SHORE: a model-driven approach that combines goal, semantic and variability models for Smart HOme self-REconfiguration

Denisse Muñante, Bruno Traverson, Sophie Chabridon, Amel Bouzeghoub

SHORE: a model-driven approach that combines goal, semantic and variability models for Smart HOme self-REconfiguration

Denisse Muñante[1][a], Bruno Traverson[2], Sophie Chabridon[3][b] and Amel Bouzeghoub[3][c]
[1]*ENSIIE & SAMOVAR, Évry, France*
[2]*EDF R&D, Saclay, France*
[3]*SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Evry, France*
denisse.munantearzapalo@ensiie.fr, bruno.traverson@edf.fr, {sophie.chabridon, amel.bouzeghoub}@telecom-sudparis.eu

Keywords: Goal-orientation, ontology, feature models, Smart Home, self-reconfiguration

Abstract: Smart Homes, and more generally the Internet of Things (IoT), are gaining more and more audience, but this kind of environment is still challenging on several aspects. Semantic representations, mainly ontologies, have been used to cope with the complexity and lack of interoperability due to the wide variety of objects and services. However, these semantic representations do not permit a fine description of variability and evolution to achieve particular objectives. Therefore, in this paper, we introduce a model-driven approach, so-called *SHORE* for Smart HOme self-REconfiguration, which addresses these limitations and covers the complexity and variability of Smart Homes allowing automated reconfiguration guided by their goals. Thus, SHORE includes artifacts, based on goal, semantic and feature models, that are used at *design-time* to conceptualise Smart Homes, their variants and goals. Whilst, at *runtime*, SHORE exploits these artifacts to detect situations that require reconfiguration, *i.e.,* detect deviations from their goals, and then to calculate optimal reconfiguration plans to cope with these situations. Finally, we present a case study which illustrates our approach and highlights its expressiveness.

1 Introduction

A Smart Home relies on information and communication technologies for controlling household appliances and other domestic equipments remotely. It is becoming possible to adapt the behavior of a house to the habits of its residents. Dynamic adaptation can take place as soon as some change is detected, like a decrease of the external temperature requiring to switch on the heating system in order to keep indoor temperature comfortable. The automation degree of a Smart Home can be very high by defining elaborated rules and targeted goals, related to the expected temperature of the house for instance, in order to provide added-value services to the residents. Thus, self-regulation of the Smart Home services could be a good means when any deviation from the Smart Home goals is detected. Deviations from goals are henceforth referred to as *undesired situations* and should be addressed.

The dynamic adaptation of a system is conceived as a way to avoid or correct the degradation of its quality of service during its execution, which may be due to changes in environmental conditions. A substantial research effort has been done towards defining methods and techniques for managing self-adaptive systems (SASs). Model-driven approaches [Cámara et al., 2017] and in particular model@run.time methods have been proposed to specify how self-adaptive systems should react in dynamically changing runtime environments [Bencomo et al., 2019, Erazo-Garzón et al., 2021]. Regarding Smart Homes, self-adaptation is guided by ontologies, *e.g.,* [Seydoux et al., 2016, Lork et al., 2019]. Ontologies allow to address the interoperability of the wide variety of Smart Homes appliances and services.

However, challenges remain to reach a high expressiveness and ease of use of dynamic reconfiguration both at design and execution times. Adaptation rules should be expressed at a high level of abstraction to be understood by inhabitants and should be managed dynamically to evolve with regards to modifications in environment properties and user preferences. Also, even more importantly, in case of multi-

[a] https://orcid.org/0000-0003-2621-8342
[b] https://orcid.org/0000-0002-1591-6754
[c] https://orcid.org/0000-0003-4890-9005

ple services operating in a same area of a Smart Home and thus sharing a set of sensors and actuators, potentially conflicting goals should be dealt with when the Smart Home requires adaptation. One optimal solution should be elected among several alternatives in a manageable and responsive way.

Therefore, our research objective concerns the building of a tool-supported approach for the **S**mart **HO**me services **RE**configuration, so-called the SHORE approach. Towards its definition, we investigate how to combine semantics, based on ontologies, for Smart Homes and their variability models, and how to exploit the resulting models during the whole life-cycle of the system, including *design-time* and *runtime* phases. More specifically, in this paper we focus on the following two research questions:

- *RQ1* - How to define Smart Home variants *at design-time* allowing reconfiguration *at runtime*?
- *RQ2* - How to select an optimal Smart Home variant to overcome undesired situations?

