BENTHAM OPEN

CrossMark

The Open Cybernetics & Systemics Journal

Content list available at: www.benthamopen.com/TOCSJ/

DOI: 10.2174/1874110X01610010263

RESEARCH ARTICLE

µGRIMOIRE: A Tool for Smart Micro Grids Modelling and Energy Profiling

Ugo Gentile[1], Stefano Marrone[2,*], Nicola Mazzocca[1] and Roberto Nardone[1]

[1]*Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Università "Federico II" di Napoli, Naples, Italy*
[2]*Dipartimento di Matematica e Fisica, Seconda Università di Napoli, Caserta, Italy*

Received: June 04, 2015 Revised: September 06, 2016 Accepted: September 19, 2016

Abstract: The construction of a usable, formal, and extensible modeller and simulator for Smart Energy Grids is of a paramount importance in the industrial settings. Final users are interested in deploying effective smart-home configurations able to satisfy energy requests in the most economical way. Hence, a tool able to forecast both energy consumption and related costs of a smart-home configuration is needed. In this paper, the µGRIMOIRE (micro GRId MOdelling envIRonmEnt) toolset is presented. This tool is based on the well-known model-driven paradigm and its successful applications in the generation of formal/quantitative models for complex systems. By using a Domain Specific Modelling Language, a final user can define a smart-home system configuration and energy saving logics. Then, the tool offers the possibility of evaluating the desired user metrics by translating the model into a Fluid Stochastic Petri Net model representing both discrete and continuous variables.

Keywords: Fluid Stochastic Petri Nets, Model-Driven Techniques, Hybrid Systems Design, Energy Consumption Evaluation, Smart Grid Modelling Language.

1. INTRODUCTION

The current power grids can be schemed as centralized systems with uni-directional flows of energy departing from the generation plants to the passive consumers through the transmission and distribution lines [1]. Such schema is not able to cope with the current challenges in the power supply domain for higher demands, higher security and reliability guarantees, lower environmental impacts and reduced operating costs. Blackouts often occur in the existing grids during peak demand periods: as an example, the blackout in Northeast India caused by a power request due to extreme heat during the monsoon season [2]. Centralized energy production systems are also single points of failure due to random faults (*e.g.*, the blackout in California caused by a procedural error made by a technician [3]) or malicious attacks (*e.g.*, the national electricity blackout in Iran caused by the Stuxnet virus attack [4]). Furthermore, bring the energy flow to consumers located far from the power plants implies high costs for managing and maintaining the transmission and distribution lines. The centralization of energy distribution facilitates energy loss of stealth during transmission and distribution (*e.g.*, in India, 27% of the generated power is lost [5]).

While several approaches are followed in protecting and diagnosing such systems [6, 7], the primary objective is to build a new generation of distribution systems. Several governments and national public power grid agencies are defining plans for improving the current situation and identifying which assets might be needed to meet future electric power demands. The solution for taking advantages of the opportunities of this novel market is to distribute electricity giving a more active role to the customers. In fact, customers shall be not only passive consumers but they could generate their own energy quota using renewable sources. Moreover, the energy generated from renewable sources can have periodic power fluctuations, which causes surpluses or lacks of energy asking for proper energy storages to stock

* Address correspondence to this author at the Dipartimento di Matematica e Fisica, Seconda Università di Napoli, Caserta, Italy; Tel: +39 0823275101; Fax: +39 0823274753; E-mail: stefano.marrone@unina2.it.

temporarily exceeding energy. Therefore, there is the need for bidirectional energy flows between consumers and traditional distribution system.

The management of these new features needs an update in the operation and control of power grids and the adoption of an advanced ICT infrastructure with self-organizing capabilities. The ICT infrastructure allows adopting an on-line monitoring of the grid to check its correct functioning and applying countermeasures to tolerate malfunctions. Additionally, it can support an efficient management of the electricity distribution and peer exchange realizing billing mechanisms based on the electricity actually produced and consumed by different customers. "Smart power grid" [8] defines such a complex, interconnected and interactive model for power grids where ICT is used to gather information on suppliers and consumers and to work in an autonomic way.

To address communication matters of such grid of the future, we choose a scaled-down model of a Smart Grid, called Smart Micro Grid and adopt a modelling approach able to simulate realistically the behavior of a given Smart Micro Grid. We focus the attention on the Smart Home (SH), a first-class citizen of the Smart Micro Grids ecosystem. A Smart Home is a house or building where devices interact over a communication network, in order to control some energy functionalities as heating or cooling the environment. A Smart Home enables consumers to produce energy from local generators (*e.g*., solar panels), to manage local storage devices and to re-sell exceeding energy.

Within the context of Smart Home, Smart Meters (SMs) play a key role. In fact, an SM is a two-way communication device, which measures the energy consumed by the customers during specific time intervals and communicates the achieved data to the Operator Control Center (OCC). Furthermore, by means of an SM, the users can remotely/autonomously control the electrical appliances within the Smart Home by power-off commands to decrease energy consumptions in the case of peak energy demand and avoiding a home blackout.

Since the high variability that these new technologies bring into this world, traditional modelling and analyzing methods need to be improved. In particular, designers want to analyze a specific configuration with predictive methods and tools. The objective of this analysis is to evaluate the compliance of the system with energy requirements lowering the total operational costs as much as possible. In general, these new methods should comply with requirements that can be summarized as follows: (1) the modelling should be based on concepts close to the expertise of the designer, avoiding complex mathematical formalization; (2) the analysis, on the contrary, should accurately evaluate different system and software configurations by providing numerical evidence on some sensitive features (*e.g*., saved money amount, total consumed energy). The system designer can decide, by analyzing simulation results, which control algorithm fits best.

Model-driven techniques are used in this paper [9] to cope with the above-mentioned requirements. By means of the creation of a Domain Specific Modelling Language (DSML), the designer of a Smart Micro Grid can use concrete and technical-related modelling primitives to create a graphical model of a Smart Micro Grid in a very simple way. Such a model can be transformed by a model-to-model transformation into a model conformant to another formalism that can be useful to perform the different analyses. In this work, we consider the language of the Fluid Stochastic Petri Nets (FSPNs) as target formalism [10]. FSPNs allow easy modelling of hybrid systems representing both discrete and continuous variables. Once an FSPN model of the Smart Micro Grid is generated, proper tools can be used to extract information of interest. This paper contains some details about the languages and the transformation rules, and it provides information about the supporting tool that the designer can use. The presented tools take part of μGRIMOIRE (micro GRId MOdellIng envIRonmEnt), an IDE for the development of Smart Micro Grids. This paper represents a consistent extension of a research work published in [11], where the modelling fundamentals behind our approach have been shown.

The structure of the paper is the following. Section 2 provides the needed background to ease the reading of the paper and a discussion about related works. Section 3 explains the process of modelling and analysing of a micro grid configuration and shows the workflow of the activities a designer has to follow. Section 4 focuses on the modelling languages used in the paper while Section 5 presents how these languages are translated into an FSPN models. Section 6 shows the effectiveness of the approach with a case study and Section 7 draws some conclusions.

2. BACKGROUND AND RELATED WORKS

2.1. Fluid Stochastic Petri Nets

A brief description of the Fluid Stochastic Petri Nets (FSPNs) is provided to improve the readability of this paper.

The FSPN formalism extends SPN by considering two types of places: discrete places, marked by a non-negative integer number of tokens, and continuous places, marked by a continuous positive real value. FSPNs provide timed and immediate transitions as well as ordinary and inhibitor arcs. Fluid arcs are also introduced to cope with the marking update of continuous place.

From a syntactic point of view, fluid places can be connected to timed transition exclusively by means of fluid arcs, while ordinary places can be connected to timed transitions by ordinary arcs and to immediate transitions by ordinary and inhibitor (in this case, only as arc sources) arcs. In particular, FSPNs fit to model hybrid systems, *i.e.*, systems having both discrete and continuous parts that evolve over time.

The fluid nature of FSPNs makes them applicable to large models, which would be problematic or infeasible to analyze with common discrete SPNs due to the state space explosion problem. In fact, with discrete-only nets, markings with a large number of tokens are needed to represent realistically the evolution of a hybrid system.

In FSPNs, only discrete places determine to enable transitions and firing sequences. Firing rules of FSPNs are very similar to the usual rules applied to SPN, as fluid places do not take part in transition enabling: thus, enabling and firing happen in the same way as standard SPN. Besides traditional effects provided by Petri nets semantics, fluid flow is permitted through the enabled timed transitions. Such flows use fluid arcs between timed transitions and fluid places. A modeller can write the differential equations for the underlying stochastic process by associating exponentially distributed or zero firing time with transitions. While a transition is enabled, each fluid arc ending in the transition drains fluid marking from its fluid place of origin at a specified rate; each fluid arc originating from such a transition increases fluid marking of its destination fluid place at a specified rate. Hence, using this formalism, both logic/discrete and physical/continuous concepts can be modelled together.

