A Feature-Based Approach to Product Line Production Planning

Jaejoon Lee[1], Kyo C. Kang[1], and Sajoong Kim[2]

[1] Department of Computer Science and Engineering,
Pohang University of Science and Technology (POSTECH), PIRL,
31 San, Hyoja-Dong, Nam-Gu, Pohang, Kyoung-buk, Republic of Korea
{gibman, kck}@postech.ac.kr
http://selab.postech.ac.kr
[2] Korea Software Institute (KSI), Korea IT Industry Promotion Agency (KIPA),
KIPA Bldg., 79-2, Garakbon-dong, Songpa-gu, Seoul, Republic of Korea
sjkim@software.or.kr

Abstract. A production plan, which describes how core assets are used to develop products, has an important role in product line engineering as a communication medium between core asset developers and product developers. Recently, there have been efforts to address issues related to production planning; however, most of them focus on the process and business/management aspects of production planning, and not much emphasis is given to technical issues such as deciding features that will be made as core assets and their granularity.

In this paper, we introduce a feature-based approach to product line production planning and illustrate how our approach addresses these technical issues. In our approach, a feature model and feature binding information are used as primary input to production plan development. A product line production plan developed using our approach could be easily customized to a product-specific production plan, because it was developed with consideration of units of product configurations as well as their integration techniques.

1 Introduction

With the product line engineering paradigm, a common set of core assets is developed and used to develop products of a product line. The core assets include requirements specifications, architecture models, software components, and adopted COTS components [1]. A production plan, which describes how the core assets are used to develop products, has an important role in product line engineering as a communication medium between core asset developers and product developers. That is, the production plan provides product developers with the way core assets are customized and integrated into products.

Recently, there have been efforts to address issues related to production planning [2], [3]. In [2], guidelines for developing a production plan and production plan evaluation criteria are proposed, and case studies are included in [3]. Most of these efforts, however, focus on the process and business/management aspects of production

planning, and not much emphasis is given to technical issues such as deciding what features will be implemented as core assets and how big each core asset will be.

In this paper, we introduce a feature-based approach to product line production planning and illustrate how our approach addresses these technical issues. In our approach, a feature model, which captures commonality and variability information of a product line, and feature binding information [4] of what and when features are included to products and delivered to customers are used as primary input to production plan development. In product line engineering, a feature model plays a central role in the management and configuration of multiple products and, therefore, core assets should be identified based on features. Also, feature binding analysis provides asset developers with information on the granularity of and binding techniques for the core assets.

The final work product of our approach is a product line production plan document for which we follow the structure of a production plan in [2]. (Note that, of the chapters and sections in [2], we only describe sections that are relevant to our approach.) Also, a Home Integration System (HIS) product line, which controls and manages a collection of devices to maintain security and safety of a building or a house, is used to demonstrate the concept (See Table 1 for the product features.). HIS generally includes the features in the table below. More advanced products may include features, such as climate control and lighting, that optimize living conditions.

Table 1 Product Features of an HIS Product Line

Product Feature	Explanation
fire detection & control	Fire events are detected by monitoring smoke detectors and heat sensors installed in the house. When a fire event is detected, HIS turns the alarm and all sprinklers on and unlocks all HIS-controlled doors. HIS also sends a pre-recorded voice message to the fire station and the owner over the telephone line to inform them of the incident. Once the fire is under control, the alarm and all sprinklers will be turned off but doors will remain unlocked for the duration of time preset by the owner.
intrusion detection & control	Intrusion events are detected by monitoring motion sensors. When an intrusion event is detected, HIS turns the alarm on and locks all HIS-controlled doors. Also, HIS sends a voice message to the police station and the owner.
flood detection & control	Flood events are detected by monitoring moisture sensors. When a flood event is detected, HIS shuts off the water main of the house. When moisture is detected on the basement floor, the sump pump will be activated.
security	The entrance and exit of all personnel are verified and recorded with identification information (e.g., name, time, ID number, etc.). There are various devices for the verification such as fingerprint recognition, voice recognition, etc. Also, the access to every room inside a building can be controlled by each person's job function.

