Real-Time Monitoring of High-Level States in Smart Environments

Real-Time Monitoring of High-Level States in Smart Environments

Fulvio CORNO [a,1] and Faisal RAZZAK [b]
[a] *Dipartimento di Automatica ed Informatica, Politecnico di Torino, Turin, Italy*
E-mail: fulvio.corno@polito.it
[b] *Dipartimento di Automatica ed Informatica, Politecnico di Torino, Turin, Italy*
E-mail: raja.faisal@gmail.com

Abstract
Modern smart environments are equipped with a multitude of devices and sensors aimed at intelligent services. The presence of these diverse devices has raised a major problem of managing complex environments. A rising solution to the problem is the modeling of user goals and intentions, and then interacting with the respective smart environments using user defined goals. Generally, the solution advocates that the user goal(s) can be represented by combining devices (smart appliances and sensor/actuators) in particular states. 'Domotic Effects' is a high level modeling approach, which provides Ambient Intelligence (AmI) designers and integrators with a high level abstract layer that enables the definition of user goals in a smart environment, in a declarative way, which can be used to design and develop intelligent applications. This paper describes an approach for the automatic evaluation of domotic effects combined through Boolean expressions, that can provide efficient and intelligent monitoring of the domotic structure of the environment. "Effects Evaluation" addresses the problem of finding the new values of all the domotic effects defined for the environment when one or more devices change their state or one or more sensor value is recorded in the environment, hence determining a new overall state of the environment. The paper also presents an architecture to implement the evaluation of domotic effects. Results obtained from carried out experiments prove the feasibility of the approach and highlight responsiveness of the implemented effect evaluation.

Keywords. Ambient Intelligence, Domotic Effects, Domotic Effect Evaluation, Higher level modeling, Monitoring Smart Environments

Introduction

The Ambient Intelligence (AmI) community identified that the provision of intelligent services lies at the core of smart environment systems, for which a wide array of devices are employed [16]. These intelligent services consist of automation of different services and provide comfort of living and easy management for users [11,12]. Bader et al. [1] described smart environments as heterogeneous dynamic ensembles: groups of co-located devices of different types. The presence of diverse devices has given rise to a major

[1] Corresponding Author: Dipartimento di Automatica ed Informatica, Politecnico di Torino, Torino, Italy
E-mail: fulvio.corno@polito.it

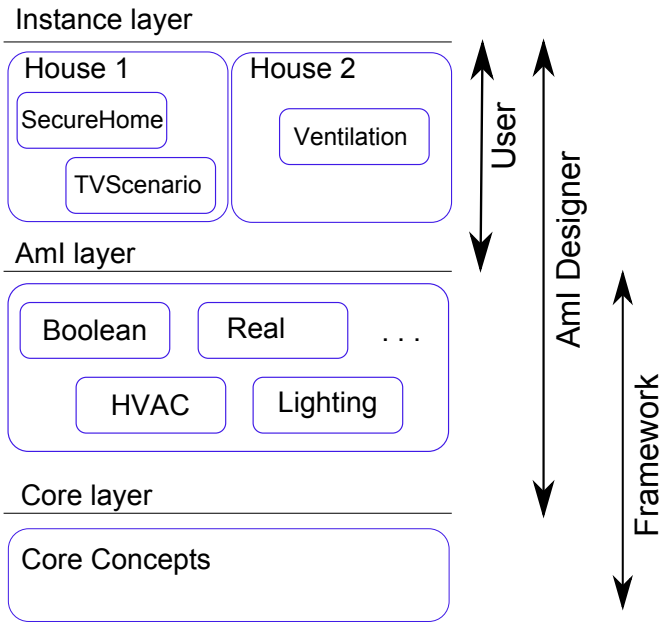

Figure 1. The Domotic Effects framework - Logic Architecture [9]

problem of managing smart environments in recent years. Current research is shifting focus from a traditional device-centric vision [13,17,20,21,24,26] to abstract modeling approaches [9,29,30] providing a higher level of design for interaction and control in smart environments [15].

In order to provide easy management of smart environments, a rising solution among abstract modeling approaches has been the modeling of user goals or intentions. Generally, the solution advocates the need to model user goals and then, to implement those goals by using a combination of devices installed in the environment. One such approach is the ontology based Domotic Effects (DE) framework [9,25] which models user goals or intentions (called *domotic effects*). The DE framework addresses the concerns of users as well as AmI designers. The users can program their personal environments by declaring their intentions and the achievement criteria for each intention are enforced by using a combination of devices. Most of the other approaches limit the ability of residents of a house or building to program their environment in terms of specific devices and their functionalities [17,18,21,22,24]. On the other hand, the AmI designers have an abstraction layer that enables the definition of generic goals inside the environment, in a declarative way, which can be used to design and develop intelligent applications: the AmI designer may choose the most suitable representation and define suitable functional operators. The logical architecture of the DE framework is shown in Figure 1.

The context is that every device in the environment is capable of providing certain visible (perceivable) effects for a user. These effects are fulfilled by possible states of the device. For example, an effect of *illumination* can be provided by a lamp in “On” state value or by a window shutter in “Open” state value. However, modern devices are complicated in nature and a single device can have a composite state, which may be modeled as concurrent sub-states. These sub-states are orthogonal regions combining multiple de-

scriptions of a device. For example, a TV set may have an on-off state (with possible values On or Off), a volume state (with a possible value between 0 and 100), a channel state (with a possible value depending on the set of programmed channels). A device state is therefore composite in nature and therefore it is modeled as the parallel composition of different sub-states. To clarify, this paper the term *state* refers to composite state of a single device.

There might be cases in which an effect can only be fulfilled by using a combination of devices having particular states. For example, the effect of *securing* a building may require all the exit doors and windows to be closed. In the context of DE framework, an effect that depends upon a single device (having a state or sub-states) is called a simple effect (SE) and an effect dependent on a combination of devices (having particular states and sub-states) is called a complex effect (CE). A CE is described by combining SEs and other CEs.

The DE modeling framework has a wide scope that aims, in the long term, at providing a unified approach for monitoring and controlling environments using user goals or intentions. The control aspect of the DE framework has been discussed [9] in the context of energy conservation. However, the ability of the DE framework to monitor the overall state of an environment has not been discussed yet. This paper expounds on the ability of the DE framework to monitor the overall state in real time (called *Effect Evaluation* in this paper). The monitoring amounts to correctly mapping the devices and their states in terms of user goals or intentions. It can provide users of smart environment the ability of monitoring the overall state of the environment in terms of more meaningful user intelligible goals instead of particular device states. Consider a house that needs to be secured at night. The *securing* of the house requires all doors and/or windows to be closed. If either by chance a door is left opened or a thief breaks a door open, then the resident of the house needs to know the implication of the door being opened at an abstract level. In this case, the door being left opened means that the house is not secure anymore or vice versa.

The discussion in this paper focuses on the applicability of the DE framework to Boolean application domains, i.e., domains in which the outcome of a user goal or intention can be either active (true) or inactive (false). This covers most monitoring use cases in smart homes, offices and industrial plants.

Formally, *Effect Evaluation* means computing the value of all domotic effects starting from the values of each device state and sensor value, according to the definition of the domotic effects and the semantics of the functional operators (that might change depending on the application domain). The computation should be in near real-time, i.e., comparable with the latency of home automation systems (roughly under one second), and well integrated with the smart environment system (e.g., a home gateway). The conception, architecture, implementation and possible application of *Effect Evaluation* are discussed in this paper.

The paper is divided into six sections. Section 1 provides overview of existing literature. Section 2 provides an overview of the underlying technologies. The problem addressed in this paper is formally defined in Section 3, while the general approach adopted for effect evaluation is described in Section 4, and later Section 4 also defines the adopted architecture and implementation. Section 5 shows results of the experiments and Section 6 concludes the paper.

1. Related Works

A neuro-cognitive model for Environment Recognition, Decision-making, and Action execution inside automated buildings is proposed in [28,29,30]. They introduce separate models for perception known as Artificial Recognition System-PerCeption (ARS-PC) and decision making, identified as Artificial Recognition System-PsychoAnalysis (ARS-PA). The perception models provide a three layered architecture, i.e., Micro Symbol Layer, Snapshot Symbol layer and Representation layer. Micro symbol layer are formed from sensor input data, which are combined to create snapshot symbols in the Snapshot symbol layer and then representation symbols are created corresponding to the perception of the system in the Representation layer by combining snapshot symbols. The representation symbols are then fed to the ARS-PA to perform a decision making process. In comparison, the "Domotic Effects" approach provides modeling of generic goals and their achievement criteria (DogEffects ontology). Besides perceiving and monitoring the environment, it has the ability to enforce and optimize those goals based on a possible set of paths [9]. Moreover, Domotic Effects provide separate views for system designers and users of the environment. System designers are allowed to define several different types of combinations for different application domains and users can define their own achievement criterias. This flexibility is missing from the neurocognitive model.

Hemrik Dibowski et al. [14] proposes an automatic design approach for large building automation systems (BAS). The top-down approach initiates by defining the structure of the building, then the system integrators define requirements using ontologies. The next step is to define abstract designs and required functionalities which are then transformed to detailed designs of the specific BAS. As indicated, the complexity of this transformation task can be very high, making the approach complicated and lacking flexibility to achieve detailed designs. On the other hand, the "Domotic Effects" approach is flexible in providing system designers separate working space independent of lower level details and allowing them to focus on general characteristics of the environment, which can later easily be extended or changed (AmI layer). The users have the ability to play a key role in programming their personal spaces.

While techniques addressing both the user and the system designer concerns are rare in literature, several papers have documented them separately. Rashidi et al. [24] propose a software architecture which incorporates learning techniques to discover patterns in user's daily activities. While the patterns of user's activities are observed and stored, the user can also define their own activity patterns as well. The activity pattern are observed as changes in states of devices occur in the house. Hierarchical activity model (HAM) is used to store discovered activity patterns and their temporal knowledge. The activities are discovered based on a device centric vision, which does not allow users to view the bigger picture about the state of the environment, in a concise way. On the other hand, Domotic Effects can provide the bigger picture about the environment concisely and in a manner understood by the user.

Cheng et al. [8] propose a smart home reasoning system called ASBR system. The system learns user's preferences by adaptive history scenarios and put forwards a way to rebuild reasoned knowledge in other smart homes. They propose that contextual information can be extracted and reasoned as a set of scenarios. In addition, the system can derive personal habits and store them in OWL files. Though it does provide an organization mechanism but here the concept of scenario is different from our proposed effect.

