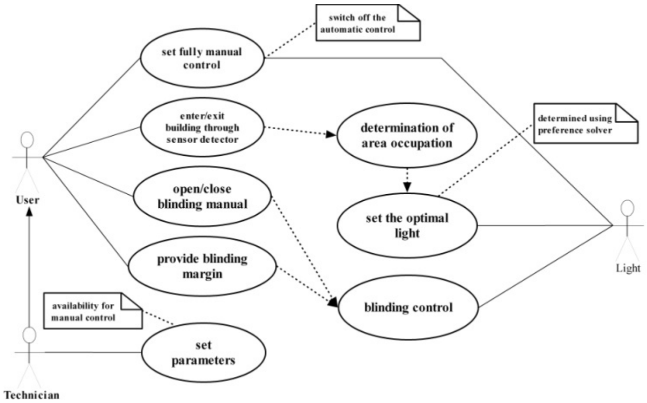

Figure 3. Use-case Diagram for the Lighting System

Fig. 3 shows a UML use case diagram for the lighting system, from which the simulation models have been created. The full details of the case study are presented in Section 4.

A. *Integrated Simulation Approach*

As stated in the introduction, our focus is integrated simulation, which we illustrate through a simple lighting scenario. Assuming that we decompose a system into sub-systems, we model each sub-system or service using automata. Most of the services are discrete-event-oriented; however, some sub-systems are dynamic and need to be modelled using hybrid systems. Consequently, we have chosen to use a hybrid systems modelling language and simulation engine. In this study we use the CHARON tool [6] for simulation of hybrid systems. CHARON offers great modelling features, in particular modularity and hierarchy, which correspond to our modelling approach.

To simulate the models, we must model the environment, e.g., the behaviour of the sensors and the building occupants. The environment will provide a stimulus (control input) for the simulation. In our work we also consider a preference model over lighting and temperature. We have integrated the preferences inside the modelling for simple preference models; however, we have implemented an interface with a constraint solver to handle complex preference models.

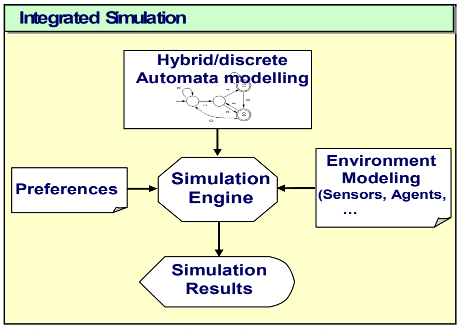
Integrated Simulation
Hybrid/discrete Automata modelling
Preferences
Simulation Engine
Environment Modeling (Sensors, Agents, …
Simulation Results

Figure 4. Integrated Simulation Architecture

The overall model is built incrementally. The whole system, or part of it, can be simulated by composing the relevant sub-systems as they would execute in reality, i.e., in sequence or in parallel. Once the models are built and the interface with the preference solver is set, the simulation is executed and the results are analysed, as shown in Fig. 4.

IV. CASE STUDY

This section describes a simulation for a building lighting scenario. The model provides lighting controls based on presence sensors, ambient light and area occupancy. It incorporates occupant preferences via interfaces with user data preferences.

The idea is to track the occupancy and the ambient light to monitor light levels. The model considers the hybrid property of the system. For example, the light level values change continually according to dynamical models.

A. *CHARON Hybrid Modelling Tool*

CHARON is a high-level language for modular specification of multiple, interacting hybrid systems, and was developed at the University of Pennsylvania [6]. The toolkit distributed with CHARON is entirely written in JAVA, providing many features: a Graphical User Interface (GUI), a visual input language, an embedded type-checker, and a complete simulator. Through an intermediate XML format the graphical editor converts the specified model into CHARON source code. The CHARON toolkit is capable to interact/use other external JAVA programs. The simulator itself is an executable Java program.