Section 2 gives an overview of the activities of our approach, and the feature binding analysis and product line asset identification activities are described in sections 3 and 4, respectively. The product line core asset development activity is described in section 5 and product line production plan documentation is illustrated with an HIS product line production plan example in section 6. Section 7 summarizes and concludes this paper.

2 Product Line Production Planning Activities

Product line engineering consists of two major engineering processes, product line asset engineering and product engineering, each of which consists of various activities [1], [5]. Of many activities of product line asset engineering process, we identified activities related to product line production planning (See Fig. 1 for these activities and their relationships.). These activities are iterative and the arrows in Fig. 1 show data flows, i.e., use of work products at each activity. Each activity is briefly described below.

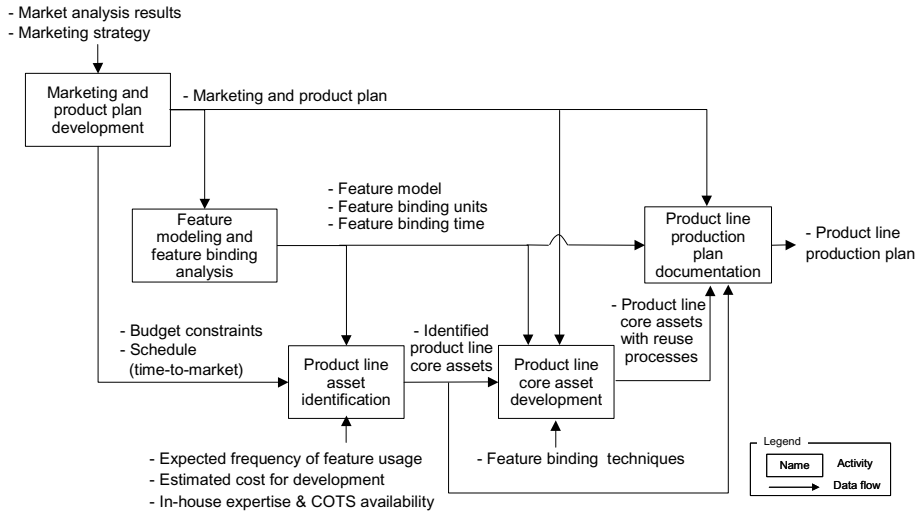


Fig. 1 Product Line Production Planning Activities

Developing a Marketing and Product Plan (MPP) for a product line initiates product line production planning. The MPP, which includes a marketing plan and a product plan, sets a specific context for product line analysis and reuse exploration for the product line. The marketing plan in MPP provides, for each market segment, information of an assessment of needs, the time-to-market, the price range, and a marketing strategy for realizing business opportunities in the market. In the product plan of MPP, product features are identified.

In feature modeling, product features from the MPP are organized into an initial feature model and this feature model is extended with design features such as operating environments, domain technologies, and implementation techniques to be used. Feature binding analysis then identifies feature binding units from the feature model and determines binding time between binding units. The results of this activity are a feature model, feature binding units, and the feature binding time information.

The product line asset identification activity takes the feature model and the feature binding analysis results as primary input. For each feature binding unit, its asset type

(i.e., core asset, product-specific asset, or COTS) is determined with consideration of budget constraints, the time-to-market, and other business/technical considerations including availability of in-house expertise.

The primary input to product line core asset development includes a feature model and feature binding units and their binding time. In this activity, variation points are identified and feature binding techniques are explored to support the required feature binding time.

The product line production plan documentation activity integrates work products provided by the other activities, as shown in Fig. 1. After the marketing and business analyses information from the MPP is incorporated into the production plan, it is refined with binding units and identified product line core assets. Then feature binding units, which are annotated with the binding information and reuse processes, are documented.

Of the production planning activities, details on marketing and product plan development can be found in [5] and [6]. The rest of the production planning activities are described in the following sections. We adapted and used the MPP example in [5] to illustrate our approach (See Table 2.).