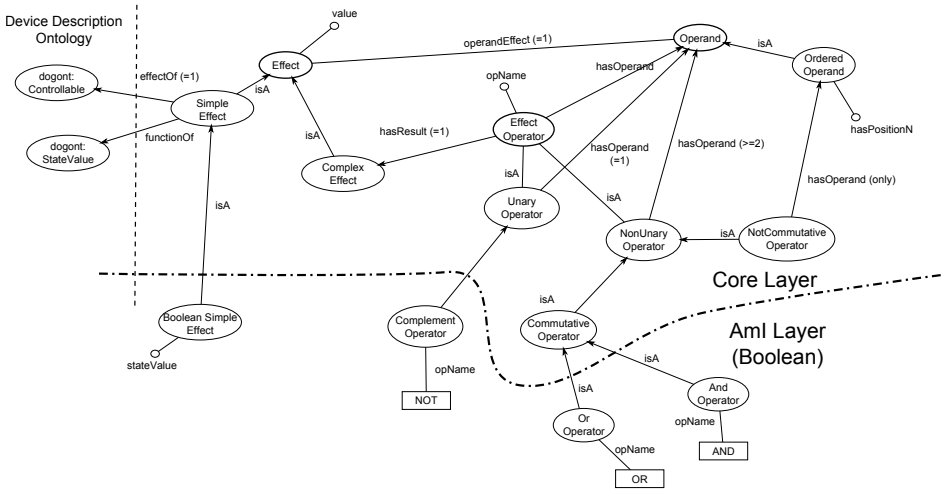

Figure 2. The DogEffects ontology (Core and AmI layers) - Boolean Application Domain

"Domotic Effects" are different from scenarios as it is not a storage of historical events or repetitive tasks, but a modeling of the environment as envisioned by the user. Therefore, during the process of monitoring the environment, the user has enhanced comprehension about his/her environment.

Salomons et al. [27] introduce a generic model for intelligent homes that describe the current state. the target state and the transition. In the context of monitoring the current state, a persona model is put forward. A persona is a model of individuals that share preferences. It stores the preferences in a second layer. The detailed description, type and frequency of the stored preferences are missing. Moreover, the approach seems to be based on storing preferences on individual devices which is contrary to the approach presented in this paper.

In the health-care domain, Esposito et al. [19] present a pervasive system for context-oriented monitoring purposes which operates on sensor data and is based on a versatile framework. The framework is organized according to a general purpose agent-based architecture centered around an ontological context representation. It identifies and monitors three kind of alarm events, i.e., Instantaneous, Interval and Multiparametric. Instantaneous and interval alarms are defined over individual sensors, while the multiparametric alarms are built by considering several parameters from multiple sensors. Though the instantaneous and interval alarms can be equated with simple effects and multiparametric can be compared with complex effects, the fundamental difference lies in the flexibility provided by the DE framework to define effect operators that govern combinations of simple and complex effects. Moreover, this paper presents a detailed ontological model (see Section 2) that allows domotic effect based monitoring to be applicable to several application domains. Last but not the least, the results presented in [19] are preliminary and the adopted approach takes time in seconds to monitor even simple scenarios. This paper presents experimentation in detail and the adopted approach monitors several complicated scenarios simultaneously and the response time is in few milliseconds (see Section 5).

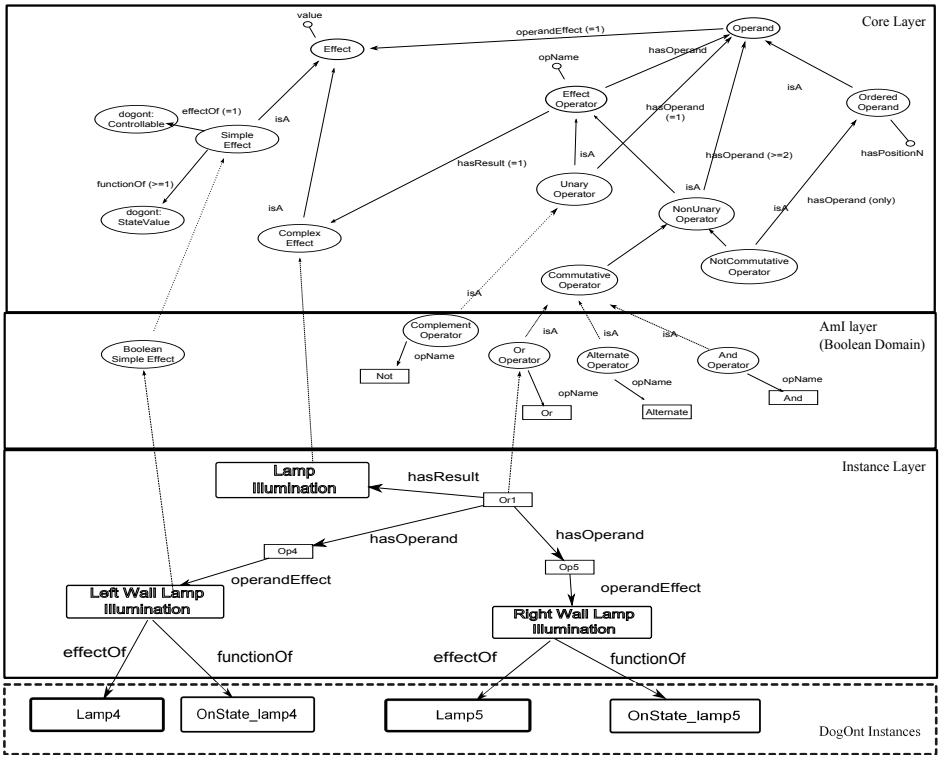

Figure 3. An example of defining a CE and SEs - Lamp Illumination use case

Lee et al. [23] present a ubiquitous system to monitor the real-time context of its individual objects and their collaboration in smart spaces. The system is called UMONS and it is able to monitor individual devices or sensors, and then aggregates the information coming from devices or sensors. In the end, the information is presented in a graphical format. The approach presented in [23] is device centric and does not allow real-time monitoring of multiple high-level states in smart spaces. UMONS is evaluated over well-being home services, but the quantitative results of the evaluation are not presented.

Personal spaces play an important role in group or individual self-definition: rather than just using them for a specific purpose, users pour their personalities and lives in the way they transform their environments [11]. “Domotic Effects” is one such framework that can enable the monitoring of the environment as the user wishes to perceive it, i.e., in terms of user-defined goals. Several other monitoring techniques have been proposed in the literature [6,32], but the “Domotic Effects” offer several advantages. First, being an ontology based approach offers large scale adoption, application development, system prototyping and a solid technological infrastructure [7]. Second, it offers a unified framework for modeling, controlling and monitoring the environment. The modeling and controlling aspects were discussed in [9] and this paper discusses the monitoring aspect. Third, the approach is robust (see Section 5) and scalable.

2. Foundations

This section briefly defines technologies and concepts lying at the foundation of the proposed approach in this paper.

2.1. User Goal Modeling: DogEffects Ontology

In order to represent the formal knowledge base of the *DE* framework, a 3-tiered ontology was proposed, called *DogEffects* ontology. The *DogEffects* ontology is formalized by using the OWL Web Ontology Language [31]. It models the user-defined goals and their mapping to devices and their corresponding states. As it requires the concepts of devices and their states, the modularity pattern was adopted for designing the *DogEffects* ontology. Modularity allows the ontology to easily integrate with various ambient ontologies that model environments.

Three modeling layers are defined, with decreasing usage complexity: a core layer, designed to establish the basic semantics of effects, an AmI-layer allowing AmI designers to define SEs and domain-dependent operators for combining them into CEs, and finally an instance layer where specific effects (both SEs and CEs) can either be defined by AmI designers or by final users.

2.1.1. Core Layer

The core layer defines concepts lying at the foundation of the *DE* framework. The main concepts of the core layer are depicted in Figure 2. Every *Domotic Effect* is formally organized into a concept hierarchy inheriting from the `Effect` class. As aforementioned, effects can either be simple (`SimpleEffect`) or complex (`ComplexEffect`). For both kinds of effects, domain-dependent subclasses can be defined in the AmI layer.

Simple Effects (SEs) are the terminal nodes of the representation and compute a value depending on a device state or sensor value. SEs act as interface points between the *DogEffects* ontology and some device description ontology (e.g., DogOnt [4]). The `effectOf` and `functionOf` open relations (i.e., relations without range restrictions) permit to identify the device and the device state for which a given SE is computed, respectively.

Every Complex Effect (CE) represents a functional expression of SEs and other CEs declared by using domain-dependent `EffectOperator` defined in the AmI layer of the DE framework. The nature of the effect operator depends upon the application domain, i.e., Boolean, Real or Energy Saving. Effect operators take either SE or CE as operands (through the `hasOperand` relation) and generate new CEs as result, identified by means of `hasResult` relation.

Furthermore, two main disjoint families of operators are modeled: unary operators (`UnaryOperator`) and non-unary operators (`NonUnaryOperator`).

2.1.2. AmI Layer

The AmI layer allows AmI designers to declare the set of effect operators that can govern the combinations of domotic effects depending on the application domain. As this paper focuses on Boolean application domains, the AmI layer (see Figure 2, bottom) has been populated with a basic set of Boolean operators, i.e., Complement, And, Or, and some derived Boolean operators as explained below:

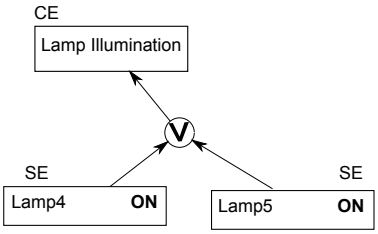

Figure 4. A simplified representation of the Lamp Illumination CE

1. *ImpliedOperator:* This operator represents the "logical implication" relationship.
2. *AlternateOperator:* This operator represents a function whose value is active when *exactly one* of its operands is active. It is commutative and non-unary. Mathematically, the Alternate effect operator can be defined as Eq. (1)

$$Alt(x_1 \ldots x_n) = \sum_i \left(x_i \cdot \prod_{j \neq i} \overline{x_j} \right). \tag{1}$$

3. *ExactlyMOperator:* This non-unary operator represents a function whose value is active when *exactly M* of its operands are active. Suppose there are n operands, i.e., $OP = \{1, 2, \ldots n\}$. Then the *ExactlyMOperator* effect operator can be defined as Eq. (2).

$$Exactly_M(x_1 \ldots x_n) = \sum_{O \subseteq OP, |P| = M} \left[\prod_{i \in O} x_i \cdot \prod_{j \notin O} \overline{x_j} \right] \tag{2}$$

2.1.3. Instance Layer

The Instance layer contains all the domotic effects (defined as an instance of DogEffects ontology) over a particular environment. A trivial "Lamp Illumination" use case is illustrated in Figure 3. The "Lamp Illumination" use case will be represented as a "Lamp Illumination" CE (an instance of `ComplexEffect`). Suppose the "Lamp Illumination" can either be achieved using "Left Wall Lamp Illumination" SE or "Right Wall Lamp Illumination" SE. The combination is governed by the "Or1" (instance of `OR Operator`) class. The "Left Wall Lamp Illumination" SE represents the "Lamp4" in "OnState_lamp4" state, while the "Right Wall Lamp Illumination" SE represents the "Lamp5" in "OnState_lamp5" state. The instance layer can be connected to any ambient ontology defining the conepts of devices and theirs states. In this paper, the devices and their states are represented as instances of the DogOnt ontology (see Section 2.3). In order to provide easy comprehension Figure 4 shows the simplified representation of the "Lamp Illumination" CE.

