International Journal of Smart Home
Vol.9, No.7 (2015), pp.135-150
http://dx.doi.org/10.14257/ijsh.2015.9.7.14

Applying the SPLE to Develop Smart Home Resource Management Systems

Jihyun Lee[1] and Sunmyung Hwang[2†]

[1]*College of Liberal Arts, Daejeon University, Daejeon, 300-716 / Republic of Korea*
[2]*Computer Engineering Department, Daejeon University, Daejeon, 300-716 / Republic of Korea*
jihyun30@kaist.ac.kr, sunhwang@dju.kr

Abstract

The software product line (SPL) is an approach that develops a family of similar software by maximizing the reusability of development artifacts. The home resource management system in the smart home system is a core module of the home network middleware system. It manages all of the devices and services installed in the smart home system, i.e., installed devices, their installed location, their status (on/off), and related services. The basic functions of the home resource management system are similar no matter which kinds of homes they are installed in, but the detailed configurations of the location, device types, and services, including service combinations, differ from each other. Therefore, developers should focus on modifying the APIs and resource managers whenever a new kind of house needs to be serviced. To reduce these efforts, the demand for product line architecture (PLA) commonly used among different kinds of smart resource management systems has been raised and this paper describes the design results.

Keywords: *Software product line engineering, Smart home, Home resource management system*

1. Introduction

As devices operating in home environments have become smarter, the services for monitoring and controlling those devices through various network links such as Ethernet and wireless LAN (WLAN) have become diverse. The diverse types of devices, services (embedded or composite), network links, and physical spaces of home environments make it difficult to manage those resources and their relationships.

There have been studies related to smart home resource management. For instance, Yang et al. [1] define smart home architecture based on the resource's name service. This architecture can resolve interoperability problems among the defined devices, even those with diversified protocols, but it cannot resolve the heterogeneity of devices in smart homes. There are also location-aware and context-aware resource management systems in smart homes [2, 3]. Roy et al. [2] predicted the inhabitant's location in order to provide services appropriate to the inhabitant's circumstance, while Liang et al. [3] discovered devices able to provide dynamic services based on the current user context. However, their resource management systems are limited to discovering resources necessary to provide services. They did not consider diverse smart home environments, including heterogeneous resources. Son et al. [4] proposed a resource management system that manages the physical and logical information dynamically. They also tried to manage heterogeneous resources by abstracting resources and their relations. Their resource

[†] Corresponding author

management system separates resource information into common and domain-specific (in other words, variable) information parts. However, the system does not deal with those kinds of information systematically. There has also been research on resource management in a single smart home for multiple inhabitants, namely how to manage resources shared by family members [5, 6].

Diverse resources can be operated in different residence types such as single family homes single or several building apartment complexes, or apartment buildings with stores. There are also diverse space types requiring different services. The existing studies have not considered these heterogeneities in smart home environments. To cope with these aspects, the software product line engineering (SPLE) was recommended as a solution for accepting similarities and singularities without modifying or creating newly developing modules for interoperation [11, 12]. Accordingly, there are studies applying the SPLE to smart home systems [2, 13-15]. However, these studies dealt with the system just as an example. The studies did not consider resource variability in different residence types that are similar but also have unique characteristics. This paper describes an experience of applying the SPLE to home resource management systems in order to resolve heterogeneities in accordance with smart home environments.

The remainder of this paper is organized as follows: In the next section, we describe the background of our research. Section 3 describes the modeling and analysis approaches used for designing product line architecture (PLA), while section 4 shows the application results of the SPLE to home resource management systems. In Section 5, we describe the evaluation results. Finally, in Section 6, we conclude this paper.

2. The SPLE and Smart Home Context

2.1. Family of Home Resource Management System

Home resource management systems in smart home environments manage resources at the level of a single house, a building, or several apartment complexes. The home resources include devices — home electronic and network appliances, home gateways, and building servers — and services such as application services, services embedded in devices, and composite services. In a home resource management system, the personal users or managers of a building or an apartment complex monitor the status of diverse home devices and services through the common interface of its middleware. Moreover, the managers of a building or an apartment complex ensure the safety and security of the devices, network links, and services and should be able to control them to detect and recover from faults.

Depending on the location where home resource management products are deployed and used, the home resource management system product line consists of the following four member products [16]:

- *Monitoring Application (HRM4Mo)*: This is the simplest product for home resource management. It includes only basic features for monitoring home resources and managing the relations among them.
- *Fault Handling Application (HRM4FH)*: With the basic features of HRM4Mo, this product has additional features for diagnosing faults and recovering from those faults.
- *Building Management Application (HRM4Bui)*: This product is installed in a single building server, so its resources under managerial support are common facilities operating in a single building.
- *Complex Management Application (HRM4Cmx)*: This product manages the resources of an apartment complex such as the common facilities in a complex, the network equipment, and the information and services of the apartment complex.

Member products within a home resource management product line are chosen in accordance with their deployed locations and equipped functionalities. Figure 1 illustrates an example in which HRM4Cmx is deployed on both a complex server and a building server such that they are connected through the Ethernet. HRM4Cmx is connected with several HRM4Bui products in a building server, and HRM4Bui is connected with several HRM4Mo or HRM4FH products deployed at home servers through RS485 link. In regards to managing home devices through a smart phone, connecting through WLAN should be supported. HRM4Mo and HRM4FH manage diverse devices and embedded/composite services at home.

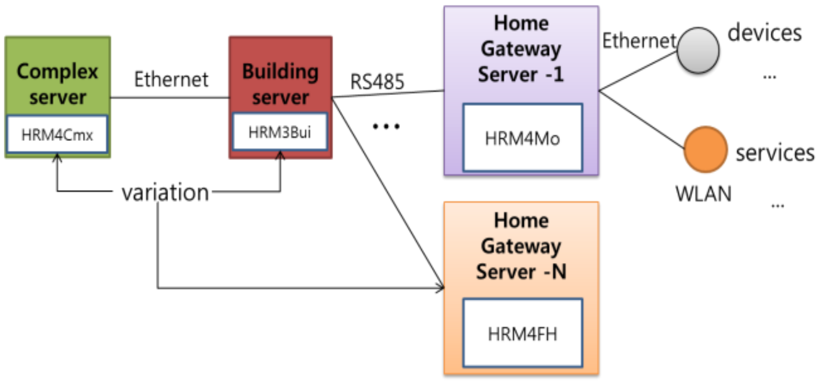


Figure 1. Family of Home Resource Management Systems

2.2. Variability in Home Resource Management System

The deployment locations for families of home resource management systems that correspond to the context of the smart home system are presented in Figure 1. As the deployment locations differ, the types of resources are also quite different. For example, a home resource management system installed into complex server new resource types such as unmanned delivery devices and parking lot security among others are added. However, even though the types of resources are the same (e.g., heating and cooling facilities), their deployed spaces and relations for controlling their behaviors differ. In order to determine proper variability implementation mechanisms [7], sources for variation are analyzed from the following aspects [8]:

- *Variation in function*: A particular function may exist in some products and not in others.
- *Variation in control flow*: A particular pattern of interaction may vary from one product to another.
- *Variation in data*: A particular data structure may vary from one product to another.
- *Variation in technology*: The technology used such as OS, user interface, hardware, and middleware may vary in exactly the same fashion as the function.