In the following, a little example showing the main features of this formalism is provided. Fig. (**1**) models a plugged lamp with a switch: when the switch is closed, the lamp works and drains energy; otherwise, no energy is consumed. The fluid place *Lamp* contains all the energy consumed by the lamp while it is active. When a token is in the *On* ordinary place, the *Consuming* transition is enabled and *Clamp* energy flows into the *Lamp* place per unit of time using the fluid arc from *Consuming* to *Lamp*. *SwitchOn* and *SwitchOff* transitions respectively model the switching on and off the lamp.

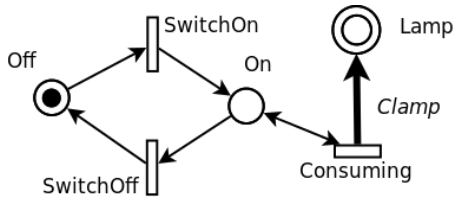

Fig. (1). Example of an FSPN model.

2.2. Model Driven Engineering

Model Driven Engineering (MDE) methodology was initially introduced as a software development methodology whose objective was the generation of code starting from domain models (*i.e.*, abstract representations of the knowledge and activities present in a particular application domain). In fact, MDE simplifies the design processes by raising the level of abstraction and by reusing standardized models, best practices and recurring design patterns in the application domain. MDE also promotes the communication between working groups standardizing the jargon into a DSML used to create high-level models of the systems of interest. MDE can be considered effective when the users are familiar with the primitives of the DSML, that should be as simple as possible, as rich as needed [12]. The high-level models are then transformed into artifacts through model transformations.

Model-driven techniques facilitate model-based assessment of system requirements: MDE is here used to generate formal models from high-level specifications. Many attempts are present in literature showing the adoption of MDE to system development and verification in different application domains (we recall here just a few examples for non-functional properties): security [13], performance [14], dependability [15], reliability and availability [15], QoS and its relationship with the other properties [16]. These works demonstrate that the model-driven paradigm is very appealing in industrial settings, for complex and critical systems.

Two better-known MDE initiatives are: (1) the Object Management Group (OMG) model-driven architecture (MDA), which is a registered trademark of OMG [17], and (2) the Eclipse ecosystem of programming and modelling tools (Eclipse Modeling Framework) [18]. Both of them have similar foundations, but different modelling languages support them: the former is based on the UML profiling technique [19], which is a lightweight metamodelling technique to extend UML refining its semantics in a strictly additive manner, while the latter requires the complete definition of a new metamodel (*e.g*., by the Ecore metamodelling framework).

Model transformations generate new artifacts from existing ones: they can be divided into model-to-model (M2M), text-to-model (T2M) and model-to-text (M2T).

2.3. Related Works

In the last years, many works have been proposed with the objective of profiling energy in smart micro grids. In [20] the authors analyze different solutions centered on the role the ICT infrastructures must play in the definition of energy-aware policies.

Other approaches focus on the definition of efficient scheduling algorithms for Smart Meters. For example, the work described in [21] proposes a scheduling algorithm for SMs able to balance energy consumption within a neighborhood on a shared electrical channel; the work also proposes a billing model, separated from the scheduler, encouraging the adoption of this shared mechanism. The work in [22] analyses a smart charging system that uses a local energy storage to provide savings in customers' electric bill by stocking energy during low-cost periods.

Many simulation tools, which analyze the energy consumption, have been developed. In [23] the authors describe a framework for the interaction of common home devices and appliances to control and monitor energy consumptions. The tool shows the estimated consumption to the customer with the related costs in order to allow him/her adopting a better strategy for energy and costs reductions. Choosing an optimal energy consumption policy can also be pursued by transferring this responsibility to a device within a Smart Home. GridLab-D [24] and EnergyPlus [25] are two existing tools, which examine the energy consumption with the focus on heat generation and thermal load of a building, considered as the main energy-consuming activity. PowerMatcher Simulation Tool [26] also analyze the billing, giving the possibility of varying the price of energy throughout the day, but only the energy demand is considered as static, without allowing to model energy consumption profiles. The Smart Home Simulator [27] introduces the capability to model a smart home configuration, to set the energy workloads, starting from real-world data and to allow how a Smart Meter logic behaves in this configuration.

Demonstration of the effectiveness of PNs and related formalisms in the modelling of smart grids are in [28, 29].With respect to the above-revised literature, the tool presented in this paper relies on model-driven techniques which bring important advantages. Specifically, an high-level model of the system under analysis is automatically transformed into a formal model, developed according to the well-known formalism of the FSPNs. This approach has two main advantages: (1) the high-level model can be highly customized since it is developed in an ad-hoc developed modelling language; the graphical interface of the tool can be easily adapted with respect to the end user's needs and preferences; (2) the adopted FSPN model structure is compositional allowing future extensions of the framework. The proposed approach benefits from the results obtained in our previous works [11] where we defined an FSPN model library of core components, which can be joined, according to compositional modelling approaches, with the objective of quantitatively assess the energy consumption in smart-home. Possible algorithms, implemented by smart meters, have been taken into account by modelling them in the FSPN formalism. This paper instead allows for the automatic generation of these models and the necessary compositional operators, starting from the high-level model of the system.

3. THE µGRIMOIRE WORKFLOW

The proposed modelling and analysis environment is named µGRIMOIRE, and this Section is an overall description of how it works. Fig. (**2**) shows the workflow of the activities from both the User and Tool perspectives: in the figure, lines tagged with the <<*of*>> tag represents data flow relationships (creation and use of artifacts) while the untagged lines are control flow relationships (determining the order of the activities).

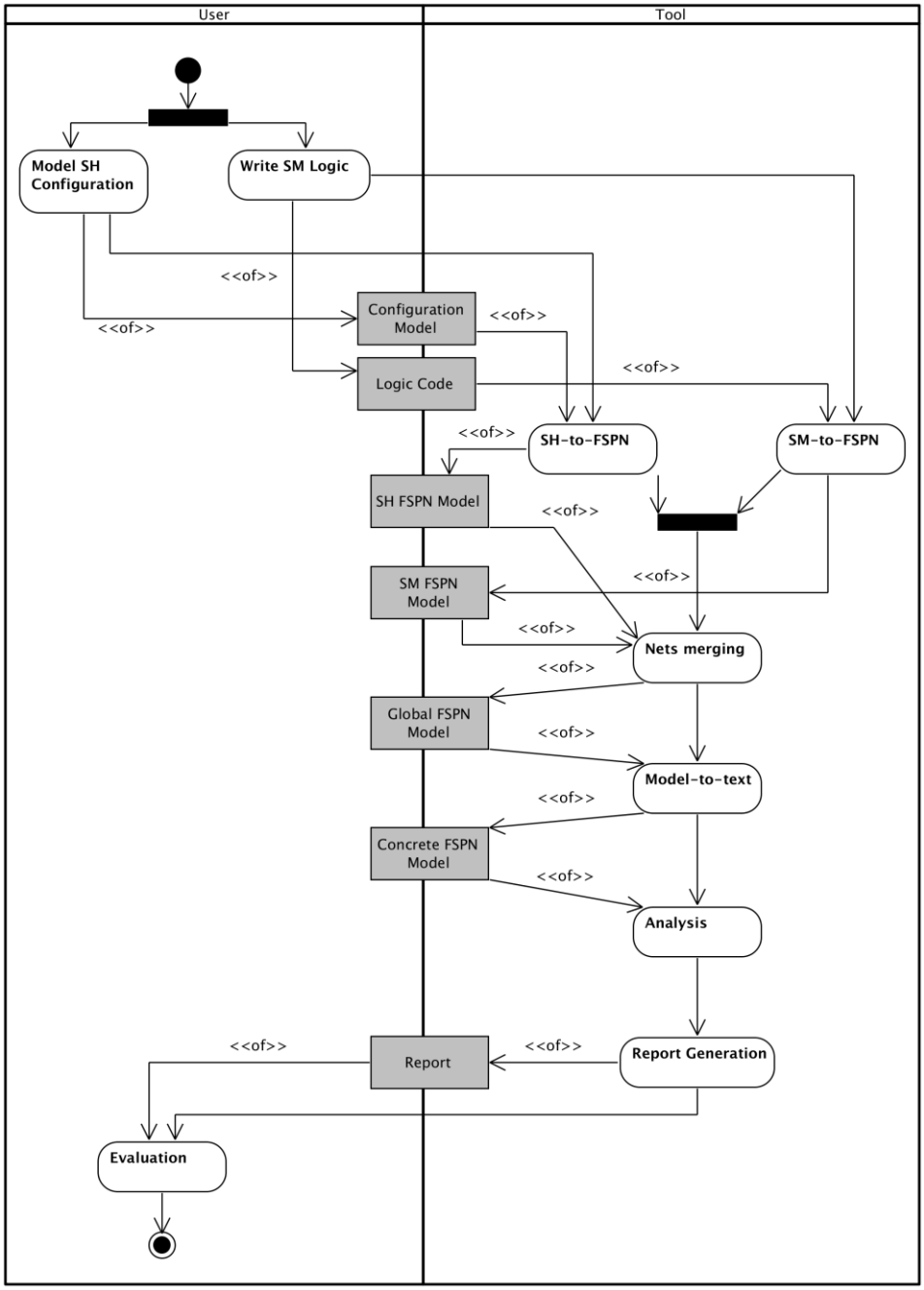

Fig. (2). Usage Workflow of the proposed approach.