The simplified representation of an evolved "Dining@Lunch" CE is shown in Figure 5. The "Dining@Lunch" CE corresponds to the overall state of a dining room, in a house, during lunch hours. It includes isolating the kitchen ("Isolated Kitchen" CE), illuminating the dining room ("LampIllumination" CE) and switching on the television for entertainment ("TV_Entertainment" SE). The functional representation of the use case

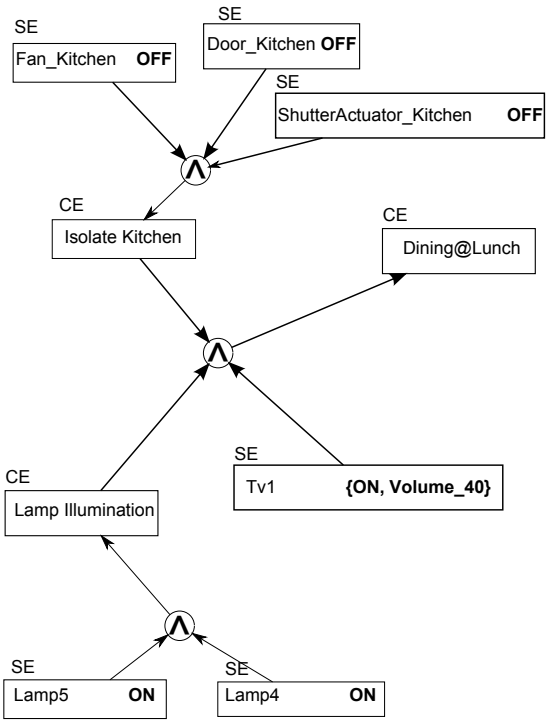

Figure 5. A simplified representation of the Dining@Lunch CE

Table 1. Dining@Lunch functional form

$Dining@Lunch = \mathbf{And}(IsolateKitchen, Tv\_Entertainment, LampIllumination)$
$IsolateKitchen = \mathbf{And}(Door\_Kitchen\_Isolate, Fan\_Off, Kitchen\_Flow)$
$LampIllumination = \mathbf{And}(LeftWallLampIllumination, RightWallLampIllumination)$
$RightWallLampIllumination = \mathbf{SE}(Lamp5, OnState\_lamp5)$
$LeftWallLampIllumination = \mathbf{SE}(Lamp4, OnState\_lamp4)$
$Door\_Kitchen\_Isolate = \mathbf{SE}(Door\_Kitchen, OffState\_Doorkitchen)$
$Fan\_Off = \mathbf{SE}(FAN\_Kitchen, OffState\_Fankitchen)$
$Kitchen\_Flow = \mathbf{SE}(ShutterActuator\_Kitchen, UpState\_SAKitchen)$
$Tv\_Entertainment = \mathbf{SE}(Tv1, OnState\_Tv1, Volume\_40)$

is outlined in Table 1. A SE is represented as **SE**(device, sub-state(s)). For instance, the "TV_Entertainment" SE is represented as TV_Entertainment = **SE**(Tv1, OnState_Tv1, Volume_40). On the other hand, a CE is represented as **Operator**($DE_1, DE_2 \ldots$). For instance, the " Dining@Lunch" CE is represented as Dining@Lunch = **And**(Isolate Kitchen, TV_Entertainment, Lamp Illumination).

2.2. Domotic Effects Formalization

This paper summarizes only the most important concepts about Domotic Effects. The reader interested in the full detail is referred to [25]

Given an intelligent ambient environment managing a set of devices $\mathscr{D}$, each device d is associated with a set of allowed states $S(d)$; depending on the nature of the device, states may be discrete (e.g., {On, Off} for a lamp) or continuous (e.g., [0, 100] for a volume knob). During system evolution, the actual state of each device is a time-dependent function $s(d,t) \in S(d)$.

The whole environment possesses a *global state space* $\mathscr{G}$, represented by the Cartesian product of all device state spaces: $\mathscr{G} = \prod_{d \in \mathscr{D}} S(d)$, thus defining a global environment state $g \in \mathscr{G}$.

Formally, a Domotic Effect *DE* is defined as a function of the global state space: $DE : \mathscr{G} \to \mathscr{V}$, where $\mathscr{V}$ is an application-dependent value space. For Boolean application domains, $\mathscr{V} = \{0,1\}$. A Simple Effect *SE* is a function that considers the state of only one device, $SE : S(d) \to \mathscr{V}$; such function is time-dependent since it depends on $s(d,t)$. An operator op is a function $\mathsf{op} : \mathscr{V}^N \to \mathscr{V}$, where *N* represents the number of operands of the specific op. A Complex Effect *CE* is represented by a couple $(\mathsf{op}, (DE_1 \dots DE_N))$ composed of an operator name op and a list of Domotic Effects DE_i whose values are used as operands. Such function is also time-dependent.

A set $\mathscr{I}$ contains all domotic effects defined for an environment, i.e., $\mathscr{I} = \{DE_1, DE_2 \dots DE_M\}$.

2.3. DogOnt

DogOnt [4] is an ontology for modeling smart environments; representing architectural elements (rooms, walls, etc.) of the environment, the devices placed inside the environment, and their states and functionalities [4]. It provides formal modeling and suitable reasoning capabilities through semantic web technologies. DogOnt has been designed along 7 main hierarchical trees, corresponding to the Building Thing, Building Environment, Functionality, Command, DomoticNetworkComponent, State, StateValue concepts. Classes descending from Building Environment and Building Thing are used to describe the environment. DogOnt provides the ability to model different devices that exist in a house through the Building Thing concept and DogOnt has a clear separation between the Controllable and Not Controllable devices. The *Functionality* class is related to the different functionalities of the device. The *Command* and *Notification* concepts are linked to specific functionality instances. Network modeling is achieved through DomoticNetworkComponent. State and StateValue are used to model the different states of a device.

3. Problem Statement

Given the definitions in the previous sections, the problem of *Effects Evaluation* deals with computing the value $\mathscr{V}$ for all the Domotic Effects DE_i defined in the instance layer of the "DogEffects" ontology, i.e., $\mathscr{I}$. The domain of a domotic effect value depends upon the application domain. As this paper addresses *Effects Evaluation* restricted to Boolean application domains, the value of a domotic effect (SE or CE) can either be true or false. The AmI layer encoded with Boolean operators is sufficient to model most control applications and many monitoring use cases in smart homes, offices, and automated industrial plants.

All the domotic effects combined represent the overall state of the environment at an abstract and user intelligible level. A change in the value of any one of the domotic effects will change the overall state of the environment. To put it simply, at any time instant the abstract state of the environment is represented by the set of domotic effects that are active. During the course of operation, different devices change their states and different sensor values are recorded. Each time a device changes its state, the values of several domotic effects dependent on the device might change their values from true to false or vice versa, therefore triggering a change in overall state of the environment.

For a SE, the value is active when the corresponding device is in the state defined for the SE. A CE is a combination of other domotic effects (both simple and complex) and the type of combination is governed by an effect operator defined in the instance layer of the DogEffects ontology. Therefore, the value is dependent upon the values of its children and the effect operator associated with the CE.

The primary objective of the paper is to design, develop and verify a software module that performs the process of *Effect Evaluation* in the context of the *DE* framework. Since the module needs to be verified inside a smart environment system, the *Effect Evaluation* module should meet the following set of requirements:

1. The module should be able to monitor the change of state of all the devices installed in the environment.
2. Different third-party applications should be able to register themselves to this module, to listen to any change in the values of the domotic effects.
3. During the *Effect Evaluation* process, the module should asynchronously send out notifications to the registered applications.
4. The process of *Effect Evaluation* should be robust without the user noticing any delay of the monitoring service, i.e., to respond in near real-time.
5. The module should provide integration with a smart environment system.

4. Solution

4.1. Approach

In order to meet the objectives of the paper a module named "Domotic Effects Evaluation" was developed. All domotic effects are organized in a hierarchical structure that corresponds to the logical structure defined in the Instance layer of the DogEffects ontology. The structure is called Effect Node Network (ENN) and each domotic effect is represented as a node in the ENN. For the *Dining@Lunch* use case defined in Section 2, the ENN is shown in Figure 6.

The module listens to any change in state of the devices and if any change occurs, it performs the process of *Effect Evaluation*. The process of *Effect Evaluation* means computing new values of all the domotic effects (SEs and CEs). In order to efficiently compute new values of domotic effects in the ENN, the classical Zero Delay Event Driven Logic Simulation algorithm [5] is chosen to perform the evaluation. The algorithm was chosen because of the property of having linear execution time in the number of nodes and taking into account that all operators are time independent.

In the *DE* framework, the effect operators defined in the AmI layer represent the evaluation criteria. Therefore, a proper software implementation of each effect operator in the Boolean application domain is provided, i.e., Complement, And, Or and Alternate.

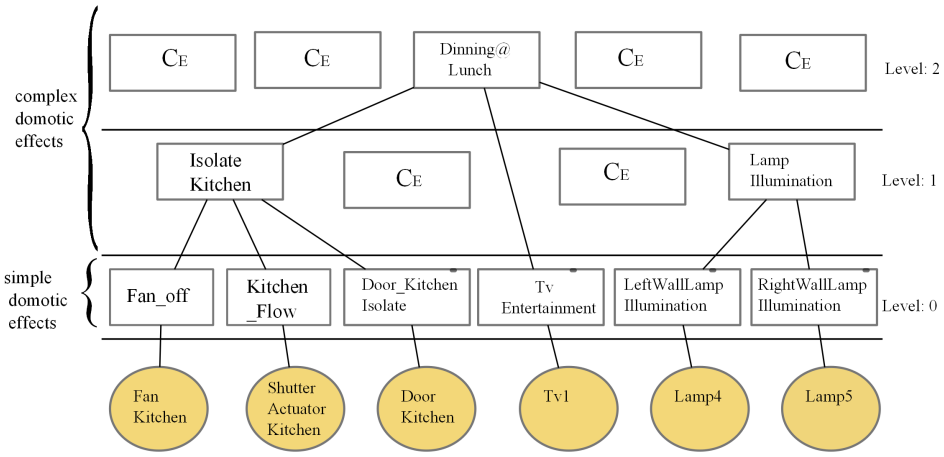

Figure 6. ENN for the Dining@Lunch use case

When a device changes its state, the values of the SEs associated with the device may change. If the values of SEs are changed, the evaluation of the CEs dependent on them are scheduled and recursively evaluated. During the evaluation, if a change in the value of a domotic effect (simple or complex) is observed, an asynchronous notification is sent out to interested third-party applications. This whole process is called *Effect Evaluation*. For example, considering the ENN shown in Figure 6 at any instant, *Lamp5* changes its state from *off* to *on*. It would possibly require the evaluation of all domotic effects in the ENN. The value of the *RightWallLampIllumination* SE will become "true" and therefore, the values of all the CEs dependent on *RightWallLampIllumination* SE would need to be recomputed, i.e., in this case only *LampIllumination* CE. Again, if the value of *LampIllumination* CE changes (different from previous value) then the values of all the dependent CEs need to be recomputed, otherwise the values of all the dependent CEs need not be recomputed. This recomputation of values is performed for the complete ENN whenever the value of a SE is changed.