CHARON adopts a hierarchal modelling framework based on the statechart modelling technique. A hybrid system is described in CHARON as follows [7]:

- *Architectural hierarchy*: The architecture of systems is described with communicating agents. Those agents share information through shared variables or communication channels. Agents are either atomic or composite.
- *Behavioural hierarchy:* A mode is a construct for the hierarchical description of the behaviour; it has well-defined control, entry and exit points. Transitions between modes are enabled when a condition called guard becomes true. CHARON provides differential and algebraic constraints representing continuous dynamics and invariants forcing a continuous flow to satisfy a condition. The language also supports the instantiation of a mode for the reuse of mode definitions.
- *CHARON variables:* CHARON provides two types of variables, continuous (analog) and discrete. Analog variables are updated continuously while time is flowing. Conversely, discrete variables are modified only when the modes of an agent change. The values of discrete variables do not change in a time flow.

B. *Lighting System Modelling*

The lighting system model tracks the presence of people in the controlled area; according to the current occupancy, a preference solver, developed by Cork Constraint Computation Centre (4C) and based on constraint programming, provides the optimal light intensity that suits the room's users. The controller then maintains this set point by calculating the appropriate internal light intensity and blinding variation margins, as depicted in Fig. 5.

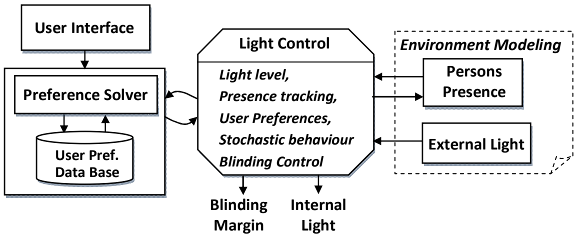

Figure 5. Lighting System Architecture

The lighting system is modelled by three main concurrent agents: Light, GlidingWindowCover and Person agents. The behaviours of each agent are described by different modes, as follows:

- *Light Agent:* Models the lighting controller and external light intensity. Fig. 6 shows the main Light modes, which are “TakeControl” (which contains two sub-modes), and “SustainableDevelopment”, which is used to model the external light variation during day time, and “Controller” which is used to control the light manually.

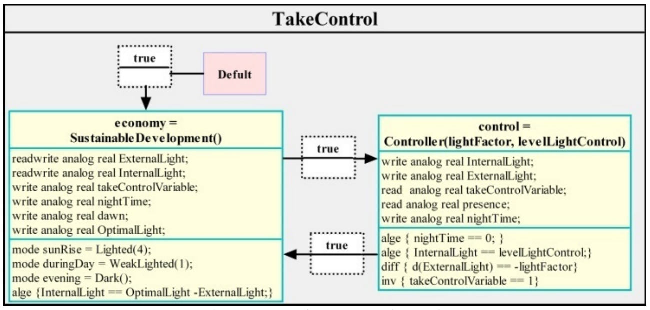
TakeControl
true
Defult
economy =
SustainableDevelopment()
readwrite analog real ExternalLight;
readwrite analog real InternalLight;
write analog real takeControlVariable;
write analog real nightTime;
write analog real dawn;
write analog real OptimalLight;
mode sunRise = Lighted(4);
mode duringDay = WeakLighted(1);
mode evening = Dark();
alge {InternalLight == OptimalLight -ExternalLight;}
true
true
control =
Controller(lightFactor, levelLightControl)
write analog real InternalLight;
write analog real ExternalLight;
read analog real takeControlVariable;
read analog real presence;
write analog real nightTime;
alge { nightTime == 0; }
alge { InternalLight == levelLightControl;}
diff { d(ExternalLight) == -lightFactor}
inv { takeControlVariable == 1}

Figure 6. Take Control Mode

GlidingWindowCover Agent: Models the gliding blinding in order to control the external light intensity. This agent is triggered if the external light intensity is higher than the optimal light intensity (optimal light intensity relies on the occupancy value). The GlidingWindowCover Agent has one main mode, “WindowCover”, described in Fig. 7. This agent controls the blinding using three sub-modes: ManualControl opoerates when the user doesn’t want any automatic control, AutomaticWithMargin operates if the user provides a variation margin for the blinding control, and Automatic operates otherwise.