The User starts with modelling both the Smart Home configuration and the energy/money saving logic that rules the functioning of the Smart Meter. Based on a graphical interface, the User generates a **Configuration Model** inside the *Model SH Configuration* activity. Then, he/she writes the **Logic Code**, a textual procedure describing the sequence of instructions executed by the Smart Meter (*Write Logic Code*): These two activities can be done in parallel since they are independent: such independence is guaranteed when well-defined modelling and programming interfaces are previously defined, as described in Section 4.

After these phases, two parallel activities start in the context of the Tool. A set of back-end tools deals with:

- taking the **Configuration Model** and translating it into an **SH FSPN Model** (*SH-to-FSPN* M2M);
- taking the **Logic Code** and translating it into an **SM FSPN Model** (*SM-to-FSPN* T2M);
- merging the **SH FSPN Model** and the **SM FSPN Model** into a single **Global FSPN Model** (*Nets Merging*

M2M);
- translating the **Global FSPN Model** into a **Concrete FSPN Model** conformant to a concrete notation used by an existing FSPN solver;
- analyzing the **Concrete FSPN Model**, producing results and generating a **Report**.

The **Report** can be then analyzed by the User to evaluate the quality of the chosen design variants.

To accomplish this workflow, the framework needs the creation of the following artifacts:

- a Domain Specific Modelling Language on which the Configuration Model is built: this modelling language is named Smart Grid Modelling Language (SGML);
- a Domain Specific Programming Language Logic Code is conformant to: this language is named Smart Meter Programming Language (SMPL);
- a Computer-Aided Software Engineering (CASE) tool able to support the creation of both Configuration Model and Logic Code;
- a metamodel of the FSPN language;
- a tool solving FSPN and accepting a concrete FSPN notation;
- the following model transformations:
    - SH-to-FSPN translating an SGML model into an FSPN model;
    - SM-to-FSPN translating an SMPL model into an FSPN model;
    - FSPN-merging combining two FSPN models into another FSPN model;
    - FSPN-writing generating concrete FSPN file starting from an FSPN model;
- a tool able to analyze solution results and generates user-readable reports;

Some of these artefacts are detailed in the following sections.

4. THE SGML AND SMPL NOTATIONS

This Section defines the graphical and the textual notations adopted in this paper. First, an overview of the domain model is provided to define the common structure of these languages, then each notation is presented in a separate subsection.

4.1. The Domain Model Overview

Three main packages constitute the language: a package containing the structured and enumerative datatypes present in this language (Datatype Library), a package of the concepts related to the hardware configuration of the Smart Micro Grid (Architecture) and one with the software concepts related to the control of such configurations (Logic). Fig. (**3**) depicts the packages as well as the dependencies between them.

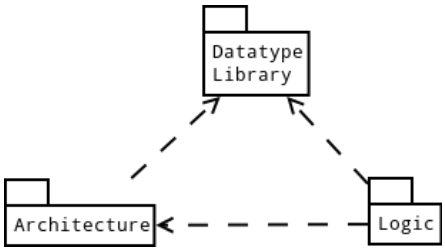

Fig. (3). Package-level overview of the domain model.

Fig. (**4**) details the **Datatype Library** package. It contains five enumerations that span from the list of the currencies (**Currency**) to the units of measurement of energy (**Energy Unit**).

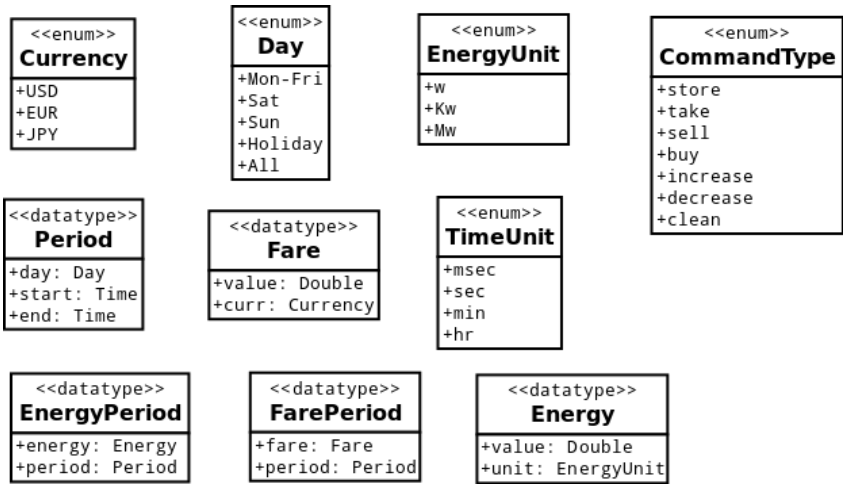

Fig. (4). Detail of the Datatype Library package.

Fig. (**5**) shows the conceptual model of the Architecture package. The root concept of this package is the **MicroGrid** metaclass, which represents a set of **Party** elements. A Party is an autonomous agent of the micro grid able to produce/consume energy and to regulate this production/consumption according to economic advantages; some examples are Smart Homes, Smart Factories, Distribution System Operators (DSOs) and Energy Providers. Each Party can be characterized by two arrays: the *buyFare* and the *sellFare* arrays. The *buyFare* (resp. the *sellFare*) array indicates the fare applicable by (resp. to) the Party during some predetermined periods of the day, when the Party wants to sell (resp. buy) some quota of energy to (resp. from) another Party. A Party can be composed of other parties.

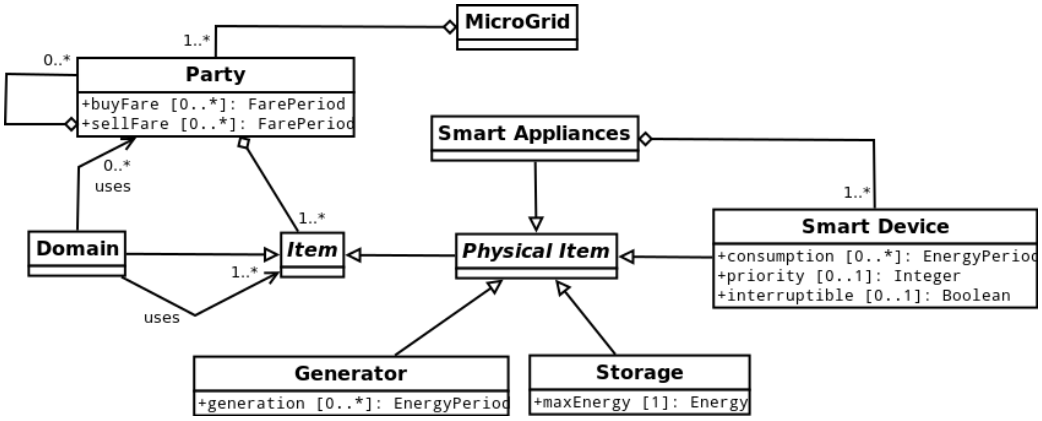

Fig. (5). Detail of the Architecture package.

A Party owns at its internal different **Item**s involved in the production/consumption of energy. Items are divided into *Physical Items* whose some examples are **Smart Device**s that consume energy (industrial machines as well as dishwashers), **Smart Appliances** (a collection of Smart Devices), **Storages** (able to store surplus of energy to be reused in future) and **Generators** (able to produce electrical energy from solar, wind or traditional energy sources). In particular, Smart Devices are characterized by a *consumption* (that can also be defined in different day periods), a *priority* (used in complex saving mechanisms where the Party can switch off some devices to save energy) and the *interruptible* flag (used to tag if the device can be interrupted or not). **Storage** facilities are characterized by a maximum level of energy storable (*maxEnergy*). **Generator**s are characterized by some rates of *generation* (related to day periods; *e.g.*, a solar panel that produces energy only during daylight hours).

Another important concept related to Items is the **Domain**. A Domain can be seen as a logical partition of a Party: domains can be used to separate the management of some items and/or some other parties. As an example, let us consider a Smart Factory with both low voltage and medium voltage needs: using domains, the modeller can separate the low voltages appliances from the medium voltage ones. These two different appliances can be then served by two distinct DSOs; there can be hence two domains: one using low voltage appliances and a DSO and another with medium

voltage appliances and another DSO.

