

Product Line Approach to Role-based Middleware Development for Ubiquitous Sensor Network*

Woojin Lee, Sungwon Kang, Dan Hyung Lee
School of Engineering
Information and Communications University, Korea
{wjlee, kangsw, danlee}@icu.ac.kr

Abstract

Currently sensor network middlewares are developed so that they include all functionalities for various nodes. Since sensor network nodes usually operate with limited resources, it is desirable for them to have middlewares that have only the functionalities necessary to perform their roles. This paper proposes a systematic method for developing sensor network middleware using the product line approach. In this method, functionalities of the sensor network middleware common to all nodes and functionalities specific to the different roles of nodes are carefully separated and grouped so that nodes with different roles have different middlewares that are specialized to their roles but minimally consumes the resources.

1. Introduction

A ubiquitous sensor network [1] consists of a number of sensor nodes with various roles that are connected to each other through wireless networks. Sensor network is used for various applications such as environment monitoring, medical care, and military application. However, it is not easy to develop sensor network applications because resources of the nodes in a sensor network are limited, wireless communication between nodes may not be reliable and often nodes should operate when power is low. Numerous such factors make it challenging to develop sensor network applications. Middleware alleviates this problem by allowing the developer to abstract from the operating systems details and supports developing, deploying, executing, and maintaining sensor network applications [2].

Sensor network nodes usually operate with limited resources. So they should have only necessary software modules in order to efficiently perform their functions. On the other hand, sensor network nodes have different roles that require different sets of functionalities. Therefore, it is desirable that sensor network

middlewares can be differently customized for different nodes so that only the functionalities necessary to perform their roles are included.

Currently, several studies on the middleware are under way in order to facilitate sensor network applications development [3-6]. However, most existing sensor network middlewares are not modularized and so it is hard to customize them according to the roles of nodes.

This paper proposes a method for efficiently developing various sensor network middlewares according to the roles of nodes using the product line approach [7]. In this method, functionalities of the sensor network middleware common to all nodes and functionalities specific to the different roles of nodes are carefully separated and grouped so that nodes with different roles have different middlewares that are specialized to their roles but minimally consumes the resources.

2. Related Work

The techniques commonly used for sensor network middleware development are component-based development and modularized development. With help of these techniques the time required for developing middlewares has been reduced. But, it is hard to efficiently develop various kinds of middleware products using the techniques because the techniques are methods for developing single middleware product. Sensor network consists of a number of sensor nodes, so various kinds of middleware products should be developed to support various node applications.

There are new techniques for efficient middleware development. Middleware developed using the aspect-oriented development can easily add new functionalities or change its functionalities without modifying source code [8-10]. In the aspect-oriented development technique [12], requirements are modeled using aspects and so requirements are easily changed

* This work was partially supported by the Industrial Technology Development Program funded by the Ministry of Commerce, Industry and Energy(MOCIE, Korea).

* This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract.

without refactoring source code. Therefore maintainability and modifiability of middleware are improved. Model-driven middleware development [11] is based on the idea that platform-independent middleware model should be first designed based on functional requirements, and then platform-dependent middlewares are constructed from the platform-independent middleware model. Accordingly, various middlewares can be developed by reusing the platform-independent middleware model.

These techniques can be used for developing general application middlewares or embedded middlewares that run on the platforms that have no resource limitations but they are not suitable for sensor network middleware, which have limited resources.

There are also dynamic reconfigurable middlewares [14, 15], which can change modules during run-time. Dynamic reconfiguration technique allows modules to be changed or added during run-time depending on requirements change.

This technique is different from the technique for developing optimal middleware. Middleware requirements can be changed during run-time even if the middleware is optimally developed according to its role, so dynamic reconfiguration technique is necessary to reflect the requirements. Accordingly, our approach is necessary to develop optimal sensor network middleware. Scalability and efficiency of sensor

network middleware will be increased if our approach and dynamic reconfiguration technique are harmonized.

3. Our Approach

Our approach for developing sensor network middleware is based on the software product line engineering approach. Its process consists of two parts - domain engineering and application engineering [7].

In the domain engineering process, middleware features for sensor network nodes with various roles are identified. The architecture for the all inclusive full middleware is designed marked with variation points. Core components for common features are implemented in this part. In the application engineering process, middlewares for nodes are actually developed by designing middlewares specialized to the roles of nodes and implementing the design by reusing domain engineering artifacts. Steps of application engineering process are iterated until middlewares for all kinds of roles for nodes in the specific sensor network are developed. Figure 1 depicts our process for developing sensor network middlewares.