To answer the first research question, we start from the state of the art of standardised semantics (*i.e.,* ontologies) for Smart Homes, *e.g., SAREF* [Smart Appliances, 2017]. Then, we investigate how to extend these standards to add notions that allow the self-reconfiguration of Smart Home systems, *i.e.,* notions to define Smart Home goals, *e.g.,* i* [Yu, 1997], and variants using variability models, *e.g., feature models* [Kang et al., 1990]. Concretely, starting from a goal-orientation viewpoint, we elicit the SHORE artifacts, which are used at design-time. Thus, SAREF$^+$ and FM$^+$ were proposed as extensions of SAREF and feature models respectively. SAREF$^+$ allows to describe Smart Homes by adding notions around their goals. Based on this description, automated rules to detect situations that require adaptation can be formulated. Whilst FM$^+$ complements this description by adding Smart Home variants. Based on these variants, we can find correct adaptations to cope with detected situations.

Concerning the second research question, we explore existing techniques to exploit gathered data when taking reconfiguration decisions in order to deal with conflicting goals. In particular, we use the optimisation approach based on *grammar guided genetic programming* that was introduced in a previous work [Kifetew et al., 2017a, Muñante et al., 2018].

This paper is organised as follows: Section 2 introduces the problem formulation for the self-reconfiguration of Smart Homes services. Section 3 gives background on goal-oriented models, semantic for Smart Homes, and variability modelling techniques. Section 4 describes the SHORE approach to support this reconfiguration. Section 5 discusses related works and Section 6 identifies next steps and concludes the paper.

2 Problem Formulation

Roughly speaking, Smart Homes are composed of *devices* (actuators) that offer *services* to make the life of residents more pleasant. Thus, one crucial goal is to keep *user comfort* high. However, Smart Homes have been also created to accomplish other *goals*, for instance to keep *energy consumption* low or at least under control. These goals might be in conflict when they cannot be reached at the same time. The potential goal conflicts could be solved using *priorities* and *variants* for services' settings. These variants may reach the goals at different levels of accomplishment, being in the range from no compliance to full compliance. While many techniques might be used to manage conflicts, we focus on priorities as they have widely shown their efficiency in other research domains such as security [Breaux and Antón, 2008].

In this paper, we address the problem of Smart Home self-reconfiguration at runtime, taking into consideration priorities for Smart Homes goals and variants for their services' settings. This problem involves choosing the configuration (variant) for the Smart Home's services that best respects the goals and their priorities. Hence, the problem of Smart Homes services configuration can be defined as follows. Given:
M **goals** for the Smart Home $SmartHome_{goals} = \{goal_1, goal_2, ..., goal_M\}$,
M **priorities** for the Smart Home goals (one priority per goal) $SmartHome_{priorities} = \{prio_1, prio_2, ..., prio_M\}$,
R **actuators** that change the state of the Smart Home $SmartHome_{actuators} = \{act_1, act_2, ..., act_R\}$,
P **services** offered by the Smart Home actuators to achieve their goals (an actuator can offer 1 or n services) $SmartHome_{services} = \{service_1, service_2, ..., service_P\}$.
Services dependencies are twofold: i) inclusion dependency: $service_a$ includes/requires $service_b$ and ii) exclusion dependency: $service_a$ excludes/refuses $service_b$.
Each service contains a set of configuration variants. For instance:
X variants for service A: $serviceA_{variants} = \{variant_1, variant_2, ..., variant_X\}$, and we complete the configuration variants for the other P services,

Then, the **problem formulation of the Smart Home services configuration** is defined as *the selection of the P variants (one per service) that best respect the M Smart Home goals and their priorities*.

In order to select a good combination of variants of the Smart Home services, we should define metrics that will be associated to these variants. Thus, the above problem formulation is updated to add the mentioned metrics, henceforth called quality attributes (QAs), as follows. Given:
S **QAs** associated to each configuration variant $SmartHome_{QualityAttributes} = \{qa_1, qa_2, ..., qa_S\}$,
2 **operators** for evaluating the QAs $\{min, max\}$, *i.e.,* minimise and maximise the values of QAs,

Then, the **updated problem formulation** is defined as *the selection of the P variants that best respect the M Smart Home goals and their priorities* ***with respect to the evaluation (min or max) of the S QAs***.

