Experiences with Software Product Line for Developing Middleware Interworking with Diverse Devices

Jihyun Lee and Sunmyung Hwang¹

¹ Daejeon University, Daejeon, 300-716, Republic of Korea
{jihyun30, sunhwang}@dju.kr

Abstract. Software product line (SPL) is an approach that develops the family of similar software by maximizing the reusability of development artifacts. As devices have become smarter and diverse the role of middleware for managing such devices has become important and complicated in smart environments such as smart ship, smart home, and smart factory. In this kind of middleware interworking with various devices for collecting their status values and controlling them is a core component. Its implementation is similar but a little different according to interworking devices. This paper describes experiences of applying SPL to develop such middleware systems that manage devices operated in smart ship and smart home. As the results common issues were analyzed and experimental design with recommended resolutions was conducted for their future usage in other kinds of smart environments.

Keywords: software product line, product line architecture, interworking function

1 Introduction

As devices have become smarter and diverse the roles of middleware for managing such devices have become important and complicated. Such devices include embedded services and those services can be solely operated, but in most cases they interact with other services for providing specific services, namely composite services. Middleware in such smart environments manages all of the devices and services installed, i.e., installed devices, their installed location, their status (on/off), and related services. Even the same smart environments such as smart home resource management system and ship maintenance system the basic functions of middleware are similar no matter where they are installed, but the detailed configurations of device types, messages for monitoring or controlling them, and services including service combinations, differ from each other.

For example, in the case of ship maintenance system in terms of requesting log files, voice recording system (VRS) requests the most recently created log file while alarm monitoring and control system (AMS) requests by the name of file or duration of creation. And in the case of home resource management system the services for monitoring and controlling home devices use different network links such as Ethernet, wireless LAN (WLAN), and RS485[1] in accordance with locations where they are
installed. These various 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. In these kinds of middleware it is almost impossible to develop one-size-fits-all middleware. To cope with these aspects SPL was recommended as a solution for accepting both similarities and singularities without modifying or newly developing modules for interoperation. Software product line (SPL) is an approach that develops the family of similar software by maximizing the reusability of development artifacts while assuring customization for each family member[2]. This paper describes two experiences of applying SPL approach for middleware design. The remainder of this paper is organized as follows: In the next section, we describe a family of middleware systems in two smart environments. Section 3 describes lessons learned together with the part of middleware design results and lastly we conclude our experiences.

2 Family of middleware systems in smart environments

As the growth of the number of smart devices within a ship the value-added ship services have been built. Though the rate of failure or fault of those devices within a ship is low, the impacts of the failure are tremendous so a manufacturer cannot help investing much budget for equipping the maintenance system. Therefore, the necessities of detecting the status changes of devices and services to recover from the crash/performance-degradation caused by faults are raised. Moreover, a device operator or manager of a ship wants functions easy to control the devices by connecting them through networks. Accordingly, the needs on middleware interacting with resources, i.e. devices, services, and networks are on the increase. However, various kinds of systems are operating in a ship such as engine monitoring system (EMS), VRS, AMS, obstacle management system (OMS), and so on. Each system has many network-connected devices and their operations are similar but they also have unique characteristics. As shown in the left side of Figure 1 different tailored middleware systems should be developed.

![Family of middleware systems](image)

**Fig. 1.** Family of middleware systems

The home resources include many devices — home electronic and network appliances, home gateways, and so on. In a smart home middleware system, the personal users or managers of a single home or 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 should ensure the safety and security of the devices, network links, and services, and they also should be able to control them to detect and recover from faults. Middleware for supporting interactions between operators and home resources within a smart home can manage resources at the level of a single house, a building, or several apartment complexes. It is a little different in accordance with their deployed locations and equipped functionalities as the right side of Figure 1[3].

3 Lessons learned in two experiences

3.1 Types of variability source and consequential mechanisms

For finding common characteristics of middleware for smart home environments sources for variation have been analyzed from five aspects; variation in function, control flow, data, technology, quality goals, and environments. In the case of home middleware, most variations occur in control flows and data. And even though the types of resources are the same (e.g., heating and cooling facilities) their controlling behaviors differ. In particular, the most significant variability occurs in data structures related to spaces, the devices installed, and the services provided. As for middleware for ship maintenance most variation sources are also located in control flows and data. However, unlike smart home there is lots of variability related to parameters due to the differences of required data other than entity model variability, so requirements tend to be decomposed into too many functions. Therefore, we used the union structure of C language as variability mechanism to cope with the differences among parameters. Through this, middleware developers could avoid to modify the parameters of the existing functions or to develop new functions.

3.2 The restricted role of feature model and OVM as a resolution

The orthogonal variability modeling (OVM) approach separates the variability model from other development artifacts as it only deals with variability. The OVM approach provides consistent variability modeling notations throughout the different development phases. The feature model that models commonality and variability at the same time has several strengths in terms of expressiveness and understandability, while practitioners have pointed out its weakness such as scalability, traceability with development artifacts, and ambiguity in variability definition. In our experiences we used the OVM approach because of the aforementioned shortcomings of the feature modeling approach. While the OVM approach was good for variability modeling, we suffered from a lack of capability to describe variability with relevant information such as binding time, constraints adhered for the right binding, and rationales for decisions. To tackle these, a tabular format called the variability analysis table was
used to analyze variation points, variants, binding information, and dependency constraints together with OVM approach.

3.3 Limitation of variability tracing with the OVM approach

The variability of a product line is spread all over the development artifacts because variability is implemented by several different artifacts and variability is introduced at all SPL life cycle stages. In the variability model, the variation points and their variants defined at a certain life cycle stage have a relationship with the other relevant development artifacts, either at the same life cycle stage or the adjacent life cycle stage. The OVM approach provides a cross-sectional view of variability across all development artifacts, so it 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). We applied the OVM-based variability tracing approach[2] to trace the variability with the development artifacts. From the initial variability defined in the feature level through the variability in architecture, they are traced with their consistent naming in OVM. Also, the relationships from variability to the development artifacts are traced from OVM to the artifacts produced at each domain engineering stage.

4 Conclusions and future works

This paper described the experiences of applying SPL to developing middleware interworking with diverse devices operating in smart environments. Through the two smart environments we found that there is lots of variability related to control flows and data due to the differences of resource types and their different behaviors other than functional variability. Variability source analysis was helpful to decide product line architecture commonly used by product line members. Another issue in our experiences was how to describe variability with their information such as binding information, constraints related to binding or implementation, and variability implementation relevant decisions and to trace variability with its relevant development artifacts. These issues have not fully resolved in our experimental researches and they are still open issues in product line researches. We are in the progress of validating OVM-based variability tracing.

Acknowledgments. 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