Step D0: Middleware PL Scoping

In this step, sensor network middleware features for various nodes are identified, and commonalities and variabilities of middleware are determined. Features-to-roles mapping table (Table 1) which presents the

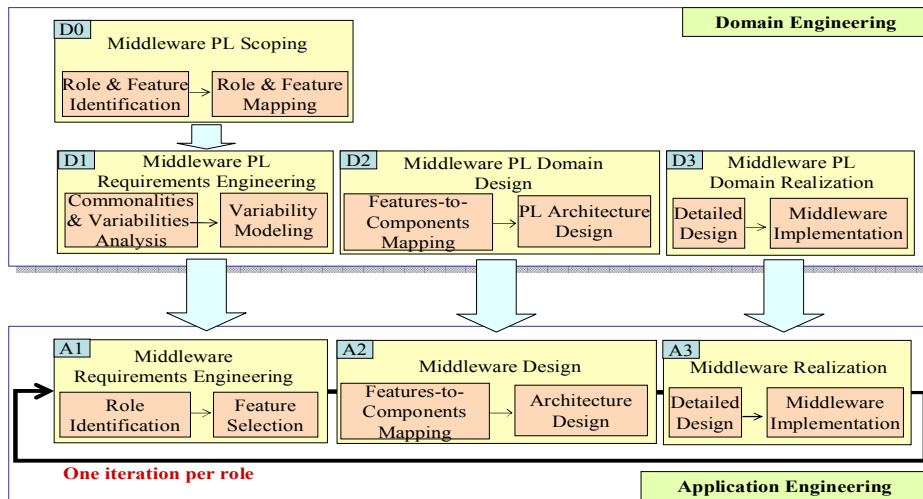


Figure 1. Middleware development process

relationship between middleware features and roles for nodes is created. Commonalities and variabilities of middleware are determined through the table. In Table 1, Feature 2 and Feature 5 become commonalities, and the others become variabilities. This step corresponds to the product line mapping in [13].

Table 1. Features-to-Roles mapping table

		Role 1	Role 2	Role 3	Role 4
Feature Group 1	Feature 1		V	V	
	Feature 2	V	V	V	V
Feature Group 2	Feature 3		V	V	
	Feature 4		V	V	
	Feature 5	V	V	V	V

Step D1: Middleware PL Requirements Engineering

After determining commonalities and variabilities of middleware, variability model is designed. Variability model is designed using Orthogonal Variability Model (OVM) [7]. OVM integrates the variability defined in different software development models into an overall picture of the software variability, so variabilities of middleware can be managed through OVM.

Step D2: Middleware PL Domain Design

To implement middleware, it is necessary to identify components for middleware and design the components based on middleware features. In this step, common features identified in Step D1 are mapped to components. Figure 2 presents the diagram depicting features-to-components mapping. In general, one feature is implemented with one or more components and one component may be used for implementing one or more features.

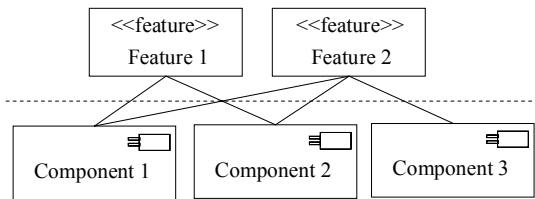


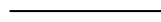


Figure 2. Features-to-components mapping

Once features-to-components mapping is determined, middleware PL architecture is designed. The notation for describing middleware PL architecture is presented in Table 2.

Table 2. Basic notation for PL architecture description

Notation	Description
 Name	Represents a component.
	Represents a variation point. It is a variant constituent of a middleware. It is replaced by components according to the role of a node in the application engineering.
	Represents the association between components, variation points, or a component and a variation point.

Only a basic set of notations is shown in Table 2. During domain design, middleware PL architecture is designed using core components and variation points. Common features are mapped to core components and variable features are marked with variation points. Later, during application engineering variable features are realized with components.

Step D3: Middleware PL Domain Realization

Common components for middleware are designed in detail and implemented. Components are designed and

implemented by considering target operating system and programming language.

Step A1: Middleware Requirements Engineering

This is the first step of application engineering. Middleware features for each role of a node are selected from features-to-roles mapping table. For optional features, corresponding variants are selected from the variability model.

Step A2: Middleware Design

Features-to-components mapping for common features obtained in domain engineering is used in application engineering. In this step, components for implementing middleware features specific to each role of a node are identified and variant features selected from variability model are mapped to the components. Selected variation points for a specific middleware are replaced with components identified in features-to-components mapping and other variation points are removed from PL middleware architecture.