To tackle the problem of the Smart Home services reconfiguration, we add the following variables. Given: the **current configuration** of the P Smart Home's services $Config_{cur} = \{variantA_m, variantB_n, ..., variantP_z\}$, the **alert** that contains information on the deviation of any Smart Home goal, and the current values of the S QAs of the Smart Home's services configurations $\{cv_1, cv_2, ..., cv_s\}$ (cv_1 corresponds to the current value of qa_1, .. and cv_s corresponds to the current value of qa_s,),

The **decision-making for the Smart Home reconfiguration** is to find a new configuration $Config_{new} = \{variantA_{m'}, variantB_{n'}, ..., variantP_{z'}\}$ of its services, such that: $Config_{new}$ is better than $Config_{cur}$. The definition of being "better" is based on a *fitness function* which computes an aggregated value of the QAs that reflects a configuration of the Smart Home that increases the level of achievement of its goals. Since QAs could have different natures, a normalization technique is employed as shown in Equations 1 and 2 ([lv,rv] represents the interval from the minimal to maximal values that could take a QA). The minimisation problem is translated in a maximisation one. Then, the aggregated fitness value of all the variants in $Config_{new}$ is calculated.

$$max(var_x) = \sum_{i=1}^{m} prio_i * (qa_i - lv)/(rv - lv) \quad (1)$$

$$min(var_x) = \sum_{i=1}^{n} prio_i * [1 - (qa_i - lv)/(rv - lv)] \quad (2)$$

3 Background

3.1 Ontology-based approaches for Smart Home

An ontology refers to the shared understanding of some domain of interest which may be used as a unifying framework to solve complex problems [Uschold and Gruninger, 1996]. In the domain of Smart Home, several ontologies have been proposed – see the study presented by Bajaj *et al.* in [Bajaj et al., 2018] for an in-depth survey of IoT ontologies. Among them, the Smart Appliance REFerence SAREF[1] is a reference ontology for smart appliances. It was developed with the support of the European Commission and is published by ETSI [Smart Appliances, 2017].

SAREF mainly focuses on devices that are "tangible objects designed to accomplish a particular task in households, common public buildings or offices" and may be further refined as sensors or actuators. From a user perspective, a device is designed to achieve a task (`Task`) related to a feature of interest (`Commodity`) using a specific strategy (`Profile`). In case of sensors, devices make measurements (Measurement) related to some properties (`Property`) and in specific units (`Unit of measure`). Devices are accessed via an interface (`Service`) implemented on top of functionalities (`Function`) that are composed of commands (`Command`) and states (`State`).

SAREF is intended to be a core ontology that may be extended for specific domains. Hence, extensions have already been published in several domains like Energy, Agriculture, Health, etc. However, some limitations of SAREF can be identified. For instance, devices are only described as atomic, i.e., they are not composite, not allowing to include complex devices. Also, services are not composable and are tightly coupled to devices preventing a higher level of abstraction. Moreover, services cannot be customised, this means that services cannot be configured/regulated according to some parameters. Finally, potential conflicts between Smart Home goals cannot be managed.

3.2 Variability Models in the Software Product Line

Software Product Line Engineering (SPL) follows the principles of product lines to enable product mass customisation. SPL rests on the idea of characterising the products in terms of the features they offer, and categorise them into common features that are part of each product and variable features that are only part of

[1] `https://forge.etsi.org/rep/SAREF`

some products. The selection of a set of suitable features when a product is configured is a crucial activity that has been widely studied both in academia and industry. Such selection is typically made by exploring the space of trade-offs along different attributes of interest, for instance *cost* and *value*.

Feature models (FMs) [Kang et al., 1990] are often used in SPL to describe the commonalities and variability of product families. In FMs, features are hierarchically organised by means of parent-child and tree constraint relations. Tree constraint relations are: Mandatory, Optional, Alternative, and Or. If a feature has a `Mandatory` relation with its parent, it must be included in all software configurations in which its parent appears. If a feature has an `Optional` relation with its parent, it can be optionally included in products in which its parent appears. A set of features are grouped as `Alternatives` if exactly one of these features has to be included when their parent appears in the configuration. A set of features are grouped using an `Or` relation if one or more of these features can be included when their parent appears in the configuration. An overview of it is shown in Figure 1 (a).