Table 2 An MPP Example for an HIS Product Line

Marketing and Product Plan for HIS product line			
Market segments		**Office building (high-end product)**	**Household (low-end product)**
Marketing plan	**Need assessment**	The customer's choices of features for high-end products are in the wide range of variability. Moreover, the *Security* feature has customer-specific requirements.	The customers are budget-conscious and they only require features that are essential for HIS products.
	Time-to-market	Less than six months	Less than three months
	Price range	To be a competitive product, the price should be less than 20,000 dollars.	Less than 1,000 dollars
	Marketing strategy (product delivery methods)	Develop and deliver a product for each customer	Prepackaged
Product plan	**Product features**	Fire, Intrusion, Flood, Security, and other customer specific features	Fire, Intrusion
	Quality attributes	Safety, Reliability, Scalability	Safety, Reliability, Usability

3 Feature Modeling and Feature Binding Analysis

A feature model captures commonalities and variabilities of a product line in terms of product features. Fig. 2 shows the feature model of an HIS product line. Capability features of HIS consist of service features (e.g., *Fire, Intrusion, Flood,* etc.) and operational features (e.g., *Alarm*, *Pumping,* etc.). Operating environment features of HIS include *Moisture Sensor* and *Sump Pump*, and domain technology features include technical features (e.g., *Monitoring & Detecting*) for implementing service and operational features. Compared with domain technology features, implementation technique features are more generic and might be applicable to other product lines (e.g., the *TCP*

and *UDP* features are used to provide an Internet connection in the HIS product line, but they can also be used in other product lines.). Details of feature analysis and modeling guidelines can be found in [7].

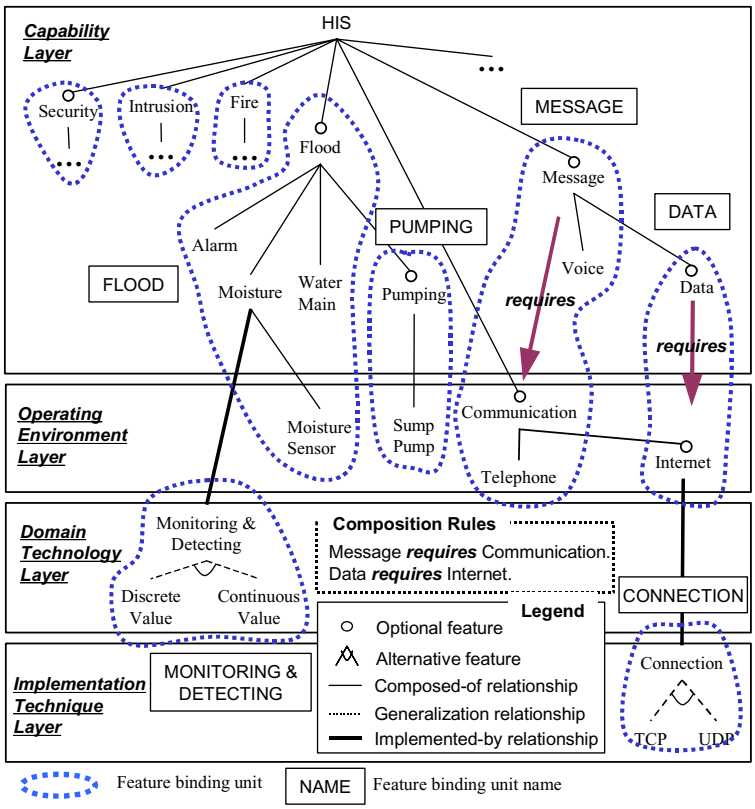


Fig. 2 Feature Binding Unit Identification: an HIS Product Line Example

After feature modeling, feature binding analysis is performed using the feature model and the MPP. Feature binding is examined from three perspectives: what features are bound together (feature binding units), when features are bound into products (feature binding time), and how features are bound (feature binding techniques). Feature binding unit and feature binding time analyses are discussed in this section. The feature binding techniques are briefly explored in section 5.2.