The *Effect Evaluation* approach being proposed in this paper is designed to be generic and it can be adopted across different ontology driven smart environment systems. The generalization comes from the fact that the approach is based on the modular *DogEffects* ontology which can be integrated with other ambient ontologies having at least two concepts, corresponding to *devices* and their *states* (in our case, DogOnt is adopted). In order to gauge the proposed approach according to different criterias (performance and integration), the *Domotic OSGi Gateway (Dog)* was chosen as a smart environment system. Dog [3] is an ontology-powered Domotic OSGi Gateway that is able to expose different domotic networks as a single, technology neutral, home automation system. It is versatile as it is built on top of the OSGi framework and the adoption of semantic modeling techniques allows Dog to support intelligent operations inside the home environment. Dog uses the DogOnt ontology to model an environment and it provides the ability to overcome issues like interoperability among different device/sensor protocols, validating device state or sensor value, etc. However, other ontology driven smart environment systems may also be adopted if they provide the ability to represent devices/sensors, their states/values and the ability to monitor the state/value of a device.

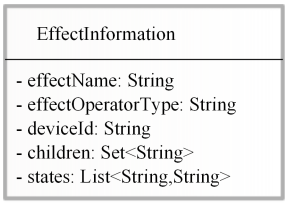

Figure 7. Information template of each domotic effect

4.2. Architecture

The Dog architecture was extended to integrate the *DE* based approach in smart environments. Dog has a modular architecture composed of 12 core bundles, among which the *HouseModel* bundle is responsible for managing the DogEffects and DogOnt ontologies. The *HouseModel* bundle provides information about all the domotic effects defined in the environment. A *DogState* bundle monitors the state of the devices installed in the environment and a *Domotic Effects Evaluation* bundle was developed, which organizes all the domotic effects in the ENN and carries out the effect evaluation process at runtime. Figure 1 shows the architecture of the proposed solution and highlights the components of the *Domotic Effects Evaluation* bundle. The functionalities of three three bundles are explained below.

4.2.1. HouseModel Bundle

The *HouseModel* bundle exploits standard DogOnt classes and DogOnt instances referred by a specific environment to provide knowledge-rich access to the environment properties and capabilities. It also accesses the DogEffects ontology classes and their instances, and provides all the management operations related to the DogEffects ontology. At Dog startup, inference and reasoning tasks are carried out at this level for computing consistency checking and transitive closure, at runtime. Moreover, the bundle gives the information associated with all the domotic effects defined in the environment to the *Domotic Effects Evaluation*. In case of a CE, the associated information includes the name of the effect, the name and number of children, the type of the domotic effect (defined using the Effect operator). In case of a SE, the name of the effect and its associated device in a given state is mentioned. The information regarding each domotic effect provided to the *Domotic Effects Evaluation* at startup is shown in Figure 7 as a class template.

4.2.2. DogState Bundle

The *DogState* bundle is responsible for providing the *Domotic Effects Evaluation* bundle information about the current states of the devices connected to the Dog platform. Whenever a device changes its state, it sends out device state change notifications containing the name of the device and its current state.

4.2.3. Domotic Effects Evaluation Bundle

This bundle is responsible for performing the effect evaluation process, whenever a device changes its state. The logical architecture of this bundle is depicted in Figure 8. It consists of an *Organization* component, a *Simulator* and an *Effect Operator Store*.

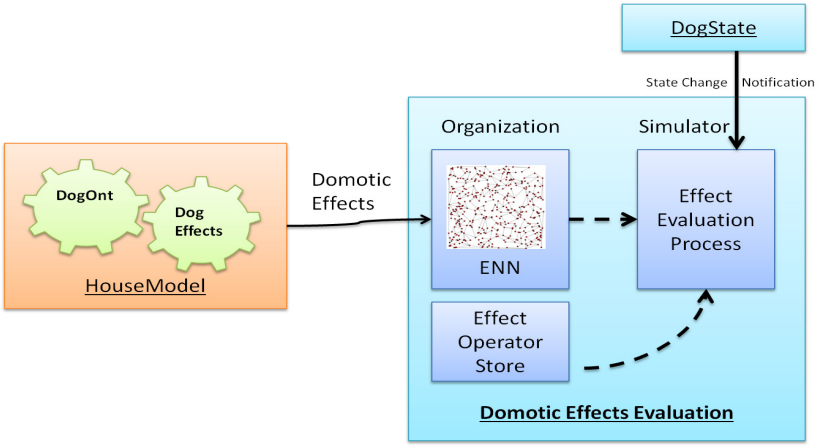

Figure 8. Logical architecture of Domotic Effects Evaluation bundle

During Dog startup, the *Organization* component requests and receives all the domotic effects defined in the instance layer for a particular environment. It parses the received response from the *HouseModel* and organizes all the domotic effects in the ENN.

As defined in Section 4.1, to evaluate different types of domotic effects, proper evaluation algorithms (to achieve the activation criteria) for the effect operators defined in the AmI layer are needed. The *Effect Operator Store* contains the Java code for each effect operator, which provides semantics to perform the evaluation for the effect operator.

The effect evaluation process is managed by the Simulator component and comprises receiving notifications of device state change from the *DogState* bundle, evaluating all the domotic effects in the ENN and sending out notifications if changes in the values of domotic effects are observed.

The following steps are followed:

- When a device state change notification is received, the simulator matches the received notification to the SEs associated with the device to check their activation criteria. If the evaluation criteria matches, it activates the SE and deactivates other SEs associated with the device. Afterwards, all the CEs dependent on the SEs are evaluated.
- In order to compute the value of a CE, the values of its children (SEs or CEs) are determined and then the type of the effect operator. Once the effect operator is determined, the appropriate Java implementation in the effect operator store takes the values of its children and determines the value of the current CE.
- The classical Zero Delay Event Driven Logic Simulator Algorithm has been chosen to perform the effect evaluation process on the ENN. The transformed algorithm for our proposed approach, in pseudo-code is reported in Algorithm 1. Basically, Algorithm 1 deals with computing new value of an effect (SE or CE) using *evaluate* method. If there is a change in the value of the effect, then the Algorithm 1 schedules the evaluation of all CEs (in next level) dependent on the current effect. The semantics of the *schedule* method, in pseudo-code, is reported in Algorithm 2. This process is repeated for the complete ENN.

- Furthermore, during the evaluation if a change in value of a domotic effect is observed, a notification is issued with the domotic effect and its new computed value.

Algorithm 1 Zero Delay Event Driven Logic Simulator Algorithm

```
for all queues as currentqueue do
    for all currentqueue as node do
        oldValue=(node).isValue();
        newValue = (node).evaluate(); → Evaluation Criteria for the node
        if oldValue NOT_EQUAL newValue then
            node.setValue(newValue);
            node.sendNotification();
            node.schedule();
        end if
    end for
end for
```

Algorithm 2 schedule method Algorithm

```
parents=this.getParents();
for all parents as node do
    p=(node).getLevel();
    queue=(queues).get(p);
    if queue.contains(node) == false then
        queue.add(node);
    end if
end for
```

4.3. Extensibility

The Domotic Effects approach is extensible, and AmI designers have the ability to define new and different Boolean operators depending on the environment. Appropriate implementation of new effect operators should be included in the Effect Operator store in order to perform *Effect Evaluation*. Suppose an AmI designer wants to declare a new Boolean effect operator called 'NewBooleanEffectOperator'. Then, in addition to defining the new effect operator in the AmI layer of the DogEffects ontology, the AmI designer must also provide a proper implementation of the new Boolean effect operator in the Effect Operator Store. A new 'NewBooleanEffectOperator' class must be implementation by extending the abstract *EffectNode* class. The semantics for the evaluation must be provided using the *evaluate* method.

When a new Boolean effect operator is defined in the AmI layer of the "DogEffects" ontology, the implementation of the new effect operator can be provided inside the *Domotic Effect Evaluation* bundle by two steps:

1. A new class extending the abstract *EffectNode* class should be implemented. The class diagram of the *EffectNode* class is shown in Figure 9.
2. The *EffectNode* class gives the ability to access values of the children nodes and based on them, a proper condition (implementation) for activation (true) or deactivation (false) must be provided by implementing the abstract *evaluate* method.

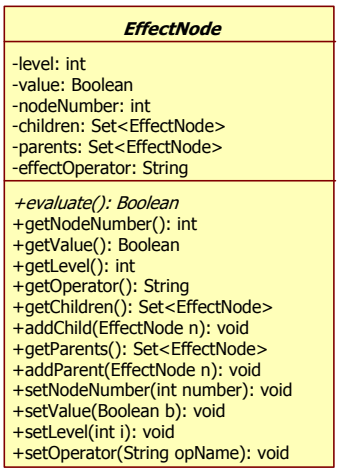

Figure 9. Template of the abstract EffectNode Class

Currently, the *effect operator store* includes the Complement, And, Or and Alternate operators. The Complement, Or and And effect operators are mapped as Not, Or and And Boolean operators, respectively.

The Alternate effect (Algorithm 3) operator represents a function which is active when only one of its children is active.

Algorithm 3 Alternate evaluation algorithm

```
children=node.getChildren();
counter=0;
for all children as childNode do
    if childNode.isValue() == true then
        counter= counter + 1 ;
        if counter > 1 then
            return false;
        end if
    end if
end for
if counter == 1 then
    return true;
else
    return false;
end if
```

5. Experimental Study

To study the feasibility of the proposed approach and measure performance parameters of the proposed approach, two sets of experiments were carried out. In the first set of experiments the integration of the proposed approach inside the Dog was assessed. Conversely, in the second set of experiments performance parameters were measured. These parameters include the time taken by the *Domotic Effect Evaluation* bundle to request, receive and organize the domotic effects in the ENN during Dog startup and the average time taken to completely perform the effect evaluation process.