In addition to the previous relations, a FM may also include cross-tree constraints between features: Implies and Excludes. If featureA `implies` featureB, and featureA appears in a product, then featureB must be selected for the product. If featureA `excludes` featureB, and featureA appears in a product, then featureB must not be selected for the product.

[Batory, 2005] introduces a mapping between FMs and grammars allowing to apply formal reasoning. Our mapping from tree constraints in FMs to grammars is summarised as follows:

Optional: `OPTIONAL(F)` → `<F_opt> ::= <F> |` λ

Alternative: `ALTERNATIVE(F1,F2)` → `<F> ::= <F1> | <F2>`

Or: `OR(F1, F2)` → `<F_or> ::= <F> <F_or> | <F> ;`
`<F> ::= <F1> | <F2>`

And (*): `AND(F1,F2)` → `<F> ::= <F1> <F2>`

(*) `And` was included to capture grouped features with the same parent but without alternative/or relations.

Non-terminals are represented by names in angle brackets while terminals are represented by quoted strings. For the sake of simplicity, λ rules, which mean *empty*, are represented by a `|`.

Cross-tree constraints in FMs could be suffixed to the grammar so that they can be read and applied by a program. In our case, we adopt the propositional logic notation proposed by [Batory, 2005].

Attributes are associated to each feature so that optimal product configurations can be derived using the values of the attributes.

4 SHORE: An approach for Smart HOme self-REconfiguration

[Weyns, 2019] introduces a conceptual model for building SASs. It is mainly composed of the *environment*, *managed system*, *managing system* and *adaptation goals* components. We envision the SHORE approach by leveraging the high level components of Weyns' model. The *environment* refers to the external world with which the SAS interacts. For instance, the environment of a Smart Home includes the weather, which can influence the levels of humidity, temperature and illuminance. The *managed System* comprises the application code that realises the system domain functionality. For instance, in the case of Smart Homes, services that switch on/off actuators or configure their settings in order to accomplish a specific purpose such as the regulation of levels of temperature, humidity or carbon emissions inside building spaces. The *adaptation Goals* are concerns of the managing system over the managed system to preserve certain conditions or qualities. For instance, an adaptation goal of a Smart Home could be to *keep residents comfort high*. It can be translated as reaching the acceptable values for the temperature, humidity or level of carbon in a room. These acceptable values could be established using standardised regulations or users' preferences. The *managing System* conducts the managed system, it comprises the adaptation logic that deals with one or more adaptation goals. Different terms may be used to refer to a managing system, *e.g.,* autonomic manager, adaptation engine, reflective subsystem or controller.

The architecture of managing systems is usually based on the MAPE-K (Monitor, Analyse, Plan, Execute, over shared Knowledge) loop [Kephart and Chess, 2003]. *Monitor* allows to collect the system environmental context, besides components of the managed system. In a Smart Home, sensors collect data about current levels of temperature, humidity and carbon concentration in a room. Using the monitored data, *Analysis* determines if the adaptation of the managed system is needed or not. For instance, if the specified goals (see the adaptation goals component) are violated, an alert is sent. Next, *Plan* evaluates a set of system variants in order to choose the most adequate one to address the situation alerted to by the analysis. Finally, *Execute* takes into account the architecture model of the managed system in order to deploy the adaptation plan at runtime. In the Smart Home context, we should take into account the actuators present in the house, for instance an HVAC (Heating, Ventilation, Air-Conditioning) system.

In the next subsections, we answer the research

Table 1: Variables needed for the Smart Home services self-reconfiguration. Where: *min = minimisation, max = maximisation, - means that the variable is not supported by the formalism, + means in addition to its predecessor, and ≡ means equivalent to its predecessor. (*) Notice that profile measurement is related to profile and not to variants.*

Variable	**SAREF**	**FM**	**SAREF$^+$**	**FM$^+$**
Goals	profile	*select new configuration*	≡	+ *optimising (min, max) QAs*
Priorities of Goals	-	-	priority for profile	weight of QAs
Actuator	device	feature	actuator	≡
Service of Actuators	service	child features of actuators	service / composite service	≡
Dependency of services	-	imply, exclude	-	≡
Variants for services	-	child features of services	-	≡
QAs for variants	-	-	profile measurement for profile (*)	attribute where attribute is QA
Values for QAs	-	-	-	default value of QAs
Running configuration	-	feature conf.	-	≡

questions. We thus present the SHORE artifacts that are used at design-time and exploited at runtime. It is worth to mention that SHORE mainly supports the *analyse* and *plan* steps of the MAPE-K model.