In the case of home resource management systems, most variations occur in data and technology. In particular, the most significant variability occurs in data structures related to spaces, the devices installed, and the services provided. There are also variations in function between member products (especially HRM4Mo and HRM4FH) within a product line. For example, the fault handling application requires an additional monitoring function for diagnosing and recovering faults. A complex management application has additional functions for monitoring network equipment or handling its faults. Because of those variations, there are also many variations in the user interface as well as in architectural components.

3. Modeling and Analysis Techniques

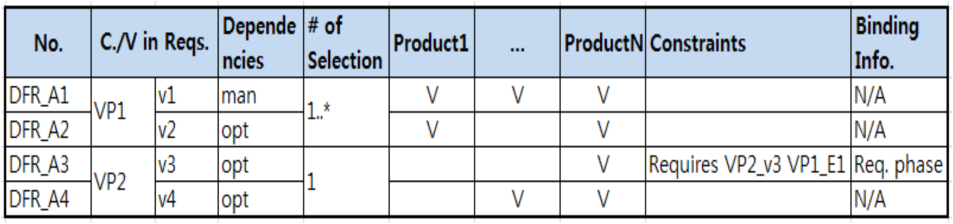
3.1 Variability Modeling Approach

The Orthogonal Variability Model (OVM) approach separates the variability model from other development artifacts as it only deals with variability, as shown at the top of Figure 3. The OVM approach uses consistent variability modeling notations throughout the different development phases, and it provides an easy way to maintain end-to-end SPL traceability among several development artifacts produced within the same phase and inter phases [19]. The feature model that models commonality and variability at the same time [20] has several strengths in terms of expressiveness and understandability, while practitioners have pointed out its weakness such as scalability, traceability with development artifacts [21], and ambiguity in variability definition [12]. The OVM approach overcomes these problems of the feature model. Furthermore, the OVM approach makes it possible to manage variability in a group [12], while the feature model suffers from a lack of feature grouping mechanism.

While the OVM approach is good for variability modeling, it suffers from a lack of capability to analyze variability efficiently and to describe variability with relevant information such as binding time, constraints adhered for the right binding, and rationales for decisions. To tackle this, a tabular format called the variability analysis table was used to analyze variation points, variants, binding information, and dependency constraints. The variability analysis table is similar to the decision model approach proposed by Schmid et al. [17] However, it differs from the decision model approach in that the variability analysis table includes commonality together with variability because its purpose is to analyze and describe all of the relevant variability information.

Figure 2 shows a variability analysis table in the requirements phase. The variability analysis table can be applied throughout all of the development phases together with the orthogonal variability model. In the variability analysis table, all requirements are listed and classified to identify variation points. In the first column of the table, which is called "No.", the ID for each requirement is represented. Then, the name of each requirement is followed in the column called "C/V in Req.". The dependencies and the number of selections denoting variability types such as exclusive-or, inclusive-or, or optional types follow in the third and fourth column. The next columns Product1 through ProductN define the selection of each requirement per product. The constraints among the requirements are listed, and in the last column, the binding information is described. In the binding information column, N/A means binding is not conducted at this phase.

The variability analysis table is completed easily by using spreadsheets like Microsoft Excel. This brings the following advantages: First, it describes clearly a variation point and its variants. Secondly, it is easier to describe binding information than the graphical variability model (The graphical variability model should use much more spaces than tabular formats.) Thirdly, it shows good usability due to its familiar tabular formats.



No.	C./V in Reqs.		Dependencies	# of Selection	Product1	...	ProductN	Constraints	Binding Info.
DFR_A1	VP1	v1	man	1..*	V	V	V		N/A
DFR_A2		v2	opt		V		V		N/A
DFR_A3	VP2	v3	opt	1			V	Requires VP2_v3 VP1_E1	Req. phase
DFR_A4		v4	opt			V	V		N/A

Figure 2. Structure of Variability Analysis Table

3.2. Variability Tracing Approach

The variability of a product line is spread all over the development artifacts because variability is implemented by several different artifacts. In the variability model, the variation points and their variants derived from a certain development phase have a relationship with the relevant development artifacts, either in the same development phase or the different development phases. Thus, there exist multi-to-multi relations between the variability and development artifacts. This makes it difficult to trace the variability realized in the development artifacts.

The OVM approach provides a cross-sectional view of variability across all development artifacts. This approach can relate the elements of variability to different development artifacts such as feature models, requirement artifacts, architecture, detailed designs, codes, test artifacts, and after compile time artifacts (e.g., makefile). Therefore, this paper uses the OVM-based variability tracing approach to trace the variability with the development artifacts. This approach is conceptually defined by Pohl et al. [12]. Figure 3 shows the conceptual view of the OVM based variability tracing approach. From the initial variability defined in the feature level through the variability in architecture, they are traced with its consistent naming in OVM. Also, the relationships from variability to the development artifacts are traced from OVM to the artifacts produced in each domain engineering phase.