Step A3: Middleware Realization

Components specific for variation part of middleware architecture are designed in detail and implemented. A middleware is completed by composing the components which are implemented in domain engineering or application engineering.

4. Application Example

In this section, the method proposed in Section 3 is illustrated with the example of the Gas Sensing System. The system is composed of sensor nodes, router nodes and a sink node. Sensor nodes sense gas data and transmit the data to router nodes. The router nodes receive the data and transmit it to the sink node. The sink node is connected to the monitoring system and determines the action command, and a gas valve that contains an actuator to turn off the gas valve.

4.1. D0: Product Line Scoping for Sensor Network Middleware

Nodes in the sensor network are classified as sensor, router, sink, and actuator according to their roles. Middleware features for a node are different depending on the role of a node. In this step, all necessary features for sensor network middleware are identified by considering the roles of nodes as shown in Table 3.

Table 3 also shows the relationship between middleware features and roles for nodes. Resource Awareness, Channel Discovering, Security, and Packet Delivery are common features. So they are always included in the sensor network middleware. On the other hand, for example, Location Awareness, Routing, Node Discovering, and Fault Recovery are required only in router and sink nodes.

Table 3. Middleware features required for each role of nodes

Feature	Role	Sensor	Router	Sink	Actuator
Context Awareness	Location Awareness		V	V	
	Resource Awareness	V	V	V	V
Network Management	Routing		V	V	
	Node Discovering		V	V	
	Channel Discovering	V	V	V	V
Service	Security	V	V	V	V
	Fault Recovery		V	V	
	Data Aggregation		V	V	
	Monitoring			V	
	Packet Delivery	V	V	V	V
Hardware Abstraction	Sensing Control	Temperature	V		
		Light	V		
		Gas	V		
		Point-infra-red	V		
		Humidity	V		
	Ultra-sonic	V			
	Actuating Control				V

4.2. D1: Middleware PL Requirements Engineering

Variable middleware features may or may not be included in the middleware depending on the role of a node. A variability model depicts such variations

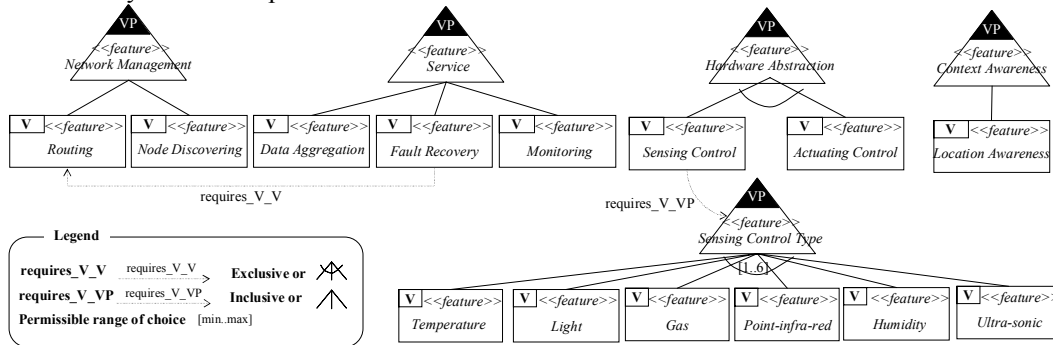


Figure 3. Variability model for middleware

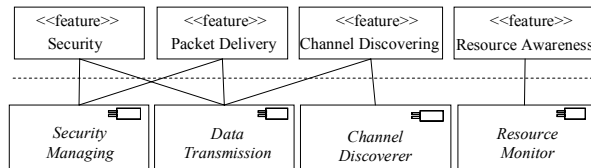


Figure 4. Features-to-components mapping for common features

Figure 5 presents the middleware product line architecture. Components identified in Step D2 are used for common features, and variation points in the variability model designed in Step D1 are used for variant features. Lines between components mean that

among related features. Figure 3 shows a variability model for the Gas Sensing System.

In Figure 3, Network Management variation point has two variants – Routing and Node Discovering feature. Service variation point has three variants - Data Aggregation, Fault Recovery, and Monitoring feature. Fault Recovery variant and Routing variant have a “requires_V_V” relationship, which means that a variant must be selected if another variant is selected. In the sensor network, since routing is necessary to recover communication fault, Routing feature must be selected if Fault Recovery feature is selected.

4.3. D2: Middleware PL Domain Design

After identifying middleware features, components for middleware implementation have to be identified, and features have to be mapped to them. Figure 4 presents the mapping results for common middleware features.