4.1 RQ1:

How to define Smart Home variants *at design-time* allowing reconfiguration *at runtime*?

To answer RQ1, we use the variables identified in the problem formulation (see Section 2). Thus, the variables used to semantically define the variants of Smart Homes services are: *goals*, *priorities* of these goals, *actuators*, *services* offered by actuators, *variants* of services settings, *quality attributes* associated to these variants, the values for quality attributes and the running/current Smart Home services configuration. Notice that for this paper, we only consider the reconfiguration for services of actuators and not for services of sensors. It is because we consider that actuators are the only ones to be directly implied in the changing of the state of Smart Homes. Whilst sensors are required to evaluate Smart Homes states gathering environmental contextual data.

Table 1 shows how to represent the identified variables using SAREF and/or FMs formalisms. As observed, not all of the variables can be expressed by one of the two formalisms. To fill this gap, we introduce an extension of SAREF, called *SAREF$^+$* [Baghli et al., 2018], and an extension of FMs, called *FM$^+$* [Muñante et al., 2018], and we integrate them. SAREF$^+$ is mainly dedicated to define semantically the Smart Home, whilst FM$^+$ complements this definition adding variants for the service configuration.

In SAREF$^+$, `actuators` are explicitly defined. The concept of `service` is central and is highly flexible. It may be composed, a service can be a composition of multiple services, and even dedicated to composition, *e.g.,* an orchestrator service that arbitrates conflict among concurrent services. It may be loosely coupled to objects (or composition of objects) enabling the implementation of a service on a variety of objects matching the service specification and dynamic replacement in case of failure or other context evolution. On the other hand, in FM$^+$, `actuators` and `services` are represented as features. Thus, services are child features of actuators. Moreover, `dependencies` between services can be expressed through the cross-tree constraints: `imply` and `exclude`. Notice that, FMs tend to represent everything as features. In order to keep the domain specific terminology of Smart Homes, we need to use of an ontology such as SAREF.

In SAREF$^+$, a service is specified not only by its interface but also by its dynamic behaviour using SWRL rules. Static and dynamic aspects of services may be customized using profile (`goal`) and profile measurements, for instance `priorities`. A profile measurement describes the metric used to evaluate the achievement level of a certain profile. For instance, in a profile such as *energy efficiency* or *renewable energy*, profile measurements could be *energy consumption* or *type of energy source*. In FM$^+$, `goals` are not explicitly represented but their achievement are interpreted as the selection of an optimal configuration that will be "better" than the current configuration. To do that, a space of variants should be evalua-

ted. `Variants` are represented as features, and they represent the alternatives for services configurations. Thus, variants are child features of services. To select "better" configurations, we use quality attributes (QAs) that are associated to variants. QAs and profile measurements accomplish the same objective, however they are not applied at the same level of analysis. Profile measurements are related to profile (see (*) in Table 1) (*i.e.,* goals), while QAs are related to variants. The `priorities of goals` in FM$^+$ are expressed as the *weights* for QAs, and the `values for QAs` are defined using a *default value* of QAs. Moreover, the *minimum range value* and *maximum range value* are used to compare QAs of different nature.

Finally, the current running configuration of the Smart Home is only represented by the feature configuration model derived from a FM$^+$ model.

4.2 RQ2:

How to select an optimal Smart Home variant to overcome undesired situations?

As observed, several variables are involved in the Smart Home services self-reconfiguration. It quickly makes the problem more complex to solve with the exponential growth of variables when new actuators, services and variants are added to the Smart Home. Therefore, we address this problem as an *optimisation problem* (an not an exact problem) allowing the use of optimisation (meta-heuristic) techniques widely adopted in the literature.