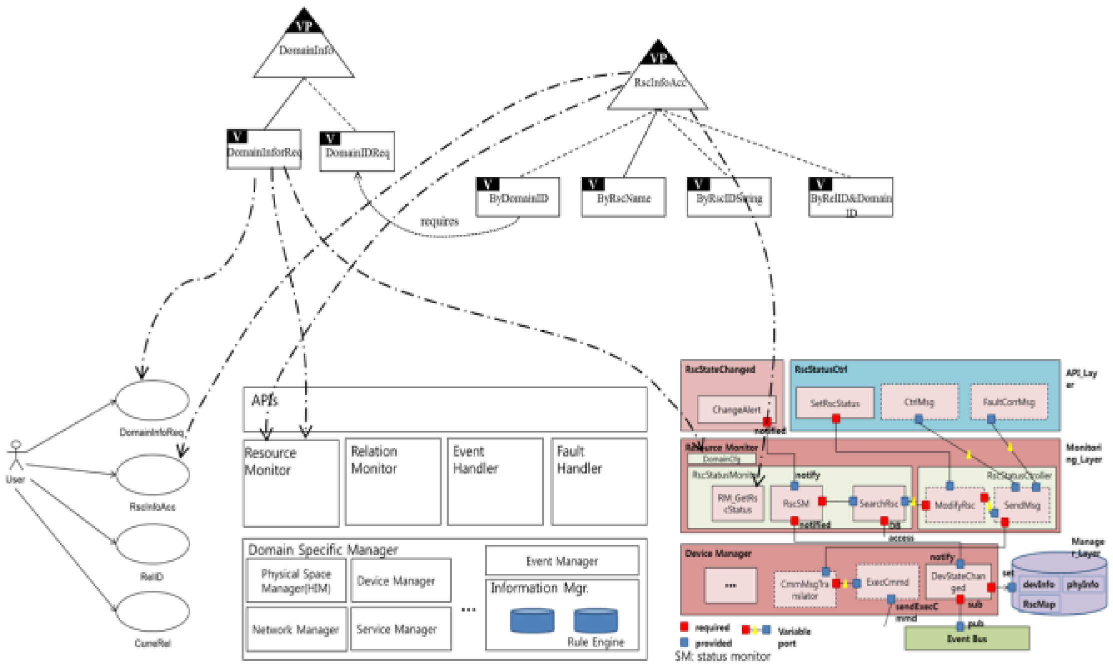
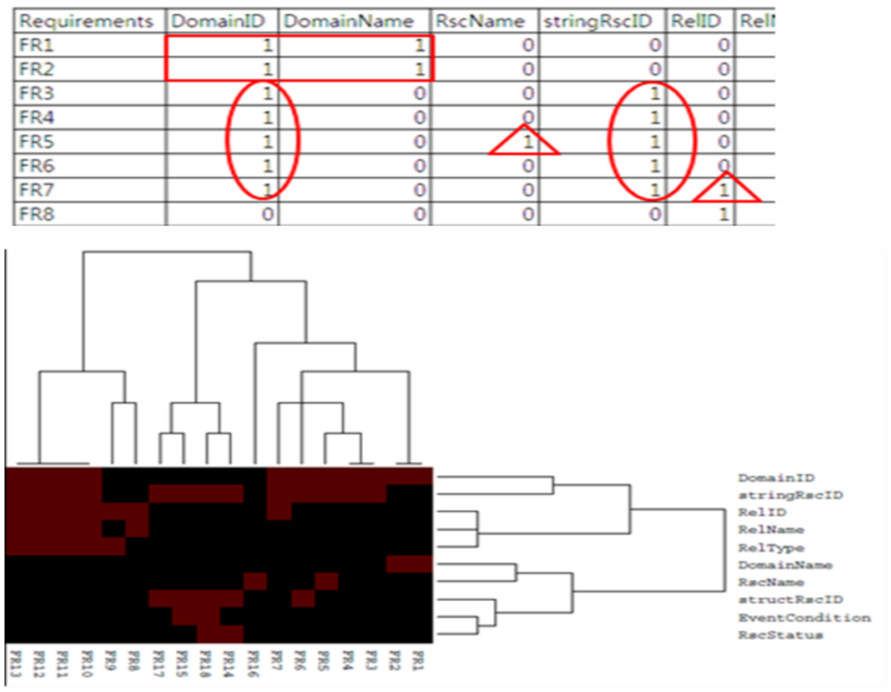


Figure 3. OVM based Variability Tracing

3.3. Clustering using Similarity Analysis

The product line should analyze and manage variation points, variants, and their relationships effectively. Most importantly, deciding a variation point and its relevant variants is important. Moreover, a single variability can be implemented through several components, so clustering similar requirements and derived requirements in design (and thereafter designing them in the form of the same subsystem) is significant [18]. In order to do this, this paper measures similarities among textual requirements by using the Euclidean distance clustering technique [9, 10], and then establishing requirement clusters. We assigned names to requirements so that the domain engineers intrinsically knew each requirement for enhancing understandability and organizing them in the form of requirements and sub-requirements.

To first organize the requirements and sub-requirements, we analyzed the input/output parameters of each requirement so as to define the attributes used for analyzing similarity. In Figure 4, the header row means the analyzed similarity evaluating attributes, and the value "1" means relevancy between the requirements and the attribute while "0" represents the irrelevancy between them. Each of the FR1 and FR2, FR3 and FR4, FR6 and FR7 in Figure 4 are highly similar requirements. From FR3 through FR7, there are similar requirements in terms of "DomainID" and "stringRscID". Thus, those requirements can be grouped as one representative requirement and its sub-requirements. For example, FR1 and FR2 can be grouped as a requirement called "DomainInfo" and can be assigned to two relevant sub-requirements called "DomainInfoReq" and "DomainIDReq", respectively.



Requirements	DomainID	DomainName	RscName	stringRscID	RelID	Rel
FR1	1	1	0	0	0	
FR2	1	1	0	0	0	
FR3	1	0	0	1	0	
FR4	1	0	0	1	0	
FR5	1	0	1	1	0	
FR6	1	0	0	1	0	
FR7	1	0	0	1	1	
FR8	0	0	0	0	1	

Figure 4. FR-attribute Matrix and Clustering Results

3.4. Data Modeling

The home resource management system should manage several types of devices, the physical spaces of a residential environment, network links, resource status, and the complex relations among these. These are managed through the database system. Thus, entity modeling is essential for managing resources installed in a home and their relations. In addition, in the smart home system, regardless of the types of homes where it is installed, there are relations among the resources for providing application services.

The home resource management system can be installed into diverse home types, i.e., a building or an apartment complex. Because resources composed of domains differ from home types, there exist variations in data modeling entities. Moreover, variations in the relations among resources due to variations in resources make the resources difficult to detect and maintain. To tackle this, we designed a relationship structure of a home resource management system in the hierarchical order of physical spaces, network, device, and its embedded service. All resources in a smart home should be connected through network ports and all network ports with unique identifiers should be built in physical spaces. Thus, using physical spaces that rarely change as a basis is reasonable rather than using those that have a tendency to change like devices and networks. Figure 5 shows the relation structure satisfying these aspects. Entity modeling is conducted based on this relation structure.