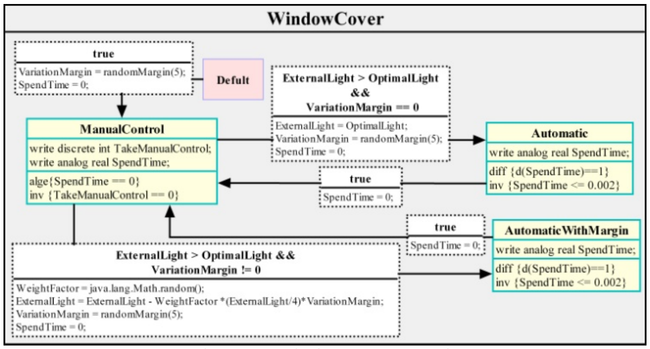
WindowCover
true
VariationMargin = randomMargin(5);
SpendTime = 0;
Defult
ExternalLight > OptimalLight
&&
VariationMargin == 0
ExternalLight = OptimalLight;
VariationMargin = randomMargin(5);
SpendTime = 0;
ManualControl
write discrete int TakeManualControl;
write analog real SpendTime;
alge{SpendTime == 0}
inv {TakeManualControl == 0}
Automatic
write analog real SpendTime;
diff {d(SpendTime)==1}
inv {SpendTime <= 0.002}
true
SpendTime = 0;
true
SpendTime = 0;
AutomaticWithMargin
write analog real SpendTime;
diff {d(SpendTime)==1}
inv {SpendTime <= 0.002}
ExternalLight > OptimalLight &&
VariationMargin != 0
WeightFactor = java.lang.Math.random();
ExternalLight = ExternalLight - WeightFactor *(ExternalLight/4)*VariationMargin;
VariationMargin = randomMargin(5);
SpendTime = 0;

Figure 7. Window Cover Mode

Finally, *Person Agent* is part of the environment model and will be explained in the following section.

To deal with the occupancy information (since the simulation doesn’t handle arrays), we have defined a variable called *occupancy value*. This variable encodes the presence information as follows: each person has an index, and if he is in the zone then the corresponding digit in the equivalent binary value is set to ’1’. For example, if we consider 5 people using the zone, when persons 2 and 4 are in the zone then the occupancy value is equal to 10 (corresponding to "01010" binary value). This method has been used to interface with the preference solver.

C. Lighting System Environments

In order to perform the simulation we have to model the environment of the lighting controller. We distinguish, as shown in Fig. 5, three inputs: person presence, external light and blinding variation margin.

- *Person presence:* the two modes “IsInGenerated” and “IsOutGenerated” determine the occupancy value of a controlled zone. In order to model the presence, a random number generator has been used for each person in the controlled area (e.g. the controller receives ‘1’ when the person enters the zone and ‘0’ when he leaves it). Fig. 8 describes the “IsInGenerated” mode; the “IsOutGenerated” mode is similar except that we consider a person leaving the zone.

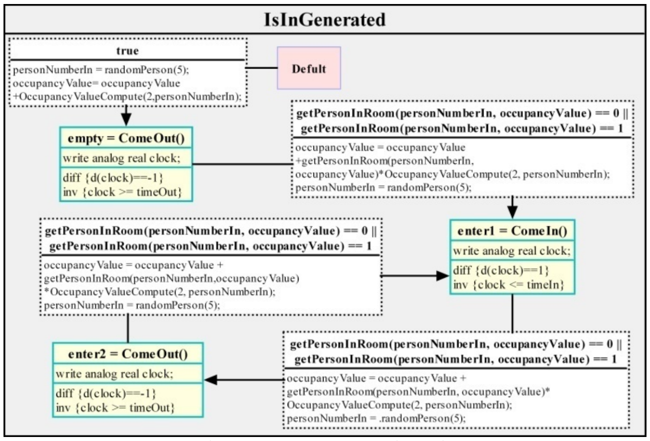
IsInGenerated
true
personNumberIn = randomPerson(5);
occupancyValue= occupancyValue
+OccupancyValueCompute(2,personNumberIn);
Defult
empty = ComeOut()
write analog real clock;
diff {d(clock)==-1}
inv {clock >= timeOut}
getPersonInRoom(personNumberIn, occupancyValue) == 0 ||
getPersonInRoom(personNumberIn, occupancyValue) == 1
occupancyValue = occupancyValue
+getPersonInRoom(personNumberIn,
occupancyValue)*OccupancyValueCompute(2, personNumberIn);
personNumberIn = randomPerson(5);
getPersonInRoom(personNumberIn, occupancyValue) == 0 ||
getPersonInRoom(personNumberIn, occupancyValue) == 1
occupancyValue = occupancyValue +
getPersonInRoom(personNumberIn,occupancyValue)
*OccupancyValueCompute(2, personNumberIn);
personNumberIn = randomPerson(5);
enter1 = ComeIn()
write analog real clock;
diff {d(clock)==1}
inv {clock <= timeIn}
enter2 = ComeOut()
write analog real clock;
diff {d(clock)==-1}
inv {clock >= timeOut}
getPersonInRoom(personNumberIn, occupancyValue) == 0 ||
getPersonInRoom(personNumberIn, occupancyValue) == 1
occupancyValue = occupancyValue +
getPersonInRoom(personNumberIn, occupancyValue)*
OccupancyValueCompute(2, personNumberIn);
personNumberIn = .randomPerson(5);