Also in this step, middleware product line architecture is designed and used in the domain engineering phase to design various middleware product architectures according to the role of nodes.

they are connected to each other through interfaces. Using this product line architecture, specific middleware architecture for a node can be designed in the domain engineering phase by selecting appropriate variant features first according to the role of the node and then replacing it with the components for the introduced features.

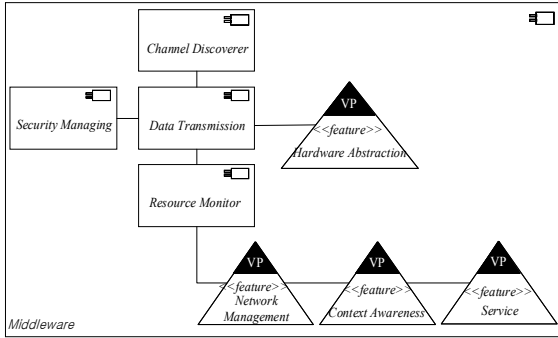


Figure 5. Middleware product line architecture

4.4. D3: Middleware PL Domain Realization

In Step D3, core components are designed and implemented. In our example, there are four core components mapped from common features as presented in Figure 4. They are Security Managing, Data Transmission, Channel Discoverer, and Resource Monitor. In this step, middleware components are designed in detail and implemented by considering target operating system for the sensor network.

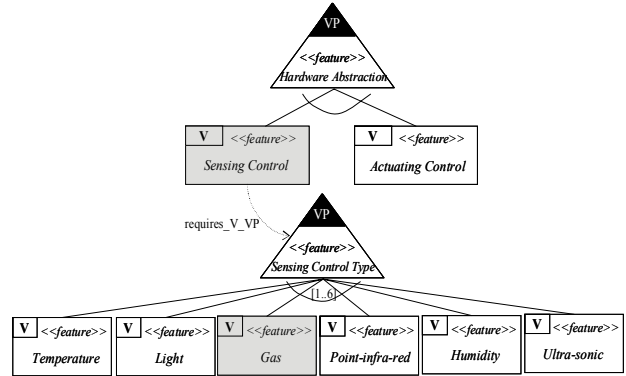
4.5. Application Engineering

The steps from middleware requirements engineering to middleware design is described by using the artifacts generated in domain engineering.

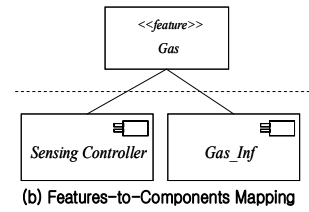
Table 4 shows the selected middleware features for gas sensor nodes. Shaded boxes are the selected features for the gas sensor node.

Table 4. Selection of middleware features for gas sensor nodes

Feature	Role	Sensor	Router	Sink	Actuator
Context Awareness	Location Awareness		v	v	
	Resource Awareness	v	v	v	v
Network Management	Routing		v	v	
	Node Discovering		v	v	
Service	Channel Discovering	v	v	v	v
	Security	v	v	v	v
	Fault Recovery		v	v	
	Data Aggregation		v	v	
	Monitoring			v	
Hardware Abstraction	Sensing Control	Temperature	v		
		Light	v		
		Gas	v		
		Point-infra-red	v		
		Humidity	v		
	Ultra-sonic	v			
Actuating Control					v



(a) Feature Tree



(b) Features-to-Components Mapping

Figure 6. Features-to-components mapping for gas sensor nodes

Figure 6 shows the selected variants from the variability model designed in domain engineering and the features-to-components mapping. Gray boxes of Figure 6(a) shows selected variants for gas sensor nodes and Figure 6(b) shows the mapping from a selected variant feature to the components that realize it.

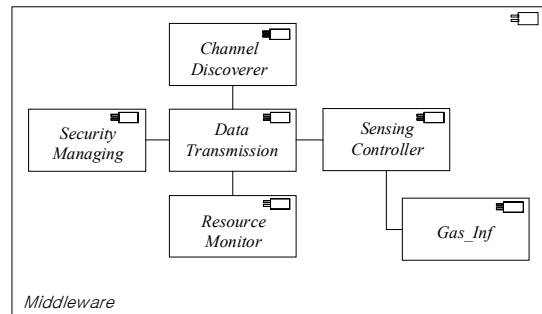


Figure 7. Middleware architecture for gas sensor nodes

In Figure 7, four components - Security Managing, Channel Discover, Data Transmission, and Resource Monitor – are the core components generated in domain engineering, and two components - Sensing Controller and Gas_Inf – are specific middleware components for the gas sensor node.