3.1 Feature Binding Unit Analysis

We define a feature binding unit as a set of features that are related to each other via composed-of, generalization/specialization, or implemented-by relationships, and composition rules (i.e., require and mutually exclude) of a feature model. Features that belong to a binding unit work for a common service and, therefore, they have to exist together for correct operation of the service.

Feature binding unit identification starts with identification of independently configurable service features. (For short, we will call these features as service features in the remainder of this paper.) A service feature represents a major functionality of a system and may be added or removed as a service unit. In HIS, *Flood*, *Fire*, and *Intrusion* features are examples of service features.

A service feature uses other features (e.g., operational, environmental, and implementation features) to function properly and the constituents of a binding unit can be found by traversing the feature model along the feature relationships and composition rules. For example, as we start from the *Flood* service feature, *Alarm, Moisture*, *Water Main*, *Pumping*, *Moisture Sensor*, *Sump Pump*, and *Monitoring & Detecting* features can be identified. All these features are needed to provide the flood service.

Within a feature binding unit, there may exist optional or alternative features that should be selected based on customer's needs. These features impose variations on the component design and, therefore, they have to be identified as separate feature binding units. For example, only one of the sub-features of *Monitoring & Detecting* can be selected based on the device type that a customer may choose. (See the *Monitoring & Detecting* feature at the domain technology layer in Fig. 2.)

Note that the *Communication* feature is included in the *MESSAGE* feature binding unit, although the feature is optional. This is because the *Message* feature requires the *Communication* feature according to the composition rule, and they have to be together to provide the message service properly. (See the arrows for the 'require' composition rule in Fig. 2.) After all feature binding units are identified, a name is assigned to each feature binding unit; the feature name that represents a binding unit was given to the corresponding feature binding unit but the name was written in upper case letters to distinguish it from the name of the feature. (See the dotted circles and the names of binding units in Fig. 2.) Once features are grouped into feature binding units, a binding time analysis is performed for the feature binding units.

3.2 Feature Binding Time Analysis

Generally, feature binding time has been looked at from the software development lifecycle viewpoint ('product lifecycle view') [8], [9], in which the focus has been the lifecycle phase incorporating a feature into a product. In product line engineering, however, there exists another dimension that is based on the *binding state* of a feature binding unit. That is, some feature binding units may be developed and included in product line core assets at core asset development time, but their availability can be determined at installation time by enabling or disabling the feature binding units. Furthermore, activation of the available feature binding units may be controlled to avoid a feature interaction problem[1]. Thus, feature binding time analysis with an additional view on *feature binding state* (which includes *inclusion* and *availability* states and *activation rules*) provides a more precise framework for feature binding analysis.

[1] The problem of unexpected side effects when a feature is added to a set of features is generally known as the feature interaction problem.

The ‘product lifecycle view’ consists of four phases: core asset development, product development, pre-operation, and operation. After product line core assets are developed, a product is developed with product specific features and the core assets. Then, the product is delivered, installed, and configured for a customer during the pre-operation phase.

Each phase of the product lifecycle shows the binding states of feature binding units. For example, if the inclusion and availability states of a feature binding unit are determined during product line core asset development, the feature binding unit is allocated to both the inclusion and availability columns of the core asset development phase. (See the *FIRE* and *INTRUSION* feature binding units in the bottom row in Fig. 3.) If the inclusion state of a feature binding unit is determined during product development and the availability state of the feature binding unit is determined during installation, the feature binding unit is allocated to the inclusion column of the product development phase and also to the availability column of the pre-operation phase. (*FLOOD* and *MESSAGE* are examples of such feature binding units.)