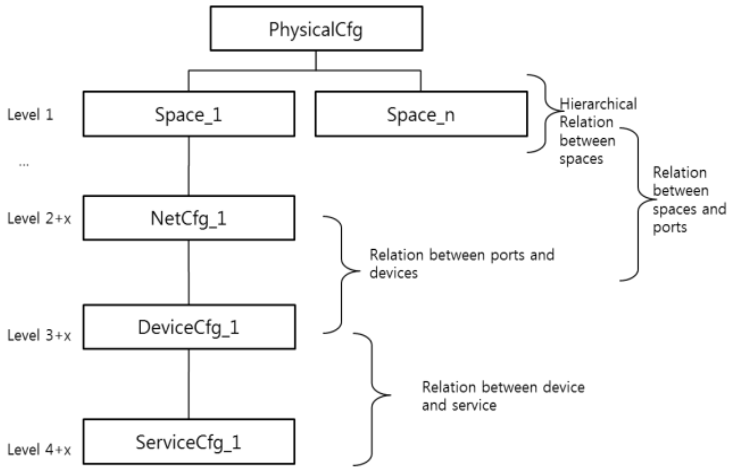

PhysicalCfg
Level 1
Space_1
Space_n
...
Hierarchical Relation between spaces
Relation between spaces and ports
Level 2+x
NetCfg_1
Relation between ports and devices
Level 3+x
DeviceCfg_1
Relation between device and service
Level 4+x
ServiceCfg_1

Figure 5. Relation View among Resources

However, the types of resources, attributes, and relations among these to be managed by a home resource management system differ from the residence types. This means that there is entity data variability in the resource types, attributes, and relations among these. Accordingly, this paper defines the following three types of entity variability for entity modeling [26]:

- Positive variability: adding new field, entity, or relation
- Negative variability: removing field, entity, or relation
- Structural variability: variable type, cardinality, or element name

The “DeviceID”, “RelationInfo”, and “RelatedDomain” in Figure 6 are positive variability, adding new fields if they are included in “DeviceCfg”, or negative variability if they are not. The association relation between the “DeviceCfg” and “EmbService” is an example of structural variability.

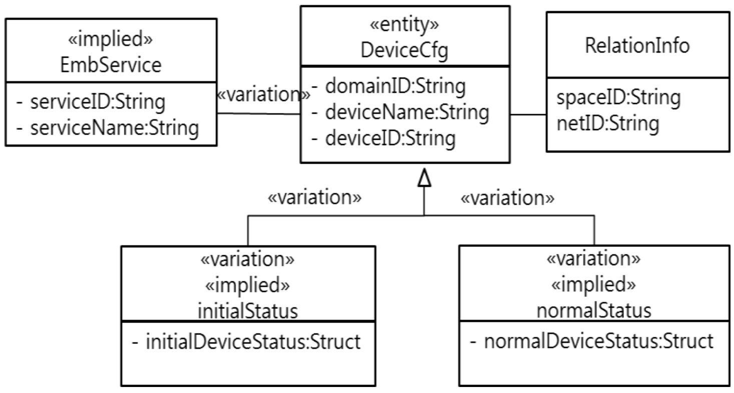


Figure 6. Entity Model for Device

In the case of the HRM4Mo product, no status information is recorded. However, the HRM4Mo product might include a function that returns the status of devices to their initial setting. In the case of HRM4FH, either one of the functions among those returning the status to the initial setting, or returning the status to before the abnormal status, or both of them, should be selected. Those entity variabilities can be defined into two variations: “initalStatus” and “normalStatus” entities, as shown in Figure 6.

4. Applying SPL to a Home Resource Management System

This section describes a few representative design results in accordance with the SPLE life cycle phases from the requirements through design, including data modeling.

4.1. Variability in Requirements

We defined the members of a home resource management system product line after analyzing residential environments where the home resource management system can be installed and operated. The defined members of a product line consist of those described in sub-section 2.1. Eighteen requirements were analyzed and below are a few representatives:

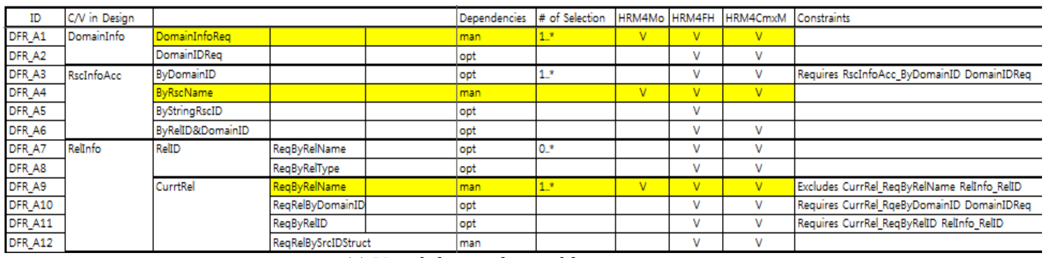
- *FR1(DomainInfoReq)*: Requests domain information by domain ID and then prints the domain names with their grouped resources.
- *FR2(DomainIDReq)*: Requests domain ID assigned to a specific domain name.
- …
- *FR12(CurrtRelReqRelByDomainID)*: Requests current relation information of a specific domain ID.
- *FR13(CurrtRelReqRelBySrcIDStruct)*: Requests relation information of a resource structure ID.
- *FR14(RscStatusCtl)*: Transfers control command to the resource object (ID) of a resource structure ID.