Middleware architecture for gas sensor nodes is designed using middleware PL architecture. Appropriate variation points among the variation points in the PL architecture are selected according to the selected features shown in Table 4. The variation

points, then, are replaced with components through features-to-components mapping. Figure 7 shows the middleware architecture for the gas sensor node.

5. Conclusion

This paper proposed a systematic method for developing sensor network middlewares that are specialized according to the roles of nodes using the product line approach. In order to help apply the product line approach to sensor network middlewares development, this paper (1) identified various roles of sensor network nodes, (2) analyzed common middleware features and variant middleware features for the roles, and (3) presented notation for describing middleware product line architecture based on Orthogonal Variability Model [7]. This paper also showed feasibility of the proposed technique through an application example.

The contributions of the paper are twofold: First, by applying the product line approach to sensor network middleware development, we opened up a new domain for applying the software product line approach. Middlewares are commonly perceived as complex and expensive software to develop and thus as the things that should be deployed as a whole even at the expense of using up much system resources. We believe that the notion of middleware specialized to the role of the system containing it will be useful in many areas and especially so when system resources are limited as in sensor networks. As the product line development methods and tools are developed more and more and becomes advanced, production of specialized middlewares and, more generally, production of particular products of product lines will require less and less effort. Second, by providing a concrete process together associated models and notations and working out example middleware development for the gas sensor node which is a constituent of the sensor network for the Gas Sensing Application, we illustrated our method so that with a little adaptation the method in this paper can be easily applied to different sensor networks and also to other software where the application of the approach is justified.

References

- [1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, March 2002, 38(4):393–422.
- [2] K. Römer, O. Kasten, and F. Mattern, "Middleware Challenges for Wireless Sensor Networks," *ACM SIGMOBILE Mobile Commun. and Commun. Rev.*, vol. 6, no. 2, 2002.
- [3] T. Liu and M. Martonosi, "Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems," Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP 03), 2003, pp. 107–118.
- [4] Eduardo Souto, Germano Guimarães, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, Carlos Ferraz, "A Message-Oriented Middleware for Sensor Networks," Proc. 2nd Int'l Workshop Middleware for Pervasive and Ad-Hoc Computing (MPAC 04), 2004, pp. 127–134.
- [5] P. Levis and D. Culler, "Mate: A Tiny Virtual Machine for Sensor Networks," Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), 2002, pp. 85–95.
- [6] D.M. Lee, S.H. Han, I.S. Park, S.H. Kang, K.M. Lee, S.J. Hyun, Y.H. Lee, and G.H. Lee, "A Group-Aware Middleware for Ubiquitous Computing Environments," Proc. 14th Int'l Conf. Artificial Reality and Telexistence (ICAT 2004), 2004.
- [7] K. Pohl, G. Bockle, F. Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, 2005.
- [8] S.S. Yau, Y. Wang and D. Huang, "Middleware Support for Embedded Software with Multiple QoS Properties for Ubiquitous Computing Environments," Proceedings of The Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2003, pp. 250- 256.
- [9] D. Kaul and A. Gokhale, "Middleware Specialization using Aspect Oriented Programming," Proceedings of the 44th ACM SE conference, 2006.
- [10] F. Hunleth, R. Cytron, and C. Gill, "Building Customizable Middleware Using Aspect Oriented Programming," Proceedings of the 2001 ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'01), 2001.
- [11] A. Gokhale, K. Balasubramanian, J. Balasubramanian, A.S. Krishna, G. Edwards, G. Deng, J. Parsons and Douglas C. Schmidt, "Model Driven Middleware: A New Paradigm for Developing and Provisioning Distributed Real-time and Embedded Applications," *Elsevier Journal of Science of Computer Programming, Special Issue on Model Driven Architectures*, 2005.
- [12] K. Lieberherr, D. Orleans and J. Ovlinger, "Aspect-oriented programming with adaptive methods," *Comm. ACM*, Vol. 44, No. 10, Oct. 2001, pp.39-41.
- [13] K. Schmid, "Product Line Mapping Report," IESE-Report No. 028.00/E Release 1.0, 2000.
- [14] G. Coulson, G.S. Blair, M. Clarke and N. Parlavantzas, "The Design of a Configurable and Reconfigurable Middleware Platform," *Distributed Computing*, Volume 15, Issue 2, Springer-Verlag, April 2002, pp.109-126.
- [15] N. Wang, C. Gill, D.C. Schmidt, and V. Subramonian, "Configuring Real-Time Aspects in Component Middleware," CoopIS/DOA/ODBASE 2004, LNCS 3291, Springer-Verlag, 2004, pp.1520–1537.