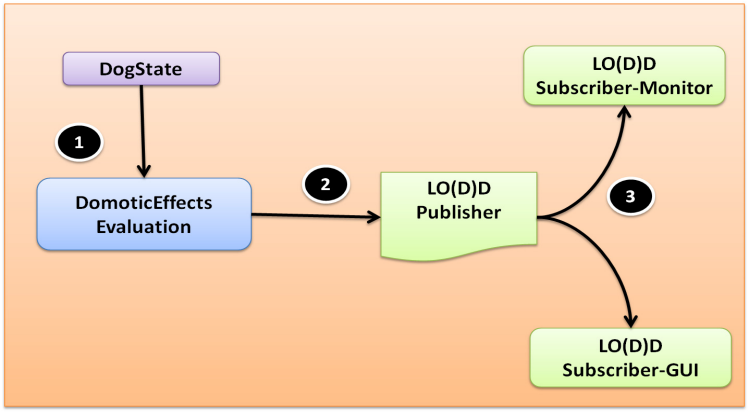

Figure 10. Action Sequence of Feasibility Testing.

5.1. Feasibility Testing

A set of standalone applications, based on the Publisher-Subscriber pattern [2], were developed to test the correctness of integration and to witness the overall effect evaluation process.

After modifying the *HouseModel* bundle and integrating the *Domotic Effects Evaluation* bundle, in order to realize the Publisher-Subscriber pattern over the web the *LO(D)D* architecture was used. LO(D)D stands for Linked Open (Dynamic) Data and is a distributed framework that provides a systematic way to publish environment data which is being updated continuously; such updates might be issued at specific time intervals or bound to some environment specific event [10]. In our case, these events are changes in the values of domotic effects, defined in the environment.

The architecture of the feasibility testing is shown in Figure 10. The testing was conducted over a simulated environment of a house whose structure is shown in Figure 11. The house is composed of a bed room, a living room, a lobby, a bath room, a store and a kitchen, and is equipped with several automatic devices/appliances like lamps, oven, television, door actuators, window actuators, shutter actuators, gas heaters etc. For each device, a certain number of simple domotic effects have been defined in the instance layer that correspond to the number of observable states that the device can achieve. Based on these simple domotic effects and using the set of Boolean operators encoded in the AmI layer, several complex domotic effects were defined.

5.1.1. Use cases

Based on the functional representation form presented in Section 2.1, this section describes six top level use cases defined over the simulated house environment, using simple and complex domotic effects. The number of total domotic effects defined to map all six use cases were 190, and the number of involved devices were 57. To check the integration, several iterations were performed. In each iteration, the states of random number of devices were changed and the corresponding changes in the values of domotic effects were monitored. Table 8 summarizes the results of the interations. It defines, for each iteration, the number of devices whose states were changed and the corresponding number of active and inactive domotic effects (after the change).

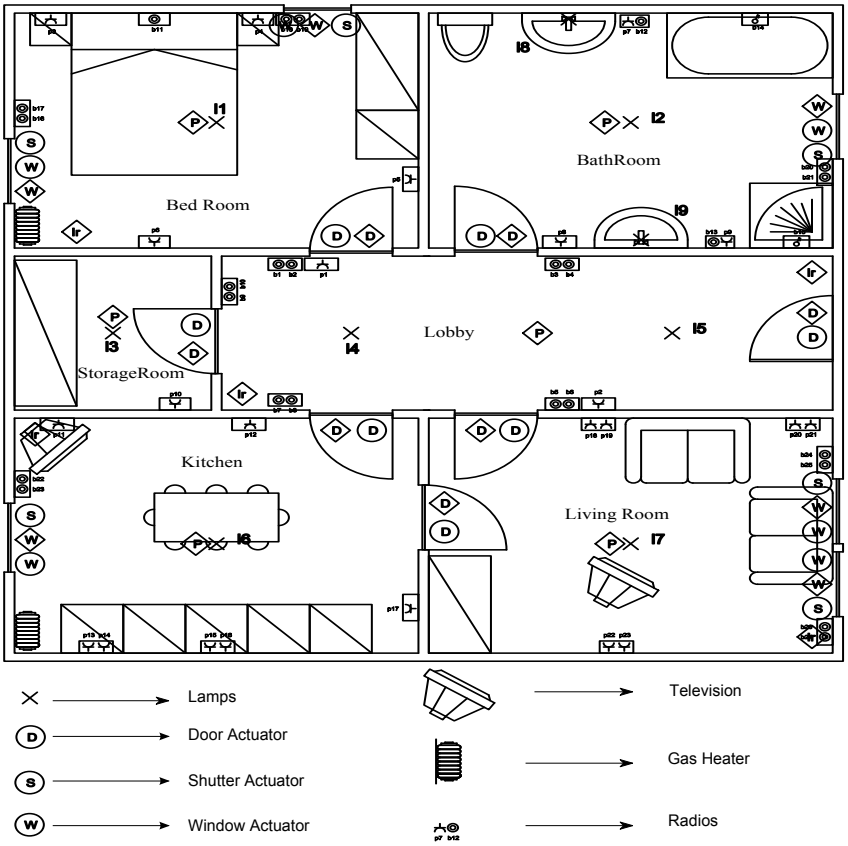

Figure 11. A Sample Structure of a house.

To perform the experiment, a new *LO(D)D Publisher* bundle was built which registers to the *Domotic Effects Evaluation* bundle for receiving notifications when the value of any domotic effect is changed. Two web applications, i.e., *LO(D)D Subscriber-Monitor* and *LO(D)D Subscriber-Graphical* were developed, which subscribed to the *LO(D)D Publisher* bundle. The former provided a command line representation, while the latter provided a graphical representation.

A test bundle also sent commands to Dog in order to change the states of different devices. Meanwhile, the *Domotic Effects Evaluation* bundle listened for any change in state of devices installed in the environment (from the DogState bundle). As it received the state change notification from the *DogState* bundle the effect evaluation process is performed and new values of all the domotic effects are computed. During the evaluation process, when it encounters a change in the value of a domotic effect, a notification is asynchronously sent to the registered applications, i.e., in our case *LO(D)D Publisher* bundle. The *LO(D)D Publisher* bundle forwards the notifications to both *LO(D)D Subscribers*.

The following use cases were defined in the experiment.

Secure Home By closing all the exit doors and shutting all the windows of the house, a "Secure Home" use case can be achieved since all the exit points of the house are closed.

Table 2. Secure Home use case

$SecureHome\_All = \mathbf{OR}(SecureHome\_Scenario\_1, SecureHome\_Scenario\_2);$
$SecureHome\_Scenario\_1 = \mathbf{AND}(LivingRoom\_l2\_WS\_CloseDown,$
$BathRoom\_WS\_CloseDown, Kitchen\_WS\_CloseDown,$
$BedRoom\_l1\_WS\_North\_CloseDown, BedRoom\_WS\_West\_CloseDown,$
$DoorActuator\_d4\_lobby\_ext\_Close, LivingRoom\_L1\_WS\_CloseDown);$
$LivingRoom\_l2\_WS\_CloseDown = \mathbf{AND}(WindowActuator\_w6\_living\_Close,$
$ShutterActuator\_sh2\_living\_Down);$
$LivingRoom\_L1\_WS\_CloseDown = \mathbf{AND}(WindowActuator\_w5\_living\_Close,$
$ShutterActuator\_sh1\_living\_Down);$
$BathRoom\_WS\_CloseDown = \mathbf{AND}(WindowActuator\_w3\_bath\_Close,$
$ShutterActuator\_bath\_Down);$
$BedRoom\_l1\_WS\_North\_CloseDown = \mathbf{AND}(WindowActuator\_w1\_living\_Close,$
$ShutterActuator\_sh1\_Down);$
$BedRoom\_WS\_West\_CloseDown = \mathbf{AND}(WindowActuator\_w2\_Close,$
$ShutterActuator\_sh2\_Down);$
$SecureHome\_Scenario\_2 = \mathbf{AND}(Secure\_LivingRoom, Secure\_BedRoom, Secure\_BathRoom,$
$Secure\_Lobby, Secure\_Kitchen);$
$Secure\_LivingRoom = \mathbf{AND}(DoorActuator\_d7\_kitchen\_Close,$
$DoorActuator\_d6\_living\_Close, Tv\_LivingRoom\_Off,$
$LivingRoom\_L1\_WS\_CloseDown, LivingRoom\_l2\_WS\_CloseDown);$
$Secure\_BedRoom = \mathbf{AND}(DoorActuator\_d1\_bed\_Close, BedRoom\_l1\_WS\_North\_CloseDown,$
$BedRoom\_WS\_West\_CloseDown);$
$Secure\_BathRoom = \mathbf{AND}(BathRoom\_Ws\_CloseDown, DoorActuator\_d2\_bath\_Close);$
$Secure\_Lobby = \mathbf{AND}(DoorActuator\_d6\_living\_Close, DoorActuator\_d5\_kitchen\_Close,$
$DoorActuator\_d4\_lobby\_ext\_Close, DoorActuator\_d3\_lobby\_stor\_Close,$
$DoorActuator\_d1\_bed\_Close, DoorActuator\_d2\_bath\_Close);$
$Secure\_Kitchen = \mathbf{AND}(DoorActuator\_d5\_kitchen\_Close,$
$DoorActuator\_d7\_kitchen\_Close, Kitchen\_WS\_CloseDown);$
$Kitchen\_WS\_CloseDown = \mathbf{AND}(WindowActuator\_w4\_kitchen\_Close, ShutterActuator\_kitchen\_Down);$

This use case comprises many DEs providing the ability to secure different rooms of the house. This can be used in case of emergency, theft, robbery or fire etc. The functional representation is shown in Table 2.

BathRoom Illumination The "BathRoom Illumination" combines small use cases that illuminate the bathroom. The illumination can be artificial by switching on the mirror lamps or ceiling lamp in different combinations, or illumination can be natural by opening the shutter of the window during morning and afternoon hours. The functional representation is given in Table 3.

Home Illumination The "Home Illumination" requires that all the rooms of the house are illuminated. Illumination can be both natural or artificial in nature. The functional representation is shown Table 4.

Afternoon Lunch In the afternoon, the resident desires to cook food and therefore, the kitchen's oven to be heated, the television to be switched on and the kitchen to be closed so that the aroma of cooking does not spread to other rooms of the house. This scenario represents "Afternoon Lunch" use case. The functional representation is given in Table 5.