Fig. (**6**) shows the details of the Logic package. This package contains the "programming" concepts using which a modeller can define the saving algorithms implemented in a Party. Each Party can be equipped with more than one **Applications**. An Application is the definition of a saving procedure that works (*controls*) on a Domain. The nature of the Application is twofold. Loop is a typical control procedure based on polling mechanism (it reads the inputs, computes the new state and make decisions, and it sends commands to the outside): **Loop**s are typically associated with a sleeping period (*period* and *unit*) after which the algorithm is executed. **Triggered** applications are instead executed when a predicate (*condition*) is evaluated true. Applications are constituted by two ordered lists of **Instruction**s: the **Init** and the **Body** lists. The first is used to indicate the instructions to execute when the application starts, the second the instructions used during the normal running of the application. An Instruction can be: (1) a **Wait**, a simple sleep of the algorithm; (2) a **Decision**, a classical decision statement of the programming languages, which contains two lists of ordered instructions (*then* and *else* branches); (3) a **Command**, used to modify the state of the Party. Commands are of the **CommandType** kind; typical commands are:

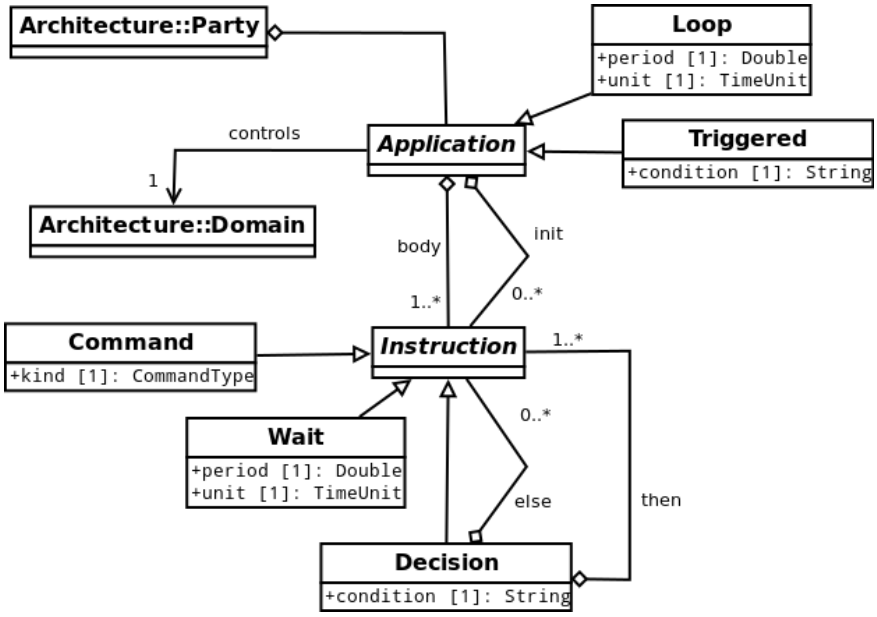

Fig. (6). Detail of the Logic package.

- *Store*: used to stock energy surplus to the storage facilities to future use;
- *Take*: used to drain energy from the batteries;
- *Decrease*: used to reduce the energy consumption of the Party by switching-off some interruptible devices;
- *Increase*: used to restore the energy consumption of the devices after a reduction;
- *Buy*: used to buy some energy from another party;
- *Sell:* used to sell some exceeded energy to another party;
- *Clean:* used to reset all the commands previously committed.

The presented domain model is general enough to deal with a large number of systems and different situations. Notwithstanding, in the following we make some simplification hypotheses when concrete representations of SGML and SMPL are presented. Such hypotheses restrict the scope to a single smart home.

4.2. The SGML Graphical Notation

Fig. (**7**) represents the metamodel the graphical notation is based on. Such metamodel is represented with an Ecore diagram, which is a formalism provided by the Eclipse Modeling Framework modelling environment.

Enumerations and data type of the data type library have been mapped with the respective Ecore EEnumeration and EDataType. Some enumerations have been added such as GenaratorKind and DeviceKind to consider different items that can be used within a smart energy domain.

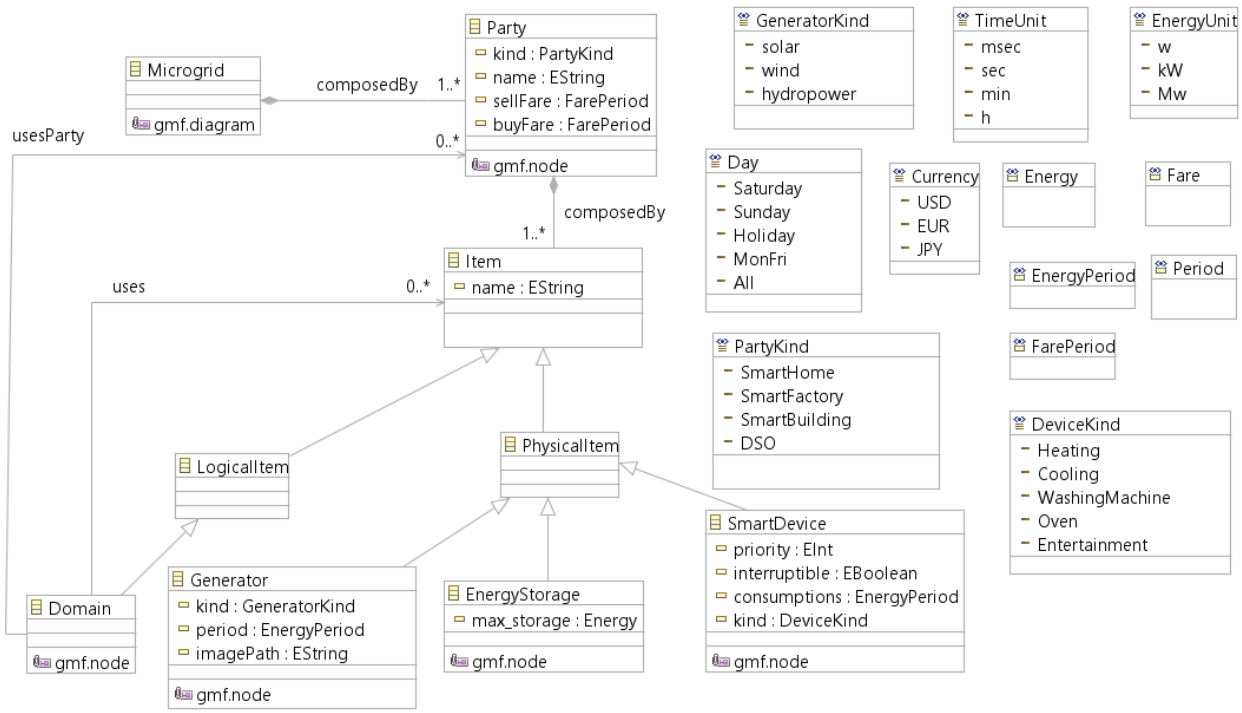

Fig. (7). Metamodel of SGML.

Regarding the architectural part, each element of the considered smart energy domain inherits from the Item abstract EClass that is characterized by a name. Items can be logical (*e.g.*, an energy domain) or physical (*e.g.*, elements that can produce and/or consume energy)

Due to the different role played by various physical items, there are three sub-categories of them:

1. Generator that represents a local generator from renewable sources, such as wind turbines or solar panels;
2. Energy storage, characterized by the maximum amount of energy that they can store;
3. Smart Device, which consumes energy in the Party.

4.3. The SMPL Textual Notation

The SMPL Textual Notation has the aim to support the modeller in the definition of the control logic of a Smart Meter. As the logic is often provided by a pseudo-algorithm, it is reasonable to implement a "programming" language for Smart Meters using a textual notation. To accomplish this task, the Logic package depicted in Fig. (**6**) has been implemented by creating a grammar. Listing **1** contains a little example of final logic: the programming concept of function is associated with an Application (of a Party): the Application specifies the name of the Domain after the APPLICATION keyword. Hence, the Init and the Body section follow, both of them containing a list of Instructions (decisional, wait and commands): init blocks are opened by the INIT keyword, body blocks are opened by LOOP or TRIGGER keywords. Decisional statements are arranged with the typical C-like syntax: conditions of the IF/IF-ELSE blocks are expressed by classical C-style predicates. Some keywords are used to denote special variables of the Smart Meters:

- ENERGY: the current energetic balance of the smart meter;
- BILL: the total economic expenses/incomes of the Smart Home;
- CONSUMED: the part of the energetic balance that is consumed by the electrical appliances;
- GENERATED: the part that is instead produced by local energy generators;
- STORED: the energy currently stocked into electrical batteries.

```
APPLICATION domain {
    INIT:
        clean();
    LOOP:
        if (ENERGY < 10.0) {
            store();
        } else {
            if (BILL > 50.0) {
                sell();
            }
        }
        wait(3,min);
        clean();
}
```

Listing 1: Example of GPML application.

Wait commands are expressed as a function WAIT(*period*,*unit*) where *period* is the numerical amount of time and *unit* is the enumerative value representing the TimeUnit in which period is expressed. Commands can be expressed by the following seven keywords: CLEAN(), STORE(), TAKE(), SELL(), BUY(), INCREASE() and DECREASE(). The grammar has been developed using SableCC [30]: an open-source parser generator. An excerpt of the EBNF-like notation of both lexical tokens and grammar production rules is provided in Listing **2**.

```
Tokens
energytok = 'ENERGY';
billtok = 'BILL';
consumedtok = 'CONSUMED';
producedtok = 'GENERATED';
storedtok = 'STORED';

Productions
condition = {plain} simple | {unary} unaryop condition | {binary} simple binaryop condition | {nested} rob condition rcb;
binaryop = {and} and | {or} or;
unaryop = not;
simple = lval cmp rval;
lval = var;
rval = {re} real | {nu} numstr | {va} var;
var = {generic} identifier | {energy} energytok | {bill} billtok | {consum} consumedtok | {produc} producedtok | {stor} storedtok;
```

Listing 2: Sample of the GPML grammar.