Figure 8. IsInGenerated Mode

- *External light intensity:* The external light intensity has been modelled using three phases, as depicted in Fig. 9. In each phase, a linear differential equation is used. The first phase corresponds to a positive slope in order to model the sunrise period, the second is a constant slope to model the day time, and the third is a negative slope to model the sunset period.

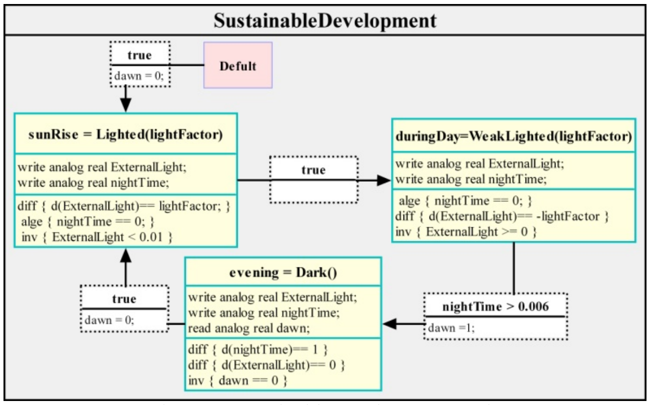
SustainableDevelopment
true
dawn = 0;
Defult
sunRise = Lighted(lightFactor)
write analog real ExternalLight;
write analog real nightTime;
diff { d(ExternalLight)== lightFactor; }
alge { nightTime == 0; }
inv { ExternalLight < 0.01 }
true
duringDay=WeakLighted(lightFactor)
write analog real ExternalLight;
write analog real nightTime;
alge { nightTime == 0; }
diff { d(ExternalLight)== -lightFactor }
inv { ExternalLight >= 0 }
true
dawn = 0;
evening = Dark()
write analog real ExternalLight;
write analog real nightTime;
read analog real dawn;
diff { d(nightTime)== 1 }
diff { d(ExternalLight)== 0 }
inv { dawn == 0 }
nightTime > 0.006
dawn =1;

Figure 9. SustainableDevelopment Mode

- *Gliding window cover variation margin:* The variation margin of the blinding has been modelled as a random number generator varying from 0 to 4. 0 margin means that there is no variation, and hence the system will fully control the blinding; a margin from 1 to 4 defines increasing levels of the control.

D. Deployment

In the real system, the environment models will be replaced by the equivalent sensors. For example, the person presence is monitored using either Radio Frequency Identification (RFID) or Persons Infra Red (PIR) sensors, or simply a person indicates his presence through a computer device; the external light intensity will be measured by the illumination sensor fixed at the external light entrance; the blinding will be used in order to control the intensity of the external light coming to the controlled zone; and blinding variation margin values are provided from the users using either handheld devices or PC application programs.

E. Extensions to the Lighting System Model

The previous model has been created for controlling a single zone. We now explain how this model can be extended to multiple zones.

In order to extend this model to control double zones, two person agents, two Light agents and two GlidingWindowCover agents have been considered. The light intensity interference has been modelled, since the light intensity in one zone is affected by the light intensity coming from the other, as shown in Eq. 1:

$$\alpha1 = \Omega1 - \frac{\alpha2}{r2} - \beta1 - \frac{\beta2}{r2} - 2 * \sqrt{\beta1 * \frac{\beta2}{r2}} * \cos(\theta1 - \theta2)$$

Equation 1. Equivalent Light Intensity

where we define the equation's parameters as follows:

α1: the internal light intensity inside zone 1; ***Ω****1*: the optimal light intensity, which has been calculated depending on the occupancy value in zone 1 (e.g. 500 Lux) [8] and corresponds to the optimal light intensity; *α2*: the internal coming to zone 1; *β2*: the external light intensity coming to zone 2; *θ1*-*θ2*: the phase difference between the β1 waves and β2 waves; and *r*: the distance from the light source in zone 2 to zone 1.