Table 3. Bathroom Illumination functional form

$Illumination = \mathbf{Or}(ArtificialIllumination, NaturalIllumination)$
$ArtificialIllumination = \mathbf{Alternate}(CeilingLampIllumination, MirrorLampIllumination)$
$MirrorLampIllumination = \mathbf{And}(LeftMirrorLampIllumination, RightMirrorLampIllumination)$
$RightMirrorLampIllumination = \mathbf{SE}(l9, OnState\_lamp9)$
$LeftMirrorLampIllumination = \mathbf{SE}(l8, OnState\_lamp8)$
$CeilingLampIllumination = \mathbf{SE}(l2, OnState\_lamp2)$
$NaturalIllumination = \mathbf{SE}(ShutterBath, UpStateValue\_ShutterBath)$

Table 4. Home Illumination use case

$Home\_Illumination = \mathbf{OR}(Natural\_Illumination\_Home, Artificial\_Illumination\_Home)$;
$Natural\_Illumination\_Home = \mathbf{AND}(BedRoom\_Natural\_Illumination,$
$DoorActuator\_d7\_kitchen\_Open, DoorActuator\_d6\_living\_Open, BathRoom\_WS\_CloseUp,$
$DoorActuator\_d5\_kitchen\_Open, DoorActuator\_d1\_bed\_Open,$
$LivingRoom\_Natural\_Illumination, Kitchen\_WS\_CloseUp)$;
$BedRoom\_Natural\_Illumination = \mathbf{AND}(BedRoom\_WS\_West\_Closeup, BedRoom\_l1\_WS\_North\_CloseUp)$;
$BedRoom\_l1\_WS\_North\_CloseUp = \mathbf{AND}(WindowActuator\_w1\_living\_Close, ShutterActuator\_sh1\_Up)$;
$BedRoom\_WS\_West\_CloseUp = \mathbf{AND}(WindowActuator\_w2\_Close, ShutterActuator\_sh2\_Up)$;
$LivingRoom\_Natural\_Illumination = \mathbf{OR}(LivingRoom\_l1\_WS\_CloseUp, LivingRoom\_l2\_WS\_CloseUp)$;
$LivingRoom\_l1\_WS\_CloseUp = \mathbf{AND}(WindowActuator\_w5\_living\_Close, ShutterActuator\_sh1\_living\_Up)$;
$LivingRoom\_l2\_WS\_CloseUp = \mathbf{AND}(WindowActuator\_w6\_living\_Close, ShutterActuator\_sh2\_living\_Up)$;
$Artificial\_Illumination\_Home = \mathbf{AND}(Lobby\_Illumination, Lamp6\_Kitchen\_On,$
$artificiallyIlluminatedBath, Lamp1\_BedRoom\_On, Lamp7\_LivingRoom\_On)$;
$Lobby\_Illumination = OR(Lobby\_Illumination\_All, Lobby\_Illumination\_Alternate)$;
$Lobby\_Illumination\_Alternate = \mathbf{ALTERNATE}(Lamp4\_Lobby\_On, Lamp5\_Lobby\_On)$;
$Lobby\_Illumination\_All = \mathbf{AND}(Lamp4\_Lobby\_On, Lamp5\_Lobby\_On)$;
$artificialIllumination = \mathbf{ALTERNATE}(celingLamp\_On, MirrorLampsOn)$;
$MirrorLampsOn = \mathbf{AND}(Lamp9\_On, Lamp8\_On)$;

Table 5. Afternoon Lunch Cooking use case

$Afternoon\_Lunch = \mathbf{AND}(Oven\_Kitchen\_On, Tv\_Kitchen\_On, Kitchen\_CookingDay\_Scenario\_Alt)$;
$Kitchen\_CookingDay\_Scenario\_Alt = \mathbf{ALTERNATE}(Kitchen\_CookingDay\_Scenario\_1,$
$Kitchen\_CookingDay\_Scenario\_2)$;
$Kitchen\_CookingDay\_Scenario\_1 = \mathbf{AND}(ExhaustFan\_On, DoorActuator5\_Close, DoorActuator7\_Close,$
$Lamp6\_Off, Kitchen\_WS\_Day\_Scenario)$;
$Kitchen\_WS\_Day\_Scenario = \mathbf{ALTERNATE}(Kitchen\_WS\_CloseUp, Kitchen\_WS\_OpenDown)$;
$Kitchen\_WS\_CloseUp = \mathbf{AND}(WindowActuator\_Kitchen\_Close, Shutter\_Kitchen\_Up)$;
$Kitchen\_WS\_OpenDown = \mathbf{AND}(WindowActuator\_Kitchen\_Open, Shutter\_Kitchen\_Down)$;
$Kitchen\_CookingDay\_Scenario\_2 = \mathbf{AND}(ExhaustFan\_On, DoorActuator5\_Close, DoorActuator7\_Close,$
$Lamp6\_On, Kitchen\_WS\_CloseDown)$;
$Kitchen\_WS\_CloseUp = \mathbf{AND}(WindowActuator\_Kitchen\_Close, Shutter\_Kitchen\_Down)$;

Air Passage inside the house The opening of windows at the opposite sides of the house, enable the flow of air between different rooms of the house. Hence, it represents the "Air Passage" use case. The functional representation is shown in Table 6.

Table 6. Air Passage use case

$AirPassage\_All = \mathbf{AND}(AirPassage\_LRBR\_Scenario\_1, AirPassage\_LRBR\_Scenario\_2,$
$AirPassage\_LRKT\_Scenario\_1, Door\_Kitchen\_d5\_Open)$;
$AirPassage\_LRBR\_Scenario\_1 = \mathbf{AND}(LivingRoom\_Windows\_Open\_Any,$
$DoorActuator\_d6\_living\_Open, DoorActuator\_d1\_Bed\_Open, BedRoom\_l1\_WS\_North\_OpenUp)$;
$LivingRoom\_Windows\_Open\_Any = \mathbf{OR}(LivingRoom\_Windows\_Open\_Alternate,$
$LivingRoom\_Windows\_Open)$;
$LivingRoom\_Windows\_Open\_Alternate = \mathbf{ALTERNATE}(LivingRoom\_Windows\_North\_Open,$
$LivingRoom\_Windows\_South\_Open)$;
$LivingRoom\_Windows\_South\_Open = \mathbf{NOT}(LivingRoom\_Windows\_North\_Open)$;
$LivingRoom\_Windows\_North\_Open = \mathbf{AND}(LivingRoom\_l1\_WS\_OpenUp, LivingRoom\_l2\_WS\_CloseDown)$;
$LivingRoom\_l1\_WS\_OpenUp = \mathbf{AND}(WindowActuator\_w5\_living\_Open, ShutterActuator\_sh1\_living\_Up)$;
$LivingRoom\_l2\_WS\_CloseDown = \mathbf{AND}(WindowActuator\_w6\_living\_Close, ShutterActuator\_sh2\_living\_Down)$;
$LivingRoom\_Windows\_Open = \mathbf{AND}(LivingRoom\_l2\_WS\_OpenUp, LivingRoom\_l1\_WS\_OpenUp)$;
$LivingRoom\_l2\_WS\_OpenUp = \mathbf{AND}(WindowActuator\_w6\_living\_Open, ShutterActuator\_sh2\_living\_Up)$;
$LivingRoom\_l1\_WS\_OpenUp = \mathbf{AND}(WindowActuator\_w5\_living\_Open, ShutterActuator\_sh1\_living\_Up)$;
$BedRoom\_l1\_WS\_North\_OpenUp = \mathbf{AND}(WindowActuator\_w1\_living\_Open, ShutterActuator\_sh1\_Up)$;
$AirPassage\_LRBR\_Scenario\_2 = \mathbf{AND}(DoorActuator\_d1\_bed\_Open, LivingRoom\_Windows\_Open\_Any,$
$DoorActuator\_d6\_living\_open, BedRoom\_WS\_West\_OpenUp)$;
$BedRoom\_WS\_West\_OpenUp = \mathbf{AND}(WindowActuator\_w2\_Open, ShutterActuator\_sh2\_Up)$;
$AirPassage\_LRKT\_Scenario\_1 = \mathbf{AND}(LivingRoom\_Windows\_Open\_Any, Kitchen\_WS\_OpenUp,$
$DoorActuator\_d7\_kitchen\_Open, ExhaustFan\_Kitchen\_On)$;
$Kitchen\_WS\_OpenUp = \mathbf{AND}(WindowActuator\_w4\_kitchen\_Open, ShutterActuator\_kiitchen\_Up)$;

Table 7. Morning Wake Up use case

$Morning\_WakeUp = \mathbf{AND}(BathRoomIllumination, Radio\_BathRoom\_On, Tv\_Kitchen\_On,$
$BedRoom\_Natural\_Illumination, Kitchen\_Cooking\_Day\_Scenario\_1, GasHeater\_BedRoom\_On)$;
$BathRoomIllumination = \mathbf{OR}(artificialIllumination, ShuterBathUp)$;
$artificialIllumination = \mathbf{ALTERNATE}(celingLamp\_On, MirrorLampsOn)$;
$MirrorLampsOn = \mathbf{AND}(Lamp9\_On, Lamp8\_On)$;
$Kitchen\_CookingDay\_Scenario\_1 = \mathbf{AND}(ExhaustFan\_On, DoorActuator5\_Close, DoorActuator7\_Close,$
$Lamp6\_Off, Kitchen\_WS\_Day\_Scenario)$;
$Kitchen\_WS\_Day\_Scenario = \mathbf{ALTERNATE}(Kitchen\_WS\_CloseUp, Kitchen\_WS\_OpenDown)$;
$Kitchen\_WS\_CloseUp = \mathbf{AND}(WindowActuator\_Kitchen\_Close, Shutter\_Kitchen\_Up)$;
$Kitchen\_WS\_OpenDown = \mathbf{AND}(WindowActuator\_Kitchen\_Open, Shutter\_Kitchen\_Down)$;
$BedRoom\_Natural\_Illumination = \mathbf{AND}(BedRoom\_WS\_West\_CloseUp, BedRoom\_L1\_WS\_North\_CloseUp)$;
$BedRoom\_WS\_West\_CloseUp = \mathbf{AND}(WindowActuator\_w2\_Close, ShutterActuator\_sh2\_Up)$;
$BedRoom\_L1\_WS\_North\_CloseUp = \mathbf{AND}(WindowActuator\_w1\_Close, ShutterActuator\_sh1\_Up)$;

Morning WakeUp Activities performed by the resident of the house after waking up in the morning map to the "Morning WakeUp" use case. The activities can be illuminating the bedroom, the kitchen and the bathroom, switching off the gas heater inside the bedroom, switching on the television in the kitchen and the radio inside the bathroom. The functional representation is shown in Table 7.