Changes originated by residents or the environment can provoke deviations from Smart Home goals These deviations can be detected through SWRL rules. If a rule antecedent holds true, it means that an alert will be sent to the planning of the reconfiguration. An alert contains information about the situation that we cope with, hence it contains constraints to be respected. It is expressed as thresholds, for instance an attribute value should be greater or less than $Xvalue$. Therefore, SHORE not only finds good solutions for the software configuration, in addition it finds **constrained** optimal solutions for it.

For the plan step, given a FM, the software product configuration problem involves the exploration and selection of the set of products representing optimal trade-offs among the various attributes of the FM. We propose to use the grammar guided genetic programming (GGGP) [Kifetew et al., 2017b] for exploring the space of product configurations (see Figure 1 (a) and (b)). GGGP is a type of genetic programming (GP) that provides a suitable representation of candidate solutions as trees that capture the relationships among features imposed by the FM. Such a representation facilitates the generation of candidate solutions and evolving them to optimal solutions.

GP is part of Evolutionary Algorithms (EA). An overview of a typical EA is shown in Figure 1 (a). An EA starts by creating an initial population of individuals. It then evaluates each individual in the population by means of an appropriate fitness function and assigns it a fitness value. Once individuals are assigned fitness values, the EA proceeds by selecting individuals (to become parents) from the current population and subjects them to the process of recombination or crossover resulting in offspring. The selection process could be performed using a variety of methods based on the fitness value. Commonly used selection procedures include fitness proportional selection, tournament selection, and rank based selection. Once offspring are produced via recombination, they could further be subjected to a process of mutation with the aim of introducing diversity into the population. The EA then selects, from the combined pool of parents and offspring, the individuals that form the new population in the next generation (survivors). Survivor selection could also be performed in a number of ways (e.g., based on fitness of individuals or based on age). This process continues to iterate until some stopping condition is reached, in which case the EA terminates.

For the SHORE solution, as shown in Figure 1 (a), the FM is first serialised into a grammar (as described in Section 3), which is then fed to the GGGP-based search module, which follows the evolution cycle of a typical evolutionary algorithm, resulting in a Pareto-optimal set of product configurations. Given the nature of the problem at hand, we propose *multi-objective* optimisation, in particular we adopt the dominance-based approach as implemented in the NSGA algorithm [Deb et al., 2000]. The steps executed by the SHORE solution are described as follows:

Candidate encoding and initialisation Each candidate solution is a set of features, represented as a *derivation tree* based on the given grammar (an example is shown in Figure 1 (b)). Collecting the leaves of the tree, we get the set of features represented by the candidate solution (*i.e.,* the combination of variants for the Smart Home services settings according to SHORE). The population is initialised by randomly generating candidates from the grammar following the process of derivation [Kifetew et al., 2017b]. Candidate solutions are by definition valid with respect to the FM (grammar).

Fitness evaluation The fitness function is computed based on the values of QAs defined a priori. Then, given a candidate configuration solution $Config = \{variant_{1..n}, variant_{P..m}\}$ (see Figure 1 (b)), the fitness value of $Config$ is composed of the aggregate values

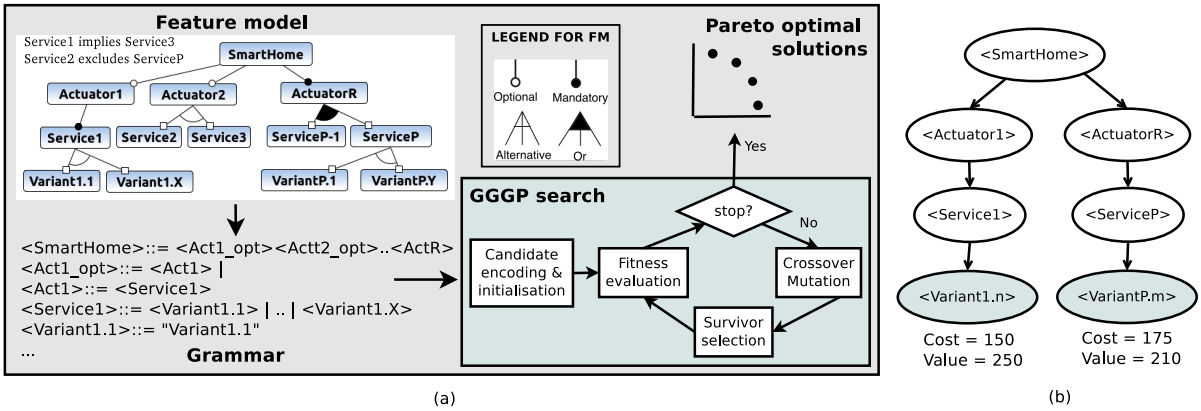

Figure 1: The SHORE approach: (a) the optimisation process, and (b) a candidate evaluation.