After eliciting the requirements, clustering using similarity analysis was performed. The following five clusters were analyzed by applying a similarity analysis approach in accordance with the dendogram of Figure 4:

- C1 = {FR1, FR2}
- C2 = {FR3, FR4, FR5, FR6, FR7}
- C3 = {FR8, FR9} {FR10, FR11, FR12, FR13}
- C4 = {FR15, FR17} {FR14, FR18}
- C5 = {FR16}

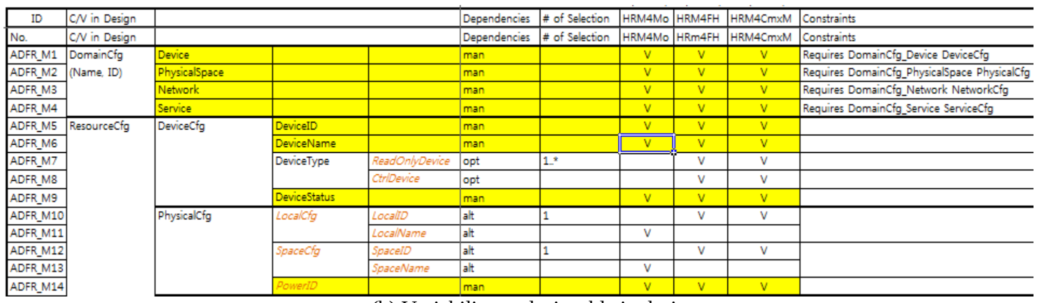
Figure 7 shows the results of grouping and classifying requirements and their sub-requirements according to the clustering results. The requirements-member product matrix was used to analyze common and variable requirements. Requirements that are required for all members were initially classified as mandatory requirements and others were classified as variable requirements. After that, the variability dependency constraints such as optional or alternative dependencies and required or excluded dependencies were determined together with the number of selections. For example, in the case of the "RscInfoAccByDomainID", the "domainID" is needed to search for relevant resources. However, when we do not know the "domainID" of a resource that we are looking for, a proper way should be provided. For this, the "DomainIDReq" of the DomainInfo" requirement is necessary. Therefore, two requirements, "RscInfoAccByDomainID" and the "DomainIDReq" of the "DomainInfo" are in the "required" dependency.

The variability model that deals with variability only is defined orthogonally. In our experience, the feature model is used only for supporting the domain requirements' elicitation and analysis. The features analyzed in the requirements phase are traced with relevant requirements, artifacts, and architectural elements, but the variability of a home resource management product line is managed through the orthogonal variability models. This is to avoid confusion due to feature models that manage variability together with commonality. Figure 7(a) shows part of the variability introduced in the requirements phase. This variability is refined or newly added as the process goes forward.



ID	C/V in Design				Dependencies	# of Selection	HRM4Mo	HRM4FH	HRM4CmxM	Constraints
DFR_A1	DomainInfo	DomainInfoReq			man	1..*	V	V	V	
DFR_A2		DomainIDReq			opt			V	V	
DFR_A3	RscInfoAcc	ByDomainID			opt	1..*		V	V	Requires RscInfoAcc_ByDomainID DomainIDReq
DFR_A4		ByRscName			man		V	V	V	
DFR_A5		ByStringRscID			opt			V		
DFR_A6		ByRelID&DomainID			opt			V	V	
DFR_A7	RelInfo	RelID	ReqByRelName		opt	0..*		V	V	
DFR_A8			ReqByRelType		opt			V	V	
DFR_A9		CurrtRel	ReqByRelName		man	1..*	V	V	V	Excludes CurrRel_ReqByRelName RelInfo_RelID
DFR_A10			ReqRelByDomainID		opt			V	V	Requires CurrRel_RqeByDomainID DomainIDReq
DFR_A11			ReqByRelID		opt			V	V	Requires CurrRel_ReqByRelID RelInfo_RelID
DFR_A12			ReqRelBySrcIDStruct		man			V	V	

(a) Variability analysis table in requirements



ID	C/V in Design				Dependencies	# of Selection	HRM4Mo	HRM4FH	HRM4CmxM	Constraints
No.	C/V in Design				Dependencies	# of Selection	HRM4Mo	HRm4FH	HRM4CmxM	Constraints
ADFR_M1	DomainCfg	Device			man		V	V	V	Requires DomainCfg_Device DeviceCfg
ADFR_M2	(Name, ID)	PhysicalSpace			man		V	V	V	Requires DomainCfg_PhysicalSpace PhysicalCfg
ADFR_M3		Network			man		V	V	V	Requires DomainCfg_Network NetworkCfg
ADFR_M4		Service			man		V	V	V	Requires DomainCfg_Service ServiceCfg
ADFR_M5	ResourceCfg	DeviceCfg	DeviceID		man		V	V	V	
ADFR_M6			DeviceName		man		V	V	V	
ADFR_M7			DeviceType	*ReadOnlyDevice*	opt	1..*		V	V	
ADFR_M8				*CtrlDevice*	opt			V	V	
ADFR_M9			DeviceStatus		man		V	V	V	
ADFR_M10		PhysicalCfg	*LocalCfg*	*LocalID*	alt	1		V	V	
ADFR_M11				*LocalName*	alt		V			
ADFR_M12			*SpaceCfg*	*SpaceID*	alt	1		V	V	
ADFR_M13				*SpaceName*	alt		V			
ADFR_M14			*PowerID*		man		V	V	V	

(b) Variability analysis table in design

Figure 7. Commonality and Variability Analysis Matrix

Figure 8 shows the defined variability model in requirements phase described orthogonally.

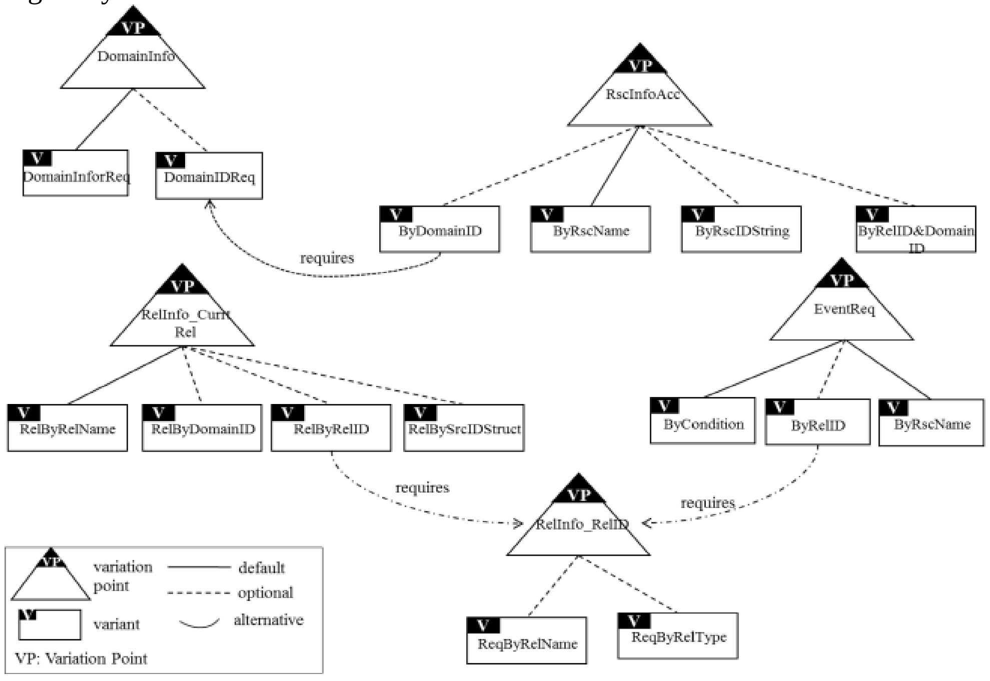


Figure 8. Variability Model in Requirements Phase

4.2. Variability in design[‡]

Variability sources were analyzed for architectural decisions based on the variability sources mentioned in subsection 2.2. The analysis results provide the rationale for deciding variability implementation mechanisms such as plug-in, parameterization, design pattern, and overloading. Table 1 describes the classified results by variability sources.