5.1.2. Comments

Table 8 depicts the summary of the number of active and inactive effects when the states of devices were changed randomly, over the aforementioned use cases. The *no. of Devices changed* column shows the number of devices for which the states were randomly changed. The resulting number of active effects and inactive effects are shown in columns *No. of Active Effects* and *No. of InActive Effects*, respectively. The changes in the states of the devices (coming from *DogState* bundle) were detected and propagated in the ENN to perform the *Effect Evaluation* process. It can be observed from Table 8 that the total number of domotic effects remains equal to 190, along with providing monitoring the environment with the help of active and inactive domotic effects.

Table 8. Results of the feasibility testing

No. of Devices changed	No. of Active Effects	No. of Inactive Effects
2	3	187
4	7	183
5	6	184
2	3	187
7	14	176
2	4	186
1	2	188
3	3	187

5.2. Performance Evaluation

To measure the performance parameters of the implemented *Domotic Effects Evaluation* and the modified *HouseModel* bundles, two experiments were conducted. For the experiments a house environment with 57 devices was simulated, whose domotic structure was defined using the DogOnt ontology. The experiments ran on a standard personal computer with a quad-core Intel i5 processor and 4GB of RAM. A *TestDogEffect* bundle was created to carry out the experiments and measure the performance parameters.

5.2.1. Experiment 1: Daily Chores Scenario

This experiment simulated a scenario that encapsulates daily chores occurring in a house. 12 test iterations were created with a different number of total domotic effects defined over the house. The number of simple and complex domotic effects were generated randomly in a range from 100 to 1500. The type of the effect operator between complex domotic effects, the number of children and parent, and the inter-dependency of all the domotic effects (level) among themselves were generated randomly. To measure the performance parameters, the *TestDogEffect* bundle randomly chose devices and changed their respective states. For each iteration, the process of changing device states was repeated at least 150 times and then the evaluation time was averaged for each iteration.

As defined in Section 4.1 the classical Zero Delay Event Driven Logic Simulation performs the effect evaluation process. The semantics of the simulation is such that if a device changes its state, the values of all the domotic effects dependent on that particular device may need to be recomputed and the values of other domotic effects remain

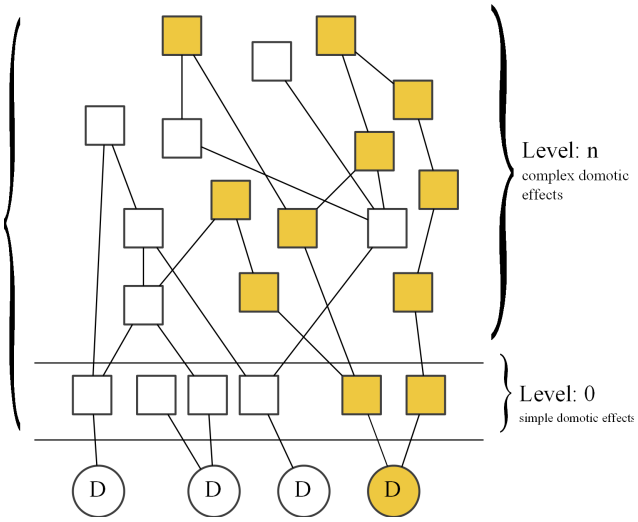

Figure 12. Semantics of Effect Evaluation process in Experiment 1.

unchanged. The semantics of the effect evaluation process in daily chores scenario is illustrated in Figure 12. If a device "D" (dark colored) changes its state, then the values of only domotic effects (SEs and CEs) dependent on the device "D" needs to be recomputed (depicted as dark colored).

The obtained performance parameters of this experiment are depicted in Table 9. The performance parameters include the total number of domotic effects (*Total Domotic Effects* column), the number of CEs (*Complex Effects* column), the time taken by the *HouseModel* bundle to extract domotic effects information from the ontologies (called Effect Extraction time), the time taken by the organization component to create the ENN (called Organization time), the maximum level of the ENN (*Maximum Level (ENN)* column) and the average time taken to perform the effect evaluation process in each iteration (called Average Evaluation Time). All the time measurements were taken in milliseconds. The effect extraction time includes loading the ontologies (DogOnt and DogEffects), checking consistency and performing realization. The organization time represents receiving all the domotic effects and organizing them in the ENN. Both effect extraction and organization time are measured by Dog, during the startup phase and they happen only once. The maximum level represents the height of the ENN. For example, the ENN depicted in Figure 6 has the height equal to two. The average evaluation time represents evaluating all the domotic effects and sending out notifications.

From the results in the Table 9, it can be observed that once the ENN is constructed, the *Effect Evaluation* process computes the new values of the domotic effects in real time (see *Average Evaluation Time* column). The tasks of effect extraction and organization in the ENN take longer time periods and therefore, these tasks are performed at the startup of Dog.

The linear relationship between the total number of domotic effects (x-axis) and the average time taken to perform the effect evaluation (y-axis) for all iterations is shown in Figure 13. On the other hand, Figure 14 shows the relationship between the maximum level of the ENN (x-axis) and the average time taken to perform the effect evaluation (y-axis) for all iterations.

Table 9. Daily chores scenario performance parameters

Total Domotic Effects	Complex Effects	Effect Extraction Time	Organization Time	Maximum Level (ENN)	Average Evaluation Time (ms)
209	100	22506	182	18	5
309	200	44929	289	21	7
409	300	22107	363	37	11
459	350	30400	482	57	15
599	490	24170	662	65	24
709	600	36596	819	122	43
809	700	28261	887	129	33
909	800	33815	1017	120	30
1069	960	45106	1143	105	27
1159	1050	49721	1280	232	108
1309	1200	70404	1450	122	52
1609	1500	67311	1994	177	118

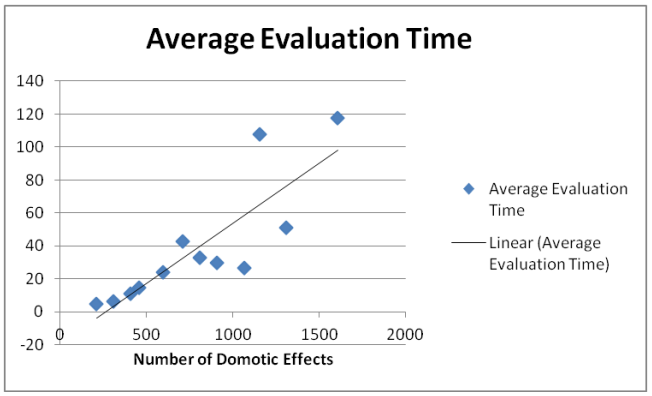

Figure 13. Daily chores scenario: Linear relationship between Total Number of Domotic Effects (x-axis) and Average Evaluation Time (y-axis)

5.2.2. Experiment 2: Maximal Propagation Scenario

This experiment was an attempt to simulate a scenario in which a change in state of a single device will initiate the recomputation of values for a significant number of nodes in the ENN. The complex domotic effects were generated randomly by taking SEs (or parent CEs) of a single device (identified by 'UD') in combination with other operands, pointing at other random devices. Therefore, when the 'UD' will change state the values of all the complex domotic effects might need to be evaluated. The semantics of the effect evaluation process in this designed experiment is illustrated in Figure 15. The crux is that when the device 'UD' changes its state, the values of all the domotic effects in the ENN may need to be recomputed.

10 iterations were performed with a random number of complex domotic effects generated, the type of the effect operator between complex domotic effects, the number of children and parent and the inter-dependency of all the domotic effects (level) were generated randomly.

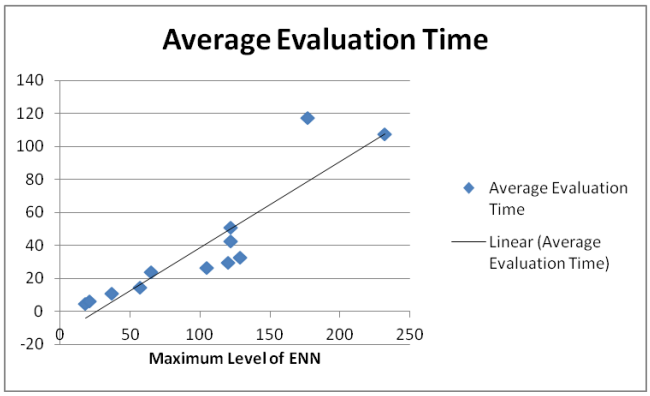

Figure 14. Daily chores scenario: Linear relationship between Maximum level of the ENN (x-axis) and Average Evaluation Time (y-axis)

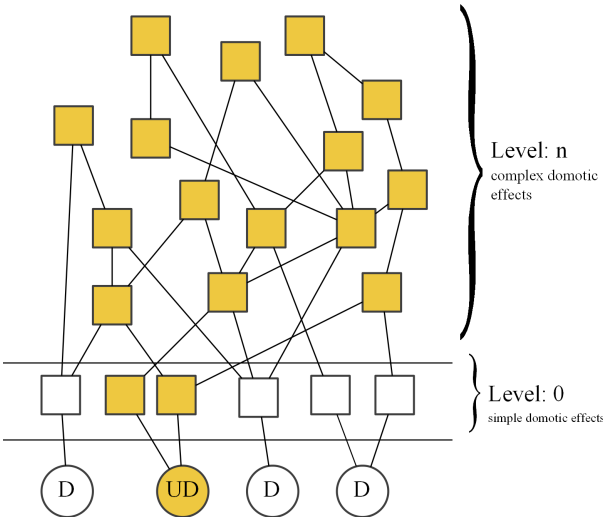

Figure 15. Semantics of Effect Evaluation process in Experiment 2.

For each iteration, the state of the unique device was changed. After that, the states of other randomly chosen devices in the house were changed and then again the state of the unique device was changed. This process was repeated at least 20 times for each iteration and in the end, the time taken to perform the evaluation process was averaged. The performance parameters of this experiment are depicted in Table 10. The performance parameters include the total number of CEs (*Complex Effects* column), the maximum level of the ENN (*Maximum Level (ENN)* column) and the average time to perform the effect evaluation process (*Average Evaluation Time*). The time is represented in milliseconds (ms).

The linear relationship between the total number of CEs (x-axis) and the average time taken to perform the effect evaluation process (y-axis) for all the iterations is shown in Figure 16.