of each QA in $Config$. Each attribute is an *objective* to be optimised. For example, the sum of the `cost` of all features (*i.e.,* variants for the Smart Home services settings) in $Config$ represents one objective, while the sum of `value` levels constitutes another objective. Notice that, candidate solutions whose fitness values or specific attributes do not respect restrictions imposed by alerts and sent by the analysis step, are discarded.

Search operators Since candidate solutions are encoded as *trees*, the operators we employ are *subtree crossover* and *subtree mutation* [Kifetew et al., 2017b]. In *subtree crossover* step is as follows: Given two parents $Parent_1$ and $Parent_2$, subtrees of the same type (rooted at the same grammar non-terminal) are identified in each parent and swapped, producing the *offspring*. In *subtree mutation* step, an individual $Offspring_2$ is transferred to $Mutant$ replacing a subtree by a newly generated subtree from the grammar.

5 Related Works

Several mathematical proposals have been introduced to manage energy consumption or user comfort for Smart Homes [Tashtoush et al., 2005]. However, a methodology supporting engineers when conceiving Smart Homes since the evaluation of their goals using metrics is rarely presented. SHORE aims to fill this gap, nevertheless the integration of mathematical proposals to SHORE can facilitate/improve the defining of variants and their (quality) attributes.

Ontologies are employed to cope with the complexity of Smart Homes. For instance, IOT-O [Seydoux et al., 2016] supports self-reconfigurable IOT systems by combining state-of-the-art ontologies such as Semantic Sensor Network (SSN). Thus, IOT-O can be used in the MAPE-K control loop. Moreover, in [Lork et al., 2019], an ontology-based framework for the energy management in buildings is presented. For this, a benchmarking module identifies situations for the building energy inefficiency, and an evaluation and control module determines and executes the actions that alleviate these situations. We differ from these two referred works in the formulation of the planner for the reconfiguration. While these works focus on rule-based approaches without any strategy for managing potential conflicts between the adaptation goals, we address this issue by an heuristic algorithm that manages conflicts and scales the problem formulation.

Regarding quality metrics for self-adaptive systems, QoSMOS [Calinescu et al., 2011] defines a service-based application as a Discrete Time Markov Chain to identify the service configurations that satisfy the quality goals. QoSMOS uses an exhaustive verification limiting adaptation decisions to small-scale settings. To address this problem of scalability, MARTAS [Weyns et al., 2018] combines quality models at runtime and statistical techniques to predict situations that require adaptation for IoT systems. However, no details about the decision making process for the planning adaptation were provided. The work presented in [De Sanctis et al., 2019] is the closest to SHORE. It allows to define QAs associated to the devices settings that guide the selection of optimal configurations. However, nothing is mentioned about potential conflicts between adaptation goals and the technique employed to implement the optimisation problem. SHORE includes priorities to manage conflicts between adaptation goals and implements a GGGP for the selection of optimal configurations. GGGP generates valid candidates for the given grammar, thus no validation steps is needed once candidates are generated.

6 Conclusion and Next Steps

In this paper, we introduced a new approach called SHORE. It is composed of a set of artifacts that allow the reconfiguration of Smart Home services settings. SHORE enhances Smart Home automation by combining the fine grain expressiveness of ontologies together with feature models allowing to determine at design-time different variants of a Smart Home behaviour by sharing and prioritising goals. At runtime, when conflicting goals appear in candidate configurations, genetic programming allows to automate further the decision process. Our contributions include extensions of the *SAREF ontology* and *FMs*, the formulation of a genetic programming technique that exploits these formalisms, and their implementation for a case study using a set of open-source tools.