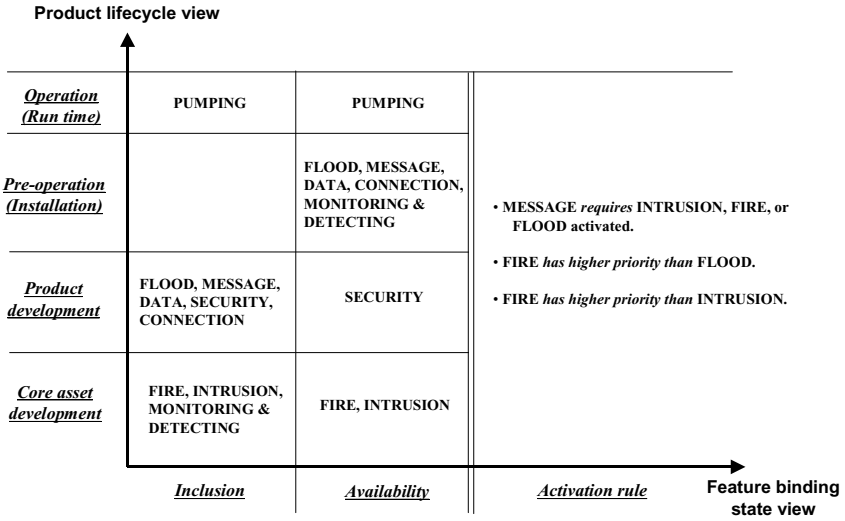


Fig. 3 Feature Binding Time Analysis

In the ‘feature binding state view,’ the inclusion feature binding state indicates when, in the product lifecycle phases, a feature binding unit is physically included to a product, and the availability binding state indicates when, in the product lifecycle phases, those included feature binding units become available to users (i.e., the feature binding unit is ready for use with all its implementation techniques that are bound). Once the feature binding unit becomes available, it is now ready to be activated, as long as it abides by the activation rules among feature binding units. (See the horizontal axis in Fig. 3.)

The activation rules provide information on concurrency of feature binding unit activation and they are defined in terms of mutual exclusion, dependency, and priority

schemes. As an intuitive example, room temperature can be kept stable by turning on both an air-conditioner and a heater at the same time, but this is not a desirable behavior. Their activation rule should be 'mutual exclusion' to avoid such situation.

The results of this activity are feature binding units and their binding time, and it is important to note that the basic units for configuring a product in our approach are the feature binding units. Hence, the 'Detailed production process' section of the production plan is described in terms of feature binding units and other binding attributes such as binding time and binding techniques. In the next section, product line asset identification based on the feature model and feature binding units is explained.

4 Product Line Asset Identification

Before developing product line core assets, we must determine which features will be made as core assets, product-specific assets, or purchased as COTS. Therefore, for each feature, its asset type (i.e., core asset, product-specific asset, or COTS) should be determined with consideration of the budget and time-to-market constraints and other business/technical considerations such as expected frequency of feature usage, estimated cost for development, and availability of in-house expertise. (Table 3 shows some of the identified product line assets of the HIS product line.)

Table 3 Identified Product Line Assets

Feature binding unit	Constituent features	Frequency of feature usage	COTS availability / COTS price (which is compared to the estimated in-house development cost)	Asset type
FIRE	Fire, Smoke, Smoke sensor, Sprinkler, ...	High	No / -	Core asset
FLOOD	Flood, Moisture, Moisture sensor, Alarm, ...	Medium	No / -	Core asset
MESSAGE	Message, Voice	Medium	Yes / Higher	Core asset
	Communication, Telephone	Medium	Yes / Lower	COTS
SECURITY	Security, Access-control, ...	Low	No / -	Product specific asset
	Biometric	Low	Yes / Lower	COTS

For example, in the HIS product line, the *Fire* feature has high frequency of usage (i.e., all products in the product line include it) and the estimated cost for development is low; this feature is identified as a core asset. The *Security* feature, however, has low frequency of usage and has customer-specific requirements; this feature is identified as a product-specific asset, i.e., it will be developed as asset when it is needed. For another example, the *Biometric* feature, which is used to authenticate users, must be developed in a short period but in-house expertise for the biometric technique is not available; COTS components will be purchased to implement this feature.