Table 10. Maximal Propagation Scenario Statistics

Complex Effects	Maximum Level (ENN)	Average Evaluation Time
100	38	21
200	124	18
300	117	59
400	178	41
500	270	55
600	100	144
800	265	176
1000	314	148
1200	201	313
1400	272	258

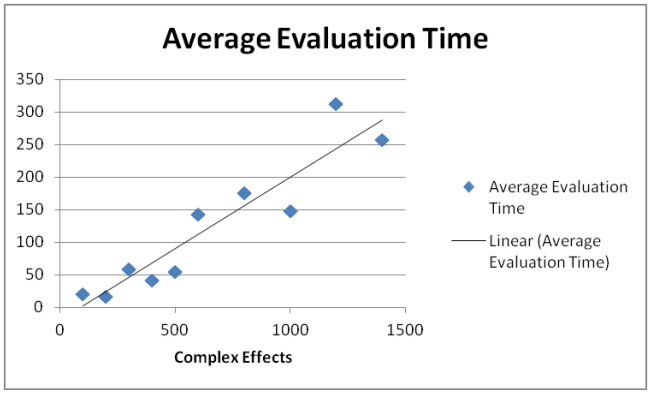

Figure 16. Maximal Propagation Scenario: Linear relationship between Total Number of CEs (x-axis) and Average Evaluation Time (y-axis)

5.3. Discussion

The developed set of applications presented in Section 5.1 proves that the "Domotic Effects" based approach can be integrated inside smart environment systems and is relatively flexible and easy to integrate with third party applications and services. Different monitoring applications (both web and desktop based) can be developed to monitor the state of the overall environment using the "Domotic Effects" approach.

The observations regarding the measured performance parameters during experiments (presented in Section 5.2) are pointed below.

1. For the daily chores scenario (Table 9), the effect evaluation process takes a maximum of around 118 milliseconds to complete the effect evaluation process and send out notifications, in case of 1609 domotic effects. It can be seen that the "Domotic Effects Evaluation" bundle is quite responsive and responds in near real-time. In most of the cases the time for evaluation and sending out notification was less than 150 ms. In fact, the number of domotic effects needed for average homes and small buildings will be in hundreds and only for large industries the number of domotic effects will exceed to thousands.

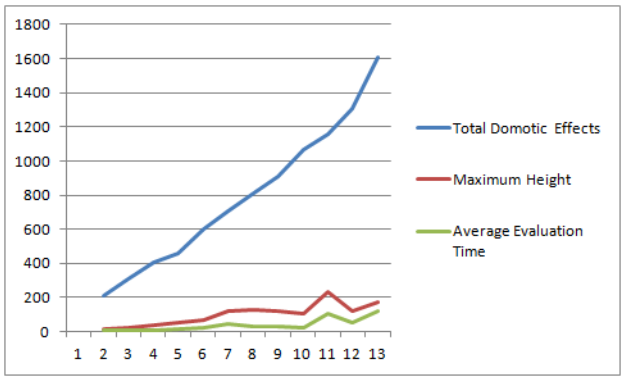

Figure 17. Comparison of Average Evaluation Time, Level of the ENN and the Total Number of DEs (for daily chores scenario).

2. As mentioned before, in the maximal propagation scenario all complex domotic effects are dependent directly or indirectly on a single unique device. Table 10 shows that the effect evaluation process takes an average of around 313 milliseconds to complete the effect evaluation process and send out notifications. Considering a unique device "UD" having 1000 domotic effects in a house means that a house with 57 devices can have thousands of domotic effects. Even in such rare occurrence (for large buildings), the average evaluation time is less than a second and takes on average 312.86 ms for 1200 CEs to be evaluated and therefore, the experiments indicates the responsiveness of the approach.
3. Both Figure 13 and Figure 16 indicate that the average time taken to perform the effect evaluation process is linearly increasing with respect to the number of domotic effects defined in an environment. However, this increase is not radical and it can be observed in Figure 16 that even if the number of domotic effects are in thousands, the effect evaluation time will be less than 1 second.
4. Figure 14 shows the linear relationship between the average effect evaluation time and the maximum level of the ENN (for the daily chores scenario). However, again it can be observed that this linear increase is not radical and even if the maximum level of the ENN is large, the effect evaluation process can be safely completed in near real-time.
5. Since the time taken to perform the effect evaluation process is less than 1 second (as proved from the results of experiments), it can be safely assumed that the proposed approach can easily be deployed in smart environments to monitor the overall state of the environment in near real-time.
6. Figure 17 shows the comparison between the total number of domotic effects (top), the average effect evaluation time (middle) and the maximum level of the ENN (bottom) for the daily chores scenario. A very interesting pattern can be observed. Though the average effect evaluation time is linearly increasing with respect to both total number of domotic effects and the maximum level of the ENN, the time curve is more correlated with respect to the maximum level of the ENN. It points to the event driven characteristic of the Zero Delay Simulation Algorithm.

6. Conclusion

This paper presented a high level approach, based on the concept of Domotic Effects, for monitoring and interpreting complex smart environments. The Domotic Effects framework, based on the DogEffects ontology, is general and extensible, and is easy to customize to specific application requirements. In particular, this paper focuses on monitoring applications, where high level effects may be described resorting to Boolean expressions operating on device states.

The paper presented extensive examples of Simple and Complex Effects over a sample home environment, and shows experimental results that prove that the complete state of the environment can be monitored using the Domotic Effects with a latency under 150 ms.

References

[1] S. Bader and M. Dyrba, Goalaviour-based control of heterogeneous and distributed smart environments, in: *7th International Conference on Intelligent Environments*, 2011, pp. 142–148.

[2] K. Birman and T. Joseph, Exploiting virtual synchrony in distributed systems, in: *Proceedings of the eleventh ACM Symposium on Operating systems principles*, ACM, 1987, pp. 123–138.

[3] D. Bonino, E. Castellina, and F. Corno, Dog: An ontology-powered osgi domotic gateway, in: *20th IEEE International Conference on Tools with Artificial Intelligence*, IEEE, 2008, volume 1, pp. 157–160.

[4] D. Bonino and F. Corno, Dogont - ontology modeling for intelligent domotic environments, in: Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2008, pp. 790–803.

[5] M.A. Breuer and A.D. Friedman, *Diagnosis and Reliable Design of Digital Systems,* . Computer Science Press, 1976.

[6] D. Chen, J. Yang, and H.D. Wactlar, Towards automatic analysis of social interaction patterns in a nursing home environment from video, in: *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, ACM, 2004, pp. 283–290.

[7] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(6):790 –808, nov. 2012.

[8] S.T. Cheng, C.H. Wang, and C.C. Chen, An adaptive scenario based reasoning system cross smart houses, in: *9th International Symposium on Communications and Information Technology* , IEEE, 2009, pp. 549–554.

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

[10] F. Corno and F. Razzak, Publishing LO(D)D: Linked Open (Dynamic) Data for Smart Sensing and Measuring Environments. *Procedia Computer Science*, 10:381–388, 2012.

[11] S. Davidoff, M. Lee, C. Yiu, J. Zimmerman, and A. Dey, Principles of smart home control, in: *UbiComp 2006: Ubiquitous Computing*, 2006, pp. 19–34.

[12] E. Aarts and B. de Ruyter, New research perspectives on Ambient Intelligence. Journal of Ambient Intelligence and Smart Environments, 1(1):5-14, 2009.

[13] A.K. Dey, G.D. Abowd, and D. Salber, A context-based infrastructure for smart environments, in: *1st International Workshop on Managing Interaction in Smart Environments*, Springer, 1999, pp. 114–128.

[14] H. Dibowski, J. Ploennigs, and K. Kabitzsch, Automated design of building automation systems. *IEEE Transactions on Industrial Electronics*, 57(11):3606–3613, 2010.

[15] M. Doorn, A. Vries, and E. Aarts, End-user software engineering of smart retail environments: The intelligent shop window, in: *Proceedings of the European Conference on Ambient Intelligence*, Springer-Verlag, 2008, pp. 157–174. .

[16] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and JC Burgelman, Ambient intelligence: From vision to reality. *IST Advisory Group Draft Rep., Eur. Comm*, 2003.

[17] J.L. Encarnaçao and T. Kirste, Towards smart appliance ensembles, in: Matthias Hemmje, Claudia Niederée, Thomas Risse, editors, *From Integrated Publication and Information Systems to Information and Knowledge Environments*, volume 3379 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2005, pp. 261–270.

[18] M. Garcia-Herranz, P. Haya, and X. Alaman, Towards a ubiquitous end-user programming system for smart spaces. *Journal of Universal Computer Science*, 16(12):1633–1649, 2010.

[19] A. Esposito, L. Tarricone, and M. Zappatore, A versatile context-aware pervasive monitoring system: Validation and characterization in the health-care domain, in: *2010 IEEE International Symposium on Industrial Electronics (ISIE)*, 2010, pp. 2791–2796.

[20] T. Heider and T. Kirste, Supporting goal-based interaction with dynamic intelligent environments, in: *Proceedings of the 15th European Conference on Artificial Intelligence*, 2002, pp. 596–602.

[21] F. Kawsar, T. Nakajima, and K. Fujinami, Deploy spontaneously: supporting end-users in building and enhancing a smart home, in: *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 2008, pp. 282–291.

[22] A. Katasonov, Enabling non-programmers to develop smart environment applications, in: *IEEE Symposium on Computers and Communications*, IEEE, 2010, pages 1059–1064.

[23] Hyo nam Lee, Sung-Hwa Lim, and Jai-Hoon Kim, UMONS: Ubiquitous monitoring system in smart space. *IEEE Transactions on Consumer Electronics*, 55(3):1056–1064, 2009.

[24] P. Rashidi and D.J. Cook, Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(5):949–959, 2009.

[25] F. Razzak, *The Role of Semantic Web Technologies in Smart Environments*, PhD dissertation, Politecnico di Torino, 2013. http://porto.polito.it/id/eprint/2506366

[26] J. Chin, V. Callaghan and G. Clarke, Soft-appliances: A vision for user created networked appliances in digital homes. Journal of Ambient Intelligence and Smart Environments, 1(1): 69-75, 2009.

[27] E. Salomons, W. Teeuw, H. van Leeuwen, and P. Havinga, Persona-based adaptation in a smart green home, in: *8th International Conference on Intelligent Environments*, IEEE, 2012, pp. 355–358.

[28] R. Velik and H. Boley, Neurosymbolic alerting rules. *IEEE Transactions on Industrial Electronics*, 57(11):3661–3668, 2010.

[29] R. Velik and G. Zucker, Autonomous perception and decision making in building automation. *IEEE Transactions on Industrial Electronics*, 57(11):3645–3652, 2010.

[30] R. Velik, G. Zucker, and D. Dietrich, Towards automation 2.0: a neurocognitive model for environment recognition, decision-making, and action execution. *EURASIP Journal on Embedded Systems*, 2011:4, 2011.

[31] W3C, OWL : Ontology Web Language. Technical report, W3C, 2004.

[32] J. Ye, G. Stevenson, and S. Dobson, A top-level ontology for smart environments. *Pervasive and Mobile Computing*, 7(3):359–378, 2011.