5. THE BACK-END TRANSFORMATION AND ANALYSIS LAYER

It is clear the advantage of having two separate models for micro grids architecture and party application: working with separate models can improve the modular design, reuse, and verifiability of the models and, of course, can exploit the most proper concrete syntax (graphical of textual). Less clear could be the necessity of having a compositional approach to building the FSPN model. A compositional generation has two main effects: it simplifies the design of the model transformations and it enables their extensibility and reusability allowing for the addition of further rules.

Furthermore, some hypotheses are made in this translation. First, *consumed* and *produced* energy values are not used. Instead, only the *stored* and *energy balance* are used. The second hypothesis is that there is only one battery for Party: this does not constitute a real limitation for many situations since a modeller can consider a virtual battery whose storage capacity is the sum of the capacities of all the actual batteries present in the Party.

5.1. Structure of the FSPN Model

The overall FSPN model is built according to the schema depicted in Fig. (**8**).

There are several modules each of one is generated by an element of an SGML model or by an SPML application. The two transformations (***sgml2fspn*** and ***spml2fspn***) work in parallel; when they are both executed, all the modules are merged by a composition mechanism to generate the overall model. This mechanism uses the place and transition superposition technique; some places and/or transitions are tagged with a label: composing two modules means to merge the places and the transitions having the same label. A full description of this approach and an implementation to

GSPN is in [31].

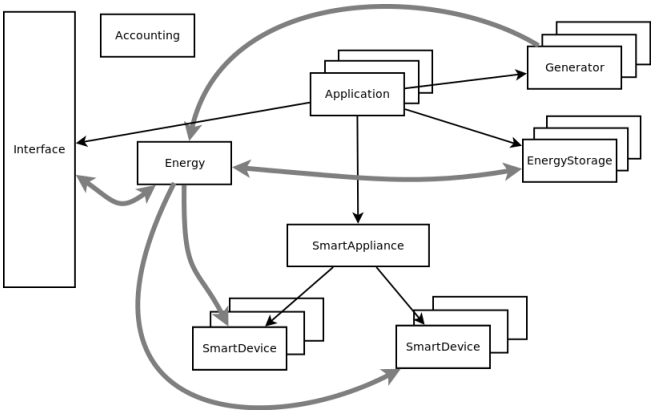

Fig. (8). FSPN Model Overall Structure.

5.2. From SGML to FSPN

Here, an excerpt of the mapping between the SGML and the FSPN formalisms is reported by illustrating the FSPN modules translating some of the SGML elements using the model transformation (***sgml2fspn***). First, Fig. (**9**) represents the FSPN pattern translating a Generator model element: in the model there are N couples of ordinary place and stochastic transition and an arc from the place to the transition. Each couple represents a period of the day and the inverse of the average duration of the i-th period is *Rgi*. When the i-th period is active, a token is in the i-th place and the i-th transition is enabled allowing the flow of an *Egi* amount of energy to the *Balance* fluid place of the Energy module.

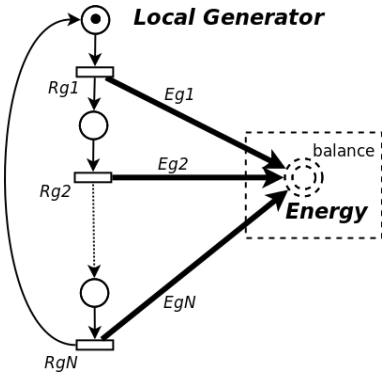

Fig. (9). FSPN pattern of the Generator.

Fig. (**10**) represents the FSPN pattern generated by a Battery. There are a fluid place and two stochastic transitions, which interact with an **Application** and the **Energy** modules. When the Application decides that energy should be stored in the battery, it puts a token in the *storeEnabler* place, which enables the *store* transition. This transition allows an energy flow from *balance* to *battery* fluid places. The dual mechanism acts in case of energy drain from the battery (*takeEnabler* place and *take* transition).

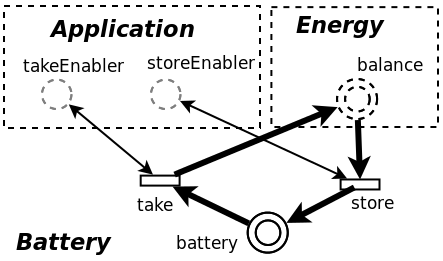

Fig. (10). FSPN pattern of the Battery.

Fig. (**11**) represents the pattern used for a Smart Device. Two parts constitute the model. The right part is similar to the one implementing the Generator: it models different times of the day and, for each time an energy consumption, a rate is associated with a fluid arc that drains energy from the global energetic balance of the party. The left part of the model deals with the interruption mechanism that a party can adopt to cut the consumption in high demand moments. The mechanism of choosing the Smart Device to interrupt is a responsibility of the Smart Appliance model that is described in [32]. However, when the device is requested to be interrupted, the *off* transition fires putting a token in the *off_cmd* place. If the device is interruptible, (*i.e.* it has a non-empty *Interruptible* place), the *start* transition is enabled, and a token is added to the *going_off* place. A similar mechanism acts when the Smart Appliance requests a restart of the device (*e.g.*, the *on* transition fires). When a token is present in the *going_off* place, the rates of the fluid arcs are set to zero (*i.e.*, the device consumes no energy).

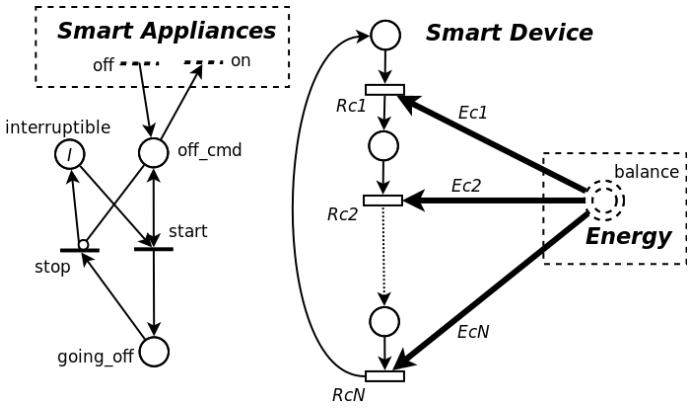

Fig. (11). FSPN pattern of the Smart Device.

5.3. From SMPL to FSPN

This translation is based on a compiler (***smpl2fspn***), partially generated by SableCC, an open-source compiler generator. The compiler implements a translation strategy that starts with the exploration and the visit of the parse tree during the analysis of an SMPL program. Once an application is translated by means of this compiler, the submodel is then composed of other FSPN submodels generated by the ***sgml2fspn*** model transformation. Here some FSPN patterns translating the main constructs of the language are reported:

- INIT: the INIT block is translated into the FSPN submodel depicted in Fig. (**12**). The pattern is constituted by a place with an initial and an immediate transition which takes the token of the place and starts the first instruction of the INIT block (translated separately). After the execution, the token (*i.e.*, the control flow of the program) is transferred to the first place of a body-related FSPN (a Loop or a Trigger);

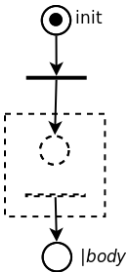

Fig. (12). FSPN translation of Init.

- LOOP: the FSPN model translating the LOOP block is depicted in Fig. (**13**). The pattern is constituted by a place (labeled with the *body* tag), a stochastic transition and the execution of a list of instructions. After the execution of these instructions, the control flow is transferred again to the start of the loop; the stochastic transition is characterized by a short delay time. Since the insertion of a delay between two iterations of the loop is a responsibility of to the programmer, who is in charge of inserting a WAIT command, a timed transition is due. This insertion is to avoid infinite zero-time loops that make a model simulation unfeasible;

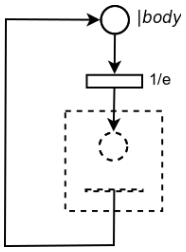

Fig. (13). FSPN translation of loop.

- TRIGGER: the structure of the FSPN translating a Trigger is very similar to the one of the Loop and it is depicted in Fig. (**14**). The main difference is having an immediate transition, which is enabled by a guard. Only when the guard is true, the transition can fire and the list of instruction of the Trigger are executed;

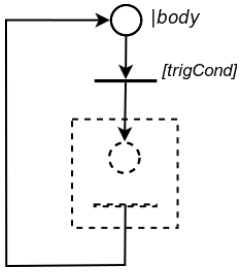

Fig. (14). FSPN translation of trigger.