[‡] Notations used in PLA design refer to H. Gomma[25]

"PhysicalSpaceCfg", "ServiceCfg", "BindingRel", and "EventType" were analyzed as new variability.

Table 1. Classification by Variability Sources

Variation in function	Variation in data	Variation in control flow	Variation in technology
• RscStatusCtl • DomainInfo • EventHandling • ServiceCfg	• DevicdCfg • NetworkCfg • PhysicalSpaceCfg • BindingRel • EventType	• RscInfoAcc • RelIDReq • CurrtRelReq • RscStactusAcc • EventReg • …	• User interface

We classified information that will be managed and maintained by home resource management systems such as "Domain", "Resource", "Relation", and "Event" based on the clustering results and conceptual relations between smart home environments and home resource management systems. Figure 7(b) shows the identified variability in the design phase. Under an environment where users do not need to know the detailed resource operations, interacting directly with each resource manager increases complexity [22].

Accessing resources through integrated resources with a single common view can decrease complexity rather than accessing resources in accordance with resource types such as device, service, and physical space. Therefore, we designed a PLA for home resource management system by using layered architecture. This PLA has four layers: 1) Application layer interfacing with users, 2) Core monitor layer monitoring resource types and relations managed in home resource management system, 3) Core manager layer detecting installed home devices, service installation, networks, and physical spaces, as well as distinguishing/ diagnosing each of the resources, and 4) Infra-structure layer. Figure 9 is a conceptual PLA for a home resource management system. The infra-structure layer including the database and operating systems is omitted in Figure 9.

The architecture style using sub-systems and layers allows architects to group similar classes. The whole member product within a product line should use the defined layered architecture style and conceptual structure [24]. Figure 10 shows one level decomposition results for the monitor layer, and Figure 11 shows the parts of the component diagram for resource status monitor and control functions.

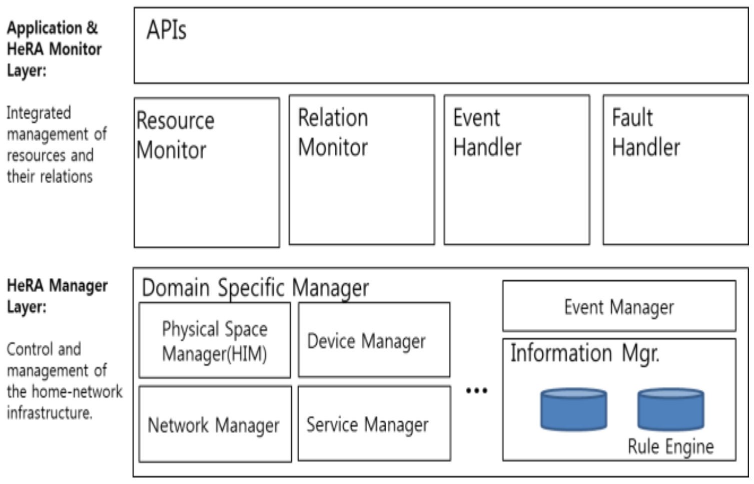


Figure 9. Conceptual PLA for Home Resource Management System

Unlike the conceptual PLA shown in Figure 9, one layer decomposition results reveal variability in more detail. In Figure 9, the “Fault Handler” subsystem is described as a common part, but in Figure 10, the subsystem includes a variable component, “FaultInference” and “FaultDetection” (the rectangle with a dashed line in Figure 10 means variable components).

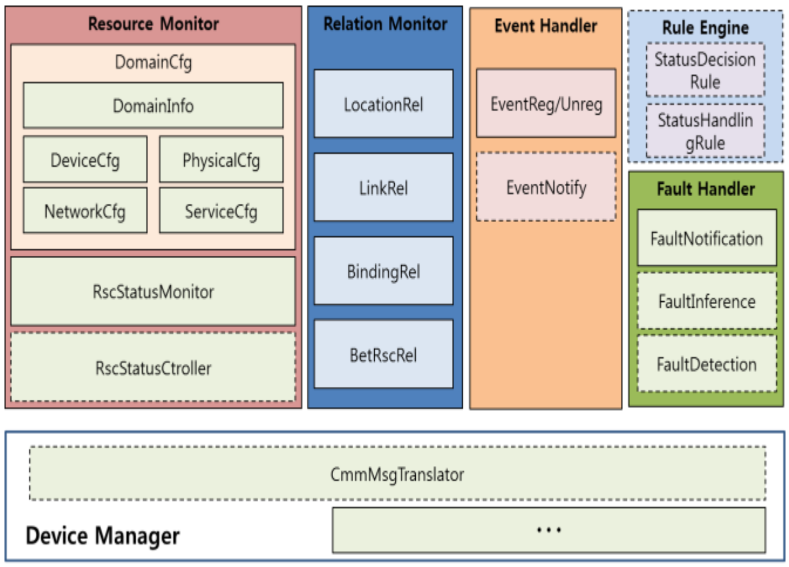


Figure 10. Decomposition for Monitor Layer

The subsystem decomposed in more detail in the form of variability in ports and connectors is shown in Figure 11.