We are currently working on complementing the SHORE approach to determine at runtime undesired situations as early as possible by a monitoring system and also to translate a calculated configuration to the target configuration of the involved appliance in the execution step of the MAPE-K model. Furthermore, empirical evaluations are part of our next steps to validate the performance and usability of SHORE.

REFERENCES

Baghli, R., Najm, E., and Traverson, B. (2018). Defining services and service orchestrators acting on shared sensors and actuators. In *6th Int. Conf. MODELSWARD, Funchal, Madeira*.

Bajaj, G., Agarwal, R., Singh, P., Georgantas, N., and Issarny, V. (2018). 4W1H in IoT Semantics. *IEEE Access*, 6.

Batory, D. S. (2005). Feature Models, Grammars, and Propositional Formulas. In *9th Int. Conf. on Software Product Lines (SPLC), Rennes, France*, pages 7–20.

Bencomo, N., Götz, S., and Song, H. (2019). Models@run.time: a guided tour of the state of the art and research challenges. *Softw. Syst. Model.*, 18(5).

Breaux, T. D. and Antón, A. I. (2008). Analyzing regulatory rules for privacy and security requirements. *IEEE Trans. Software Eng.*, 34(1):5–20.

Calinescu, R., Grunske, L., Kwiatkowska, M. Z., Mirandola, R., and Tamburrelli, G. (2011). Dynamic QoS Management and Optimization in Service-Based Systems. *IEEE TSE*, 37(3).

Cámara, J. et al. (2017). Self-aware Computing Systems: Related Concepts and Research Areas. In *Self-Aware Computing Systems*, pages 17–49. Springer.

De Sanctis, M., Spalazzese, R., and Trubiani, C. (2019). Qos-based formation of software architectures in the internet of things. In *13th ECSA, Paris, France*, volume 11681 of *LNCS*. Springer.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2000). A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6:182–197.

Erazo-Garzón et al. (2021). Models@ runtime and internet of things: A systematic literature review. In *2d Intl Conf. ICI2ST*. IEEE.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, CMU SEI.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kifetew, F. M., Muñante, D., Gorroñogoitia, J., Siena, A., Susi, A., and Perini, A. (2017a). Grammar based genetic programming for software configuration problem. In *9th Int. Symp. on Search Based Software Eng. SSBSE*, volume 10452 of *LNCS*. Springer.

Kifetew, F. M., Tiella, R., and Tonella, P. (2017b). Generating valid grammar-based test inputs by means of genetic programming and annotated grammars. *Empirical SW Eng.*, 22(2):928–961.

Lork, C., Choudhary, V., Hassan, N. U., Tushar, W., Yuen, C., Ng, B. K. K., Wang, X., and Liu, X. (2019). An Ontology-Based Framework for Building Energy Management with IoT. *MDPI Electronics*, 8(5).

Muñante, D., Kifetew, F. M., Gorroñogoitia, J., Schaniel, R., Perini, A., and Susi, A. (2018). Model Driven Software Reconfiguration by Exploiting Grammar Based Genetic Programming. In *8th IEEE MoDRE@RE Workshop, Banff, Canada*.

Seydoux, N., Drira, K., Hernandez, N., and Monteil, T. (2016). IoT-O, a Core-Domain IoT Ontology to Represent Connected Devices Networks. In *20th Intl. Conf. EKAW*, volume 10024 of *Springer LNCS*.

Smart Appliances (2017). SmartM2M; Smart Appliances; Reference Ontology and oneM2M Mapping.

Tashtoush, B., Molhim, M., and Al-Rousan, M. (2005). Dynamic model of an HVAC system for control analysis. *Energy*, 30(10):1729–1745.

Uschold, M. and Gruninger, M. (1996). Ontologies: principles, methods and applications. *Knowl. Eng. Rev.*, 11(2):93–136.

Weyns, D. (2019). Software Engineering of Self-adaptive Systems. In *Handbook of Software Engineering*, pages 399–443. Springer.

Weyns, D., Iftikhar, M. U., Hughes, D., and Matthys, N. (2018). Applying architecture-based adaptation to automate the management of internet-of-things. In *12th ECSA*, volume 11048 of *LNCS*. Springer.

Yu, E. S. K. (1997). Towards modeling and reasoning support for early-phase requirements engineering. In *3rd IEEE Intl Symp. on Requirements Engineering (RE), Annapolis, MD, USA*.