- COMMAND: the way of translating a command is similar to other commands except for Clean. The main idea is to have "enabling places" for each action (*e.g.*, sell energy to other parties, store exceeding energy in storage facilities): by putting a token in these places (which are labelled with a *cmdEnabler*-like tag), the related action is enabled and can be accomplished in other FSPN modules. The FSPN model, which translates the Command instruction, first puts a token in such places and then transfer the control to the next instruction. Fig. (**15**) clarifies this concept;

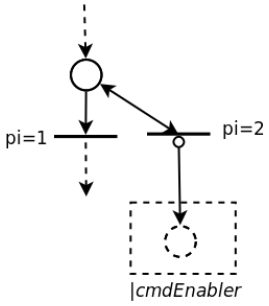

Fig. (15). FSPN translation of command.

- CLEAN: the FSPN model translating the Clean command has the opposite effect of the command FSPN model; it subtracts the enabling token (if present) from every enabling place. Fig. (**16**) graphically represents such model;

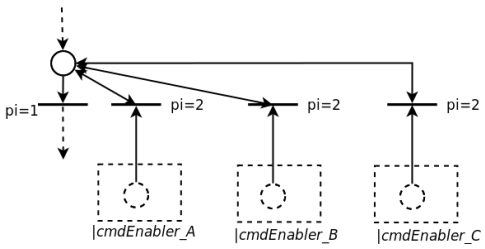

Fig. (16). FSPN translation of clean.

- IF – IF/ELSE: the decisional instructions can be simple if or if-else statements. For what concerns the if statement, when the control flow transfers the token in the first place of the decisional node, two immediate transitions are in permission to fire: if the *cond* condition is true, the submodel translating the instruction of the if block is activated, otherwise the control flow passes to the instruction following the entire decision node. Fig. (**17**) represents the FSPN submodel of the simple If.

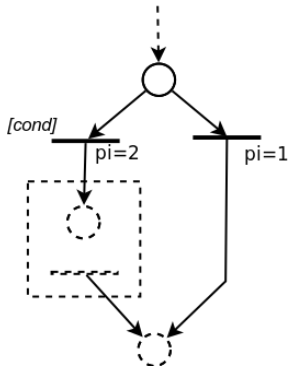

Fig. (17). FSPN Translation of If.

In the case of *if-else*, the failure of the condition provokes the flow of the control to the FSPN model translating the first instructions of the else-block: Fig. (**18**) depicts this case.

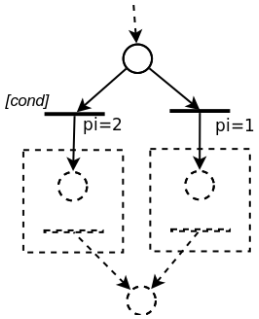

Fig. (18). FSPN Translation of If/Else.

- WAIT: the FSPN translating a wait instruction is simply constituted by a place and a stochastic transition with a rate equal to the inverse of the delay time. Fig. (**19**) clarifies this model.

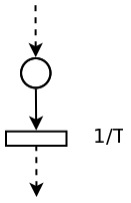

Fig. (19). FSPN Translation of Wait.

6. A CASE STUDY

Academic and industrial literature is rich of works that propose reference architectures for a Smart Home, even if none of them is a widely accepted standard. Nevertheless, smart home typically owns devices interacting with a DSO by exchanging energy on the electrical grid and information across a communication infrastructure for billing purposes.

Our model of Smart Home can produce and consume energy thanks to the presence of local generators (such as solar panels), local energy storages. As Smart Devices we have been considered the ones to manage the home environment climate (Heating and Cooling), typical domestic facilities (Washing Machine and Owen) and

entertainment devices.

To detail parameters of our model, we took into account consumptions performed by an average domestic customer, composed by three or four persons. Each considered smart device belong to the energetic class A, which is one of the classes defined by the EU Directive 92/75/EC [33]. Fig. (**20**) depicts the considered model of Smart Home. Table **1** details used configuration parameters.

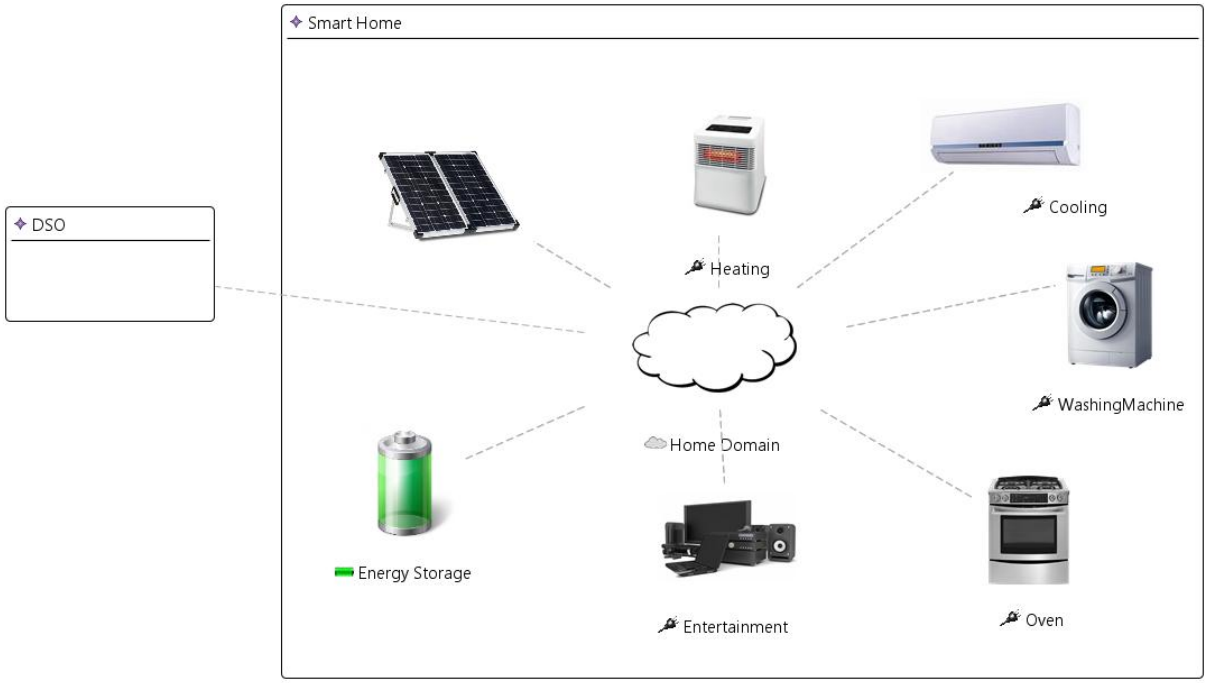

Fig. (20). Smart home model.

Table 1. Configuration of model parameters.

Name	Description	Value
Solar Panel		
Period	Generated energy during the day	• {energy=0.6 kW, start: 6.00, end: 10.00, Day:All} • {energy=1 kW, start: 10.00, end: 15.00, Day:All} • {energy=0.6 kW, start: 15.00, end: 17.00, Day:All}
Energy Storage		
Max storage	Maximum capacity of stored energy	• 3.0 kW
Washing Machine		
Consumptions	Energy consumptions of the washing machine that is activated more often at evening	• {energy=0.7 kW start: 18.00,endAt: 21.00, Day: All} • {energy=0.2 kW start: 21.00,endAt: 18.00, Day: All }
Owen		
Consumptions	Energy consumptions when the machine is activated during lunchtime and dinnertime	• {energy=0.5 kW start: 18.00,end: 19.00, Day: All} • {energy=0.4 kW start: 12.00,end: 13.00, Day: All}
Heating		
Consumptions	Energy consumptions considered for each day of the week	• {energy=2 kW start: 18.00,end: 23.00, Day: All}
Cooling		
Consumptions	Energy consumptions considered for each day of the week (use and peak)	• {energy=1.2 kW start: 11.00,end: 16.00, Day: All} • {energy=0.4 kW start: 16.00,end: 18.00, Day: All}
Entertainment		
Consumptions	Energy consumptions considered for each day of the week considering typical entertainment device (TV LCD, Personal Computer and console)	• {energy=0.3 kW start: 19.00,endAt: 23.00, Period: All}

Moreover, we enriched these data by considering buying and selling prices; according to [33, 34], some realistic prices are considered: the buying price (from the DSO to SH) is 0.4 €/kW while the sale price (from the SH to the DSO) is 0.25 €/kW.

Just to give an example, this SH model can be improved with the Application reported in the Listing **3**. The application works by storing energy in the battery and selling surplus if the energetic balance is positive (*i.e.*, an

energetic surplus); in case of energy need, the energy is acquired from the extern (the DSO) if the home does not have energy in the battery or if the bill count is below a certain threshold.

```
APPLICATION simple {
    INIT:
    LOOP:
        clean();
        if (ENERGY > 0) {
            store();
            sell();
        } else {
            if ((BILL > 50.0) && (STORED > 0 )) {
                take();
            } else {
                buy();
            }
        }
        wait(30,min);
```

Listing 3: Example application of the Smart Home.

Now, according to the process described in Section 3, the architectural model is translated into an FSPN model depicted in Fig. (**21**) while the code of the application is depicted in Fig. (**22**). All the rate parameters of the timed transitions, as well as the weight of the fluid arcs, are computed from the data reported in Table **1**.