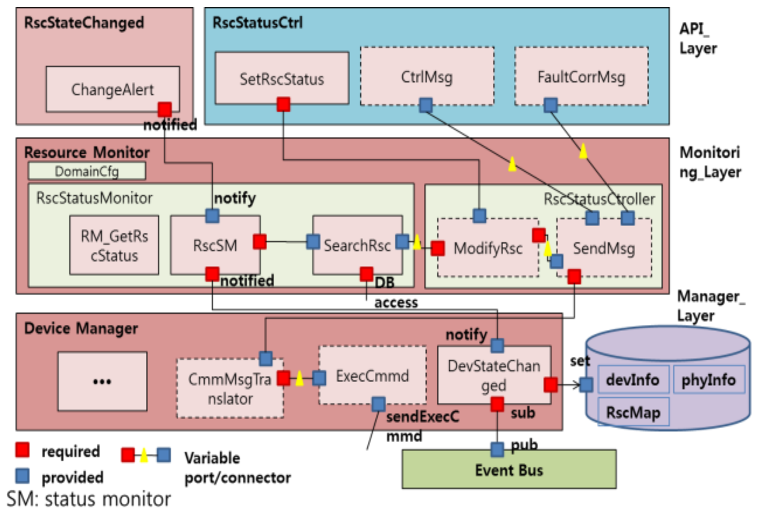


Figure 11. Component Diagram for “RscStatus” Monitor and Control Components

4.3. Variability Modeling in Resources

There exists variability in the physical space, networks, and devices in a home resource management system. In the case of services divided into an embedded service and composite service, the existence of an embedded service is decided in accordance with the existence of the specific device. Even the composite service can be served by interacting with different devices. Thus, this paper assumes there is no service variability for reducing the complexity of problem processing. Figure 12 is an entity model for physical space, including HouseholdID and OutsideSpaceType, which are positive variability types. Physical space has a hierarchical structure of address, including building and unit number, room, and wall/window (Level 1 and Level 2+x in Figure 5). The SpaceCom entity has a parent space field (named parentSpace in Figure 11). According to the types of residential environment, space type composed of space differs and the different management methods can put different IDs and names.

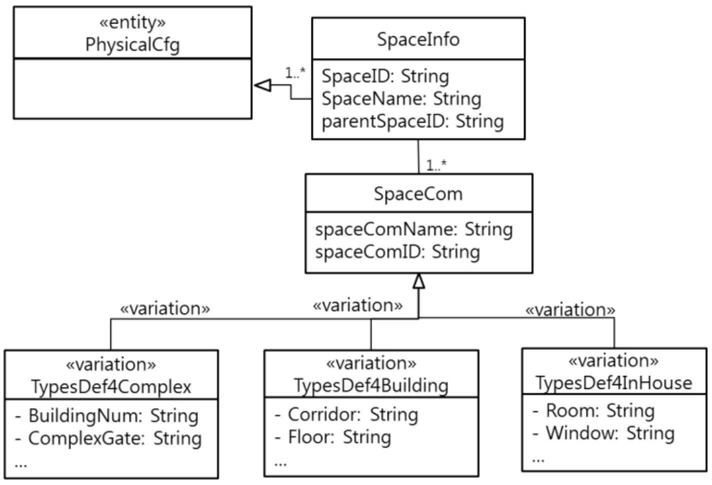


Fig. 12 Entity model for physical space

In the case of composite service, its relation structure can be complex, as that shown in Figure 13, because the composite service provides services by using other devices and services. The relation structure of the composite service is composed of devices and their embedded services (see right side of Figure 13).

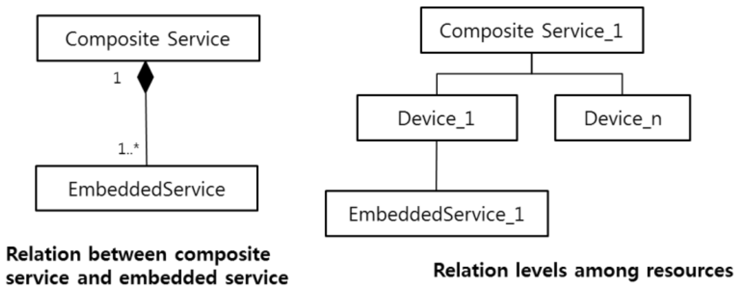


Fig. 13 Relation in composite services

5. Evaluation by comparison

This section evaluates the proposed results by comparing the research results of Son et al. [4] and Son et al. [23], who reported the home resource management system research results and of which the target system was the same. The comparison was conducted from various supports in architecture design and that in data model aspects. This is because the reusability and flexibility of the smart home resource management system depend on whether the data model can vary and whether the data variation and system modules using the varied data can be modified easily.

5.1. Variation support in data modeling

Most variations in SPL occur in functional and non-functional differences. In the smart home resource management system, the remote fault handling function can be removed in low-end products in order to reduce the prices of products. As shown in Figure 6, the “initialStatus” and “normalStatus” entities, which are significant information for fault recovery in fault handing, are defined as variations. In the case when an application that does not use fault recovery functions, it cannot select the “initalStatus” and “normalStatus” entity when an application derives its data model. Or an application can select one of the data entities among the “initalStatus” and “normalStatus” related to fault recovery.

In a similar way, the data modeling for physical space in Figure 12 is designed by reflecting the differences according to the location where the home resource management systems are installed and operated. Generally, the smart home resource management systems are not installed and operated within a household. In the case of a home resource management system installed in apartment complex or single building, the types of space are quite different from those installed in a household, and device types can be newly added. However, Son et al. [4, 23] define physical space types only for home resource management systems that will be installed and operated in a household. Therefore, the home resource management system implemented by Son et al. [4, 23] is not easy to modify or evolve for the other types of smart home systems, except for a single household.

5.2. Variation support in architecture design

This section describes the ways that PLA for home resource management systems using or processing data under variation deals with variability. In accordance with the selection in the fault handling relevant variation in the data model (1), in the case when the fault handling function is not selected, the “FaultHandler” subsystem in Figure 9 is removed from the application architecture. (2) If an application data model selects one of the fault recovery entities among the “initalStatus” and “normalStatus”, the relevant “FaultCorrMsg’” component in Figure 10 can be implemented as follows:

```
typedef struct faultCorrMsg
{
     int deviceStatusValue;
     int serviceStatusValue;
     int netStatusValue;
};

 #ifdef initialStatus
          faultCorrMsg = initialDeviceStatus;
#else
          faultCorrMsg =normalDeviceStatus;
#endif
```

The “RscMap” module in the “Information Manager” subsystem described partly in Figure 9 and Figure 11 varies in accordance with the variation of physical space. According to Son el al.[4], the resource relation map has the form of “(srcID, relation type, list of target ID)”, and the source and target can be device, location, or network link, and the relation type can be location and connection. The space in physical space defined in this paper means the resources of which the sources are devices and relation types are location. According to Figure 5 and Figure 12, the entity model whose entity name is “PhysicalCfg” describes the resources used by this location type. “PhysicalCfg” is designed in the form that the binding of the space type definition is possible in accordance with the environments where the home resource management system is installed and operated. The resource map proposed by Son et al. [4] can be considered as the map for one resource management system product.