In the multiple zone model, the Light waves coming from a light bulb are incoherent waves, so there is no interference between the internal lights in zone 1 and zone 2. Hence the equivalent intensity is the summation of both intensities after considering the light source in zone 2 is a distance *r* from zone 1 [9]. The external light intensity is subject to scattering light sources and emits incoherence waves, so the external light interference has been calculated as shown in Eq. 1 [10]. In order to model the phase difference of the interfered waves coming from the external light, a random phase generator has been created for each external light wave.

V. Simulation Results

This section outlines the simulation results for single- and multiple-zone(s) models.

A. Signal Zone

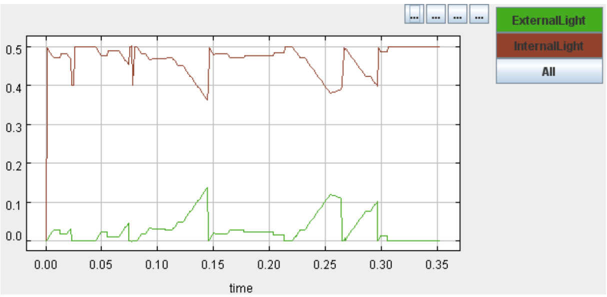

Figure 10. Internal Light vs. External Light

As shown in Fig. 10, when the external light intensity increases, the internal intensity decreases by the same value; when the external light intensity is equal to zero, the internal intensity is equal to the optimal light intensity, because there is no interference in a single zone model. In certain cases, the internal light intensity goes to 0.4 (with scaling = 400 Lux) without changes to the external light intensity. This happens when the automatic controller is switched off and the user changes the value to 0.4.

B. Multiple Zones

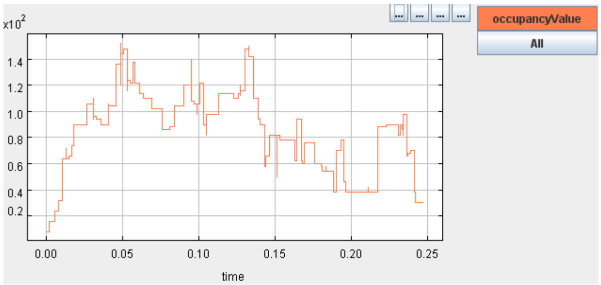

Figure 11. Occupancy Value

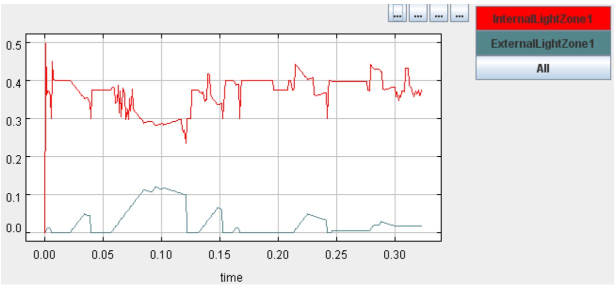

Figure 12. Internal Light vs. External Light for Zone 1

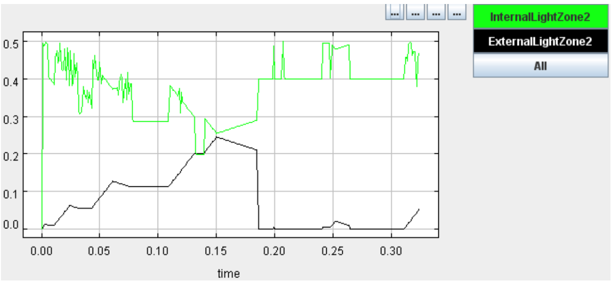

Figure 13. Internal Light vs. External Light for Zone 2

Fig. 11 shows that the occupancy value for the controlled zone changes randomly. In Figures 12 and 13, the internal light intensity in both zones change according to the external light intensity, but not by the same value because the internal light intensity is dependent on the interference term (as defined in Eq. 1).