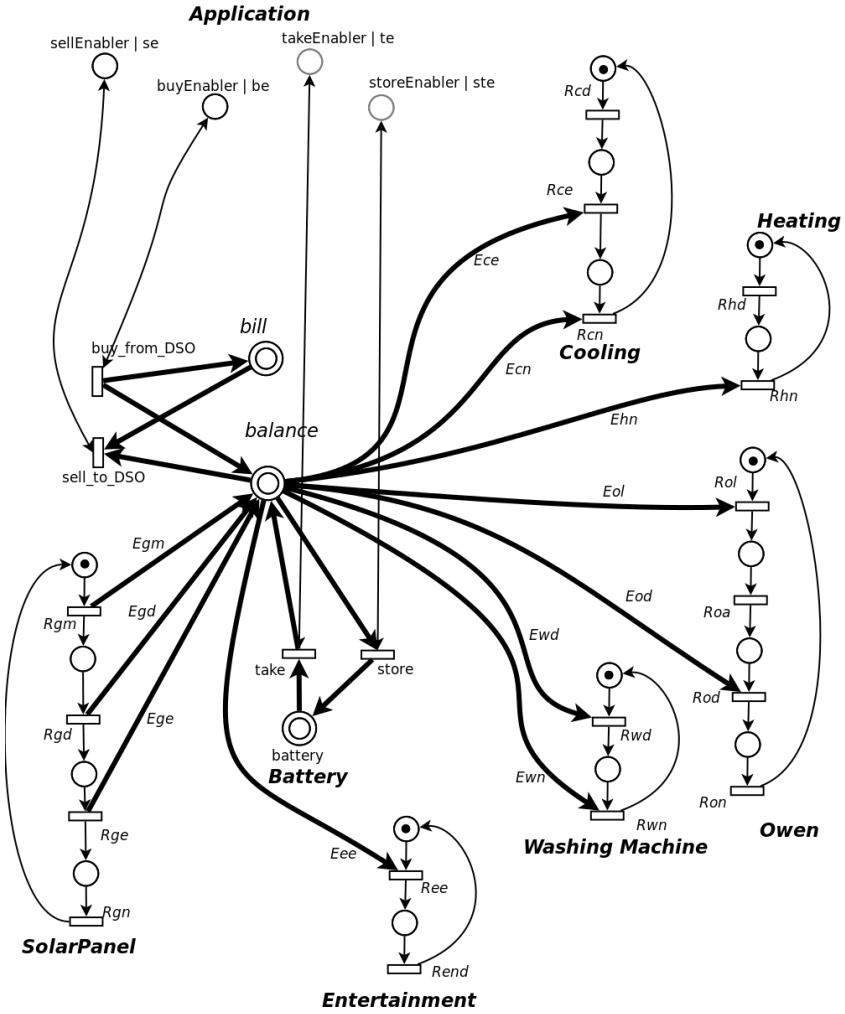

Fig. (21). FSPN model of the case-study architecture.

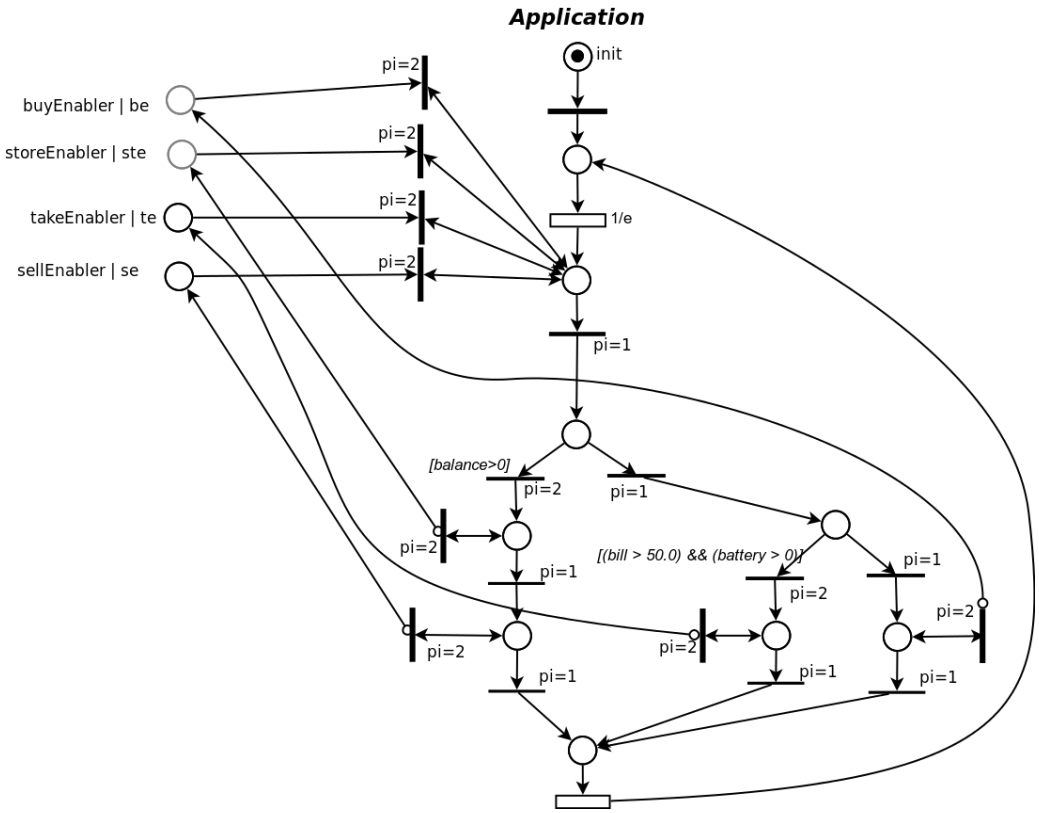

Fig. (22). FSPN model of the case-study application.

These FSPN models are merged into a single model (that is not depicted for the sake of space) by using the labels, which tag the four places *storeEnabler*, *sellEnabler*, *takeEnabler* and *buyEnabler*. For this case study, we propose a sensitivity analysis that evaluates the overall cost of the Smart Home with respect to variations of the power generated by the solar panels. Using this tool, a designer can verify how much the cost-benefits ratio varies and to quantify the payback period of a solar panel. The analyses have been conducted by means of the simulator described in [8] which took less than 30 seconds on a high-medium laptop (Intel Quad-core i7-2677M CPU 1.80GHz with 4 GB of RAM) in order to analyze a month long period.

Fig. (**23**) and Fig. (**24**) represent the results of such analyses: Fig. (**23**) plots the costs of the electrical bill with respect to the time while Fig. (**24**) represents the level of energy stocked in the battery in time. It is clear by reading the first plot that costs raise in the first period and then assess on a level of 50 euros: this is due to the specified application that inhibits SH to buy new energy when this limit is reached.

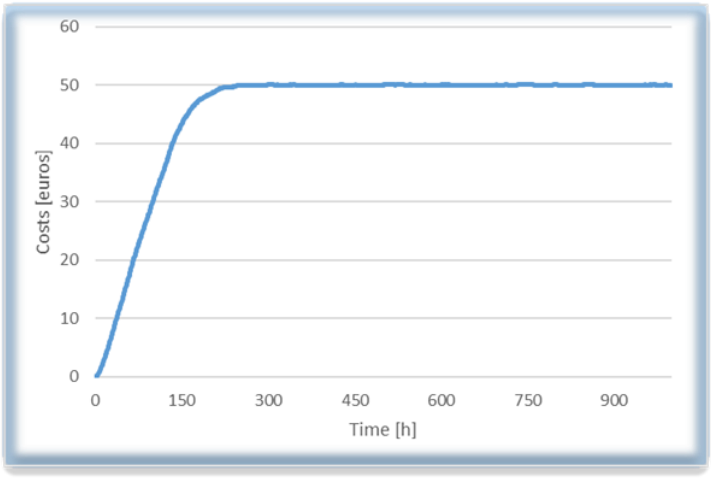

Fig. (23). Costs *vs*. time.

The proof of such statement can also be found in Fig. (**24**): coherently with the reach of the bill upper limit, the level of the battery starting decrease until it reaches a dynamic equilibrium (at about 0.3 kW). From this moment, this amount of battery can guarantee a sort of electrical independence.

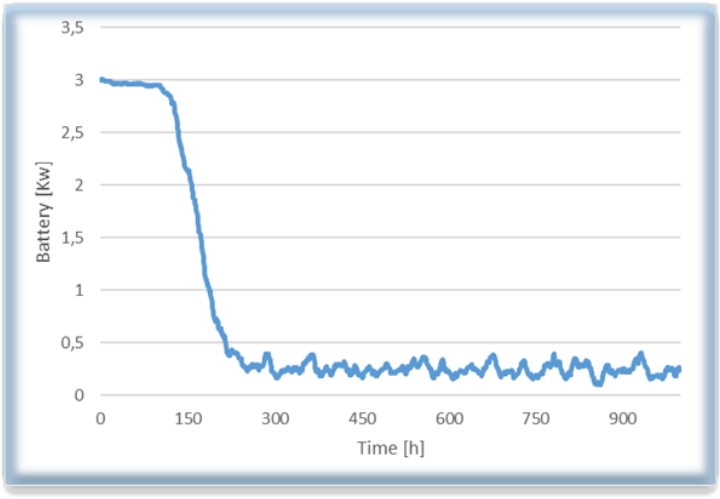

Fig. (24). Battery *vs*. time.