6. Conclusions

This paper described the experience of applying SPLE for designing a family of home resource management systems for integrated home resource management. The PLA for the family of home resource management systems was designed based on the planned resource management systems. In the home resource management system PL, there was lots of variability related to data due to the differences of resource types other than functional variability. Three types of resource relevant data variability were managed with hierarchical configuration. In the case of services, one of those resource types was divided into two types: single and composite. Services can be served only when the specific devices exist, so service variability depends on devices. For this reason, we dealt with variability in resources, except for service variability.

Another issue in our experience was how to describe variability in the product line architecture with their information: variability types, dependencies, and variability implementation mechanisms. Variability models separated with design artifacts describe variability types and dependencies, but the variability implementation mechanisms determined in each product line process have not been described. Therefore, we decided on variability mechanisms with architecturally relevant variability identification and added them to variability models or texturally described variability.

The results were provided to the smart home research project in order to validate why applying SPLE is important for smart home systems. Parts of this research, especially managing common and variable resource information, were needed to develop a smart home system.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2013R1A1A3005162).

References

[1] C. Yang, B. Yuan, Y. Tian, Z. Feng, and W. Mao, A Smart Home Architecture Based on Resource Name Service, Proceedings of the IEEE 17th International Conference on Computational Science and Engineering, (**2014**), pp. 1915-1920.

[2] A. Roy, S.K. Das, and K. Basu, A Predictive Framework for Location-Aware Resource Management in Smart Homes, IEEE Transactions on Mobile Computing, vol. 6, no. 11, (**2007**) November, pp. 1270-1283.

[3] Y. Liang, X. Zhou, Z. Yu, H. Wang, and B. Guo, A Context-Aware Resource Management Framework for Smart Homes, Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications, (**2010**), pp. 1-8.

[4] J. Y. Son, J. H. Park, K. D. Moon, and Y. H. Lee, "Resource-Aware Smart Home Management System by Constructing Resource Relation Graph, IEEE Transactions on Consumer Electronics", vol. 57, no. 4, (**2011**), pp. 1112-1119.

[5] C.-C. Hsu and L.-Z. Wang, "A Smart Home Resource Management System for Multiple inhabitants by Agent Conceding Negotiation", Proceedings of IEEE International Conference on Systems, Man and Cybernetics, (**2008**), pp. 607-612.

[6] S. Bhardwaj, T. Ozcelebi, J. Lukkien, and C. Uysal, "Resource and Service Management Architecture of a Low Capacity Network for Smart Spaces", IEEE Transactions on Consumer Electronics, vol. 58, no. 2, (**2012**), pp. 389-396.

[7] J. Lee and S. Hwang, "A Review on Variability Mechanisms in SPL", International Journal of Advanced Media and Communication, vol. 2/3, (**2014**), pp. 172-181.

[8] F. Bachmann and L. Bass, "Managing Variability in Software Architectures", ACM SIGSOFT Soft. Eng. Notes, vol. 26, (**2001**), pp. 126-132.

[9] P. Andritsos and V. Tzerpos, "Information-theoretic Software Clustering", IEEE Transactions On Software Engineering, vol. 31, no. 2, (**2005**) February, pp. 150-165.

[10] N. Niu and S. Easterbrook, "On-Demand Cluster Analysis for Product Line Functional Requirements", Proceedings of Software Product Line Conference, (**2008**), pp. 87-96.

[11] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley (**2002**).

[12] K. Pohl, G. Böckle, and F. van der Linden, "Software Product Line Engineering: Foundations", Principles, and Techniques, Springer (**2005**).

[13] C. M. Jaeger and L. Fiege, "Viewpoints in the Smart Home Product Line, Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems", (**2009**) pp. 332-339.

[14] L. Baresi, S. Guinea, and L. Pasquale, "Service-Oriented Dynamic Software Product Lines, Computer, vol. 45, no. 10, (**2012**), pp. 42-48.

[15] J. C. Royer and H. Arboleda, “Model-Driven and Software Product Line Engineering”, Wiley (**2013**).
[16] H. Ahn, S. Kang and J. Lee, “A Case Study Comparison of Variability Representation Mechanisms with the HeRA Product Line”, Proceedings of the International Conference on Computer and Information Technology, (**2013**) December.
[17] K. Schmid, R. Rabiser, and P. Grünbacher, A Comparison of Decision Modeling Approaches in Product Lines, Proceedings of the VaMos’11, (**2011**).
[18] E. Hull, K. Jackson, and J. Dick, “Requirements Engineering”, 2nd Ed., Springer (**2005**).
[19] J. G. Kim, S. W. Kang, and J. H. Lee, “A Comparison of Software Product Line Traceability Approaches from End-to-End Traceability Perspectives”, International Journal of Software Engineering and Knowledge Engineering, vol. 24, no. 4, (**2014**), pp. 677-714.
[20] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, SEI Technical Report (**1990**).
[21] K. C. Kang, “FODA: Twenty Years of Perspective on Feature Models”, Proceedings of the 13th International Software Product Line Conference, (**2009**), pp. 1–61.
[22] L. Bass, P. Clements, and R. Kazman, “Software Architecture in Practice”, 2nd Ed., Addison Wesley (**2003**).
[23] J. Y. Son, J. H. Lee, J. Y. Kim, J. H. Park, and Y. H. Lee, “RAFD: Resource-Aware Fault Diagnosis System for Home Environment with Smart Devices”, IEEE Transactions on Consumer Electronics, vol. 58, no. 4, (**2012**) November , pp. 1185-1193.
[24] S. Agarwal and D. Tomar, “A Feature Selection Based Model for Software Defect Prediction”, International Journal of Advanced Science and Technology, vol. 65, (**2014**) April, pp. 39-58.
[25] H. Gomaa, “Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures”, Addison Wesley (**2004**).
[26] J. Bartholdt, R. Oberhauser, and A. Rytina, “An Approach to Addressing Entity Model Variability within Software Product Lines”, Proceedings of the 3rd International Conference on Software Engineering Advances (**2008**), pp. 465-471.