We have illustrated the product line analysis activities and, in the next section, how the analysis results are used to develop product line core assets is discussed.

5 Product Line Core Asset Development

In product line engineering, core assets include requirements specifications, architecture models, software components, and adopted COTS components [1]. Of these core assets, we discuss how asset software components are developed.

The primary input to product line component development includes a feature model, feature binding units and their binding time, architecture models, and a design object model[2]. Design objects are the embodiment of functionalities required for the product line. Once a design object model is defined, these objects in the model are allocated to components for implementation. In this section, identification of variation points in the design object model, exploration of binding techniques, and specification of product line components are illustrated with examples.

5.1 Variation Point Identification

For feature binding to be feasible, variation points for optional and alternative binding units should be identified in the design object model. Since features of a binding unit should exist together, their binding to a product should be explicitly identified in the design object model. We also need to be sure that all objects that implement the features of a binding unit are bound together with appropriate implementation techniques. For example, when the *FLOOD* binding unit is incorporated into a product and becomes available, the objects that implement each of its constituent features (i.e., *Moisture*, *Moisture Sensor*, and *Alarm*) should also be bound in the product for correct operation of *FLOOD*.

To manage variation points of a binding unit consistently, explicit mappings between binding units and variation points must be established. If there is difficulty establishing this relationship, the related objects should be examined for further decomposition, refinement, or restructuring. If a binding unit is optional and its parent binding unit is also optional, its binding requires the parent binding unit be bound beforehand, and this dependency should also be preserved among variation points in the object model. For example, *FLOOD* should be bound before the binding of *PUMPING*. This dependency is preserved in the object model, as the variation point of *FloodResponder* object is located at a lower level than the variation point of *EventResponder* object in the aggregation hierarchy. (See Fig. 4.)

In Fig. 4, each bubble represents an optional or alternative binding unit and arrows show the corresponding variation points (denoted by ■) identified in the object model. For example, the variation point for *FLOOD* is identified at the end of the line connecting *EventResponder* and *FloodResponder* objects for an aggregation relationship. That is, if *FLOOD* is determined not to be available in a product at pre-operation time

[2] A feature-based approach to object-oriented development can be found in [10].

(See Fig. 3.), then the aggregation relation between the two objects is removed, and the objects that implement the *FLOOD* binding unit are not accessible by users. After the variation points are identified, implementation techniques for feature binding should be explored.

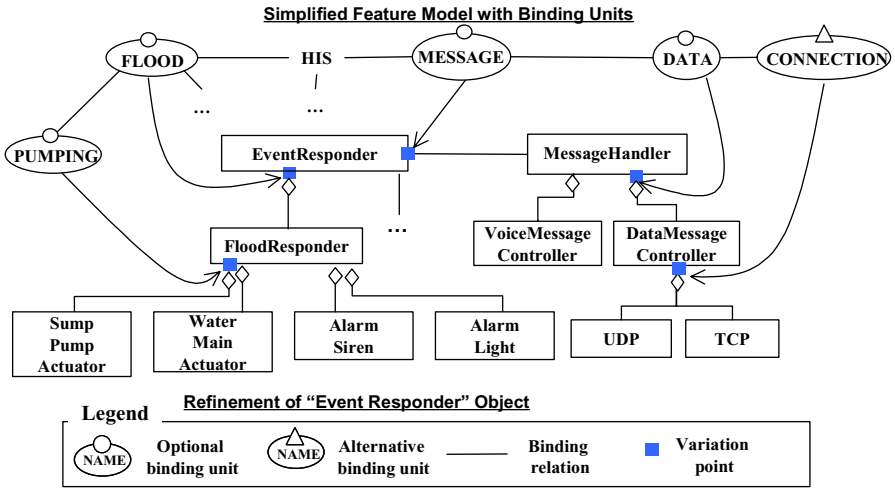


Fig. 4 Mappings between Binding Units and Variation Points in the Design Object Model

5.2 Feature Binding Technique Exploration

Selection of binding techniques depends both on binding time and quality attributes (e.g., flexibility) required for products. Delaying binding time to a later phase of the lifecycle may provide more flexibility, but applicable implementation techniques are limited and they usually require more performance overheads. Therefore, guidelines for the selection of feature binding techniques are required to help asset developers make decisions properly.