CONCLUSION

The proposed approach improves the traditional design processes for Smart Homes and Smart Grids environments by adding the capability to predict the costs of a specific configuration during a predefined period. Following this approach and the µGRIMOIRE environment, a designer can shorten time-to-market in building effective applications systems for smart grid systems. The main help resides in the realistic simulation of the designed configuration and the possibility to evaluate quantitatively different logics in a very simple way. In fact, an end-user of the µGRIMOIRE environment is supported by a graphical interface that allows the design of a smart home with primitive concepts, specific of the smart grid domain. The quantitative analysis is enabled by automatic transformations, which generate models that can be analysed by specific solvers.

The modelling approach is based on two pillars: the model-driven paradigm shifts the level of abstraction of the modelling activities to the modeller improving productivity and quality of the models; furthermore, formal models allow the µGRIMOIRE users to have quantitative evaluation also of continuous variables such as costs and energy levels. The realistic case study demonstrated how simple could be the modelling and the analysis of such problems and the determination of system properties to take proper designing decisions, at early stages of the Grid development. The future research effort will be directed towards an extension and an assessment of the languages by modelling several and different case studies.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] S.M. Kaplan, Smart Grid. "*Electrical Power Transmission: Background and Policy Issues*". The Capital. Net, Government Series, 2009, pp. 1-42.

[2] H. Pidd, "*India blackouts leave 700 million without power*". The Guardian, July 2012, Available from: http://www.guardian.co.uk/world/2012/jul/31/india-blackout-electricity- power-cuts

[3] J. Ditler, "*The Great Coronado Blackout of 2011*". Coronado Eagle & Journal, September 2011, Available from: http://www.coronadonewsca. com/news/article_43342c34-e07e-11e0-9982-001cc4c002e0.html

[4] J.A. Adams, "*Cyber Blackout: when the lights go out – nation at risk*". FriesenPress, 2015, ISBN: 9781460259818.

[5] R.K. Singh, and R. Katakey, Worst India Outage Highlights 60 Years Of Missed Targets. Bloomberg, August 2012, Available from: http://www.bloomberg.com/news/2012-08-01/worst-india-outage-highlights-60-years-of-missed-targets-energy.html

[6] M. Ficco, A. Daidone, L. Coppolino, L. Romano, and A. Bondavalli., "An Event Correlation Approach for Fault Diagnosis in SCADA Infrastructures". In: *13th European Workshop on Dependable Computing*, Pisa, Italy, Pisa, 2011, pp. 15-20, ACM. [http://dx.doi.org/10.1145/1978582.1978586]

[7] A. Drago, S. Marrone, N. Mazzocca, A. Tedesco, and V. Vittorini, "Model-Driven Estimation of Distributed Vulnerability in Complex Railway Networks", In: *Ubiquitous Intelligence and Computing, 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC),* IEEE, 2013, p. 380, 387. [http://dx.doi.org/10.1109/UIC-ATC.2013.78]

[8] U.S. Department of Energy. "*The smart grid: An introduction*". Available from: http://energy.gov/oe/technology-development/smart-grid.

[9] S. Kent., *Model driven engineering. Integrated Formal Methods*, vol. 2335 of LNCS, Springer Berlin Heidelberg, 2002, pp. 286-298. [http://dx.doi.org/10.1007/3-540-47884-1_16]

[10] M. Gribaudo, and A. Horvath., "Fluid Stochastic Petri Nets augmented with flush-out arcs: a transient analysis technique". *Software Engineering, IEEE Transactions on*, vol. 28, no.10, pp. 944,955, 2002. [http://dx.doi.org/10.1109/TSE.2002.1041051]

[11] U. Gentile, S. Marrone, N. Mazzocca, and R. Nardone., "A Cost-Energy Trade-off Model in Smart Energy Grids". In: *Proceedings of 2nd International Workshop on Cloud and Distributed System Applications in conjunction with 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, November 8-10, Guangzhou, China, ACM, 2014.

[12] M. Glinz., "For Requirements Specification - As Simple As Possible, As Rich As Needed". In: *International Workshop on Scenarios and State Machines*: Models Algorithms and Tools, Orlando, May 2002.

[13] J. Jurjens., "*Secure systems development with UML, Springer Science & Business Media*". 2005, ISBN:3642056350 9783642056352

[14] D. Petriu, and M. Woodside, "An intermediate metamodel with scenarios and resources for generating performance models from UML designs", *Software and Systems Modeling*, vol. 6, no. 2, pp 163-184, 2007. [http://dx.doi.org/10.1007/s10270-006-0026-8]

[15] S. Mustafiz, X. Sun, J. Kienzle, and H. Vangheluwe, "Model-driven assessment of system dependability". *Software and Systems Modeling*, vol. 7, pp. 487-502, 2007.

[16] S. Bernardi, F. Flammini, S. Marrone, N. Mazzocca, J. Merseguer, R. Nardone, and V. Vittorini., "Enabling the usage of UML in the verification of railway systems: the DAM-rail approach", *Reliability Engineering & System Safety*, vol. 120, pp. 112-126, 2013. [http://dx.doi.org/10.1016/j.ress.2013.06.032]

[17] R. Soley., "*Model driven architecture*". OMG white paper 308, Available from: http://www.omg.org/~soley/mda.html

[18] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, "*EMF: Eclipse Modeling Framework 2.0 (2nd ed.)*". Addison-Wesley Professional, 2009.

[19] L. F. Fernández, and A. V. Moreno, "An introduction to UML profiles. UML and Model Engineering", In: Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE. Cepis vol. 5, 2014.

[20] S. Ricciardi, F. Palmieri, U. Fiore, G. Careglio, and G. Santos-Boada., *Handbook of green information and communication systems, Towards Energy-Oriented Telecommunication Network,* 1st ed. Academic Press, 2012, pp. 491-515.

[21] A.H. Mohsenian-Rad, V.W. Wong, J. Jatskevich, and R. Schober, "Optimal and autonomous incentive-based energy consumption scheduling algorithm for smart grid", In: *Innovative Smart Grid Technologies (ISGT),* IEEE, 2010, pp. 1-6. [http://dx.doi.org/10.1109/ISGT.2010.5434752]

[22] A. Mishra, D. Irwin, P. Shenoy, J. Kurose, and T. Zhu, "SmartCharge: cutting the electricity bill in smart homes with energy storage", In: *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet,* ACM, 2012. [http://dx.doi.org/10.1145/2208828.2208857]

[23] M. Jahn, M. Jentsch, C.R. Prause, and F. Pramudianto, "The energy aware smart home", In: *5th International Conference on Future Information Technology,* IEEE, 2010, pp. 1-8.

[24] "Gridlab-d simulation software", 2011, Available from: http://www.gridlab.org

[25] US Department of Energy. "Energyplus: Getting started with energy plus, in basic concepts manual - essential information you need about running energyplus", 2011, Available from: https://energyplus.net/sites/default/files/pdfs_v8.3.0/GettingStarted.pdf.

[26] "PowerMatcher Smartgrid Technology. Basic structure and agent roles", 2015.

[27] M. Nysteen, H. Mynderup, B. Poulsen, and C. Trholt, "Simulation Tool For Energy Consumption and Production: The development of a simulation tool for measuring the impact of a smart grid on a building", In: *Proceeding of the 3rd International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, IARIA, 2013.

[28] A. Dey, N. Chaki, and S. Sanyal, "Sanyal, Modelling smart grids using Generalized Stochastic Petri Nets", *International Journal of Convergence Information Technology*, vol. 6, no. 11, pp. 104-114, 2011, AICIT.

[29] S. Chiaradonna, P. Lollini, and F. Di Giandomenico, "On the modelling of an instance of electric power systems". Technical report. 2006, Available from: http://rcl.dsi.unifi.it/publication/show/419-3.

[30] "SableCC Home Page", Available from: http://www.sablecc.org/ [accessed April 2015].

[31] S. Bernardi, S. Donatelli, and A. Horvàth, *Compositionality in the GreatSPN tool and its use to the modelling of industrial applications,* University of Arhus: Denmark, 2001.

[32] U. Gentile, S. Marrone, N. Mazzocca, and R. Nardone, "Cost-Energy Modelling and Profiling of Smart Domestic Grids, [accepted for publication]" *International Journal of Grid and Utility Computing*, Indescience 2016 [accecpted for publication].

[33] Italian Energy Authority, Available from: http://www.autorita.energia.it/it/elettricita/prezzirif.html.[accessed April 2015].

[34] GSE, Available from: http://www.gse.it/. [accessed April 2015].