VI. CONCLUSION

This article has shown how the use of Requirements-Driven modelling and Hybrid-Systems simulation, together with Model-Verification techniques, can provide a key approach for embedded models that are provably correct with respect to the requirements. In addition, this approach provides a clear mechanism to develop system interfaces and modify those interfaces as system modules change over time.

As an example of our approach, we have presented a simulation platform that successfully integrates lighting controls based on presence sensors, ambient light, area occupancy, user preferences, window blinding and multi-zones. This example incorporates occupant preferences via interfaces with a preference solver. We also showed methods for incrementally increasing the complexity of the system in a modular way.

In the future we intend to model more complex systems, such as HVAC systems. We also intend to implement demonstration of the developed system in the Environmental Research Institute (ERI) building, which is the ITOBO 'Living Laboratory' [11].

ACKNOWLEDGEMENT

This work was funded by SFI grant 06-SRC-I1091. The authors thank Hilmi Belhadjamor for his contribution in the first CHARON models for the lighting system.

REFERENCES

[1] T. Liebich, "International Alliance for Interoperability Modeling Support Group," IFC2x Edition2 Modeling implementation Guide, March 2004.

[2] D. Harel, "The STATEMATE semantics of statecharts," ACM Transactions on Software Engineering and Methodology (TOSEM), October 1996.

[3] http://zuse.ucc.ie/itobo/

[4] G Booch, J Rumbaugh and I Jacobson, "The Unified Modeling Language User Guide," Addison Wesley, 1998.

[5] K. R¨omer, O. Kasten and F.Mattern, "Middleware Challenges for Wireless Sensor Networks," ACM SIGMOBILE Mobile Computing and Communications Review, 2002.

[6] http://rtg.cis.upenn.edu/mobies/charon/index.html.

[7] Y. Hur and I. Lee, "Distributed Simulation of Multi-Agent Hybrid Systems.," IEEE International Symposium on Object-Oriented Real-time distributed Computing (ISORC), 2002.

[8] http://www.light.ie/lightlevels.htm

[9] B. playing, "POINT LIGHT SOURCES and LUMINOUS INTENSITY,"W. F. Long, 1992.

[10] R. Trebino, "Coherence, Incoherence, and Light Scattering," Georiga Institute of Technology.

[11] http://www.ucc.ie/en/ResearchandIndustry/OfficeoftheVPforResearch/Research/DocumentFile,16290,en.pbf

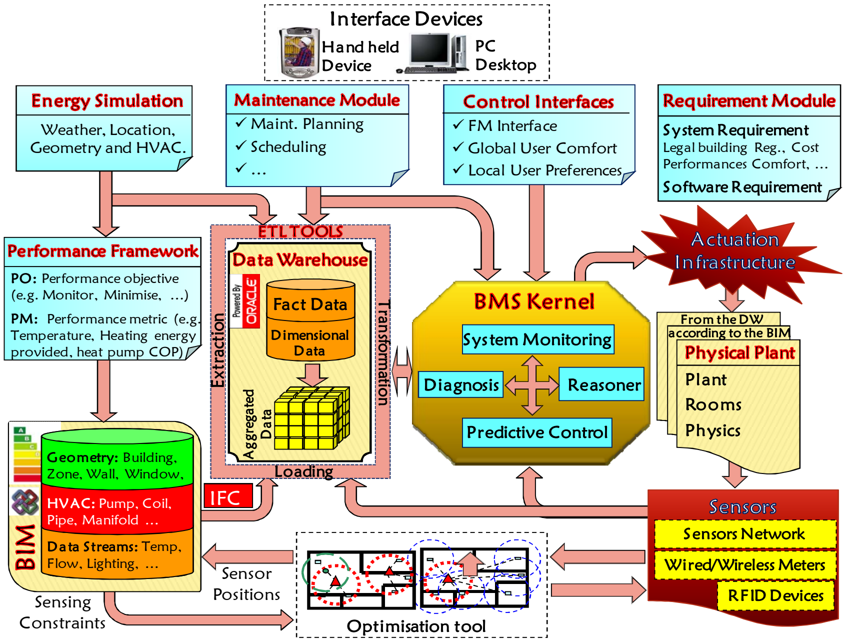

Figure 1. Preliminary ITOBO System Architecture