For that purpose, we propose a classification of feature binding techniques based on the feature binding states: binding techniques for the feature 'inclusion' and those for the feature 'availability.' Techniques belonging to the former class should be able to control feature inclusion by including or excluding code segments or components from products. Code generation, pre-processing, macro processing [11], ICD (Internet Component Download) [12] are some examples of this class. These techniques allow products to include multiple features physically but their availability can be determined at a later phase. (See the left two columns in Fig. 3.).

The second class of techniques provides mechanisms for enabling or disabling accesses to features. Load tables and authentication based access control [13] are techniques that belong to this class. In the HIS product line, for instance, the load table technique is used to determine the availability of *FLOOD*, *DATA*, etc. at the pre-

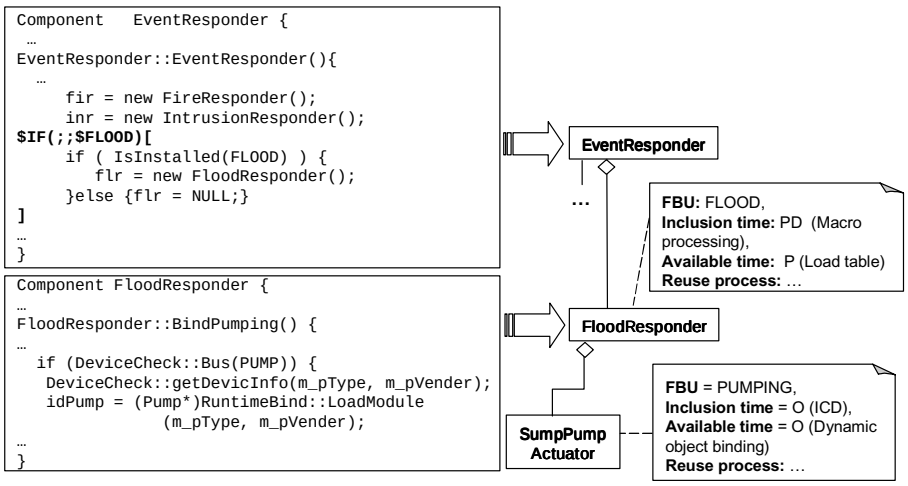
operation phase. When the system starts to execute, it refers to the load table to determine which features should be made available to the user.

In addition to those techniques belonging to the two classes, we should also explore techniques for dynamic or static binding of features. While some features may be bound statically, other features may require dynamic binding for flexibility or memory space efficiency. Dynamic binding of objects, menus, and plug-ins [14] are techniques that belong to this class. For example, *PUMPING* is bound at the operation time, as its device drivers for sump pumps may vary.

In the following section, an implementation of the HIS example is illustrated.

5.3 Component Specification

Next, we refine the design object model into concrete components with the selected feature binding techniques. Product line component design consists of specifications of components and relationships among them. (See Fig. 5 for the specifications of the *EventResponder* and *FloodResponder* components.)



- FBU: Feature Binding Unit - PD: Product Development time - P: Pre-operation time
- O: Operation time - ICD: Internet Component Download

Fig. 5 Component Specifications of *EventResponder* and *FloodResponder* with Annotations of Feature Binding Information

For product development time inclusion of *FLOOD*, the macro language (i.e., `IF(;;$FLOOD)[…]`) is used, and for the pre-operation time availability, the load table is used. Instantiation of the *FloodResponder* depends on the return value of `IsInstalled(FLOOD)`, which confirms whether or not *FLOOD* is allowed to be made available. (See the upper left part of Fig. 5.)

As a customer is authorized to use *PUMPING* at operation time (See the operation time row in Fig. 3.), the `FloodResponder::BindPumping` method of the

FloodResponder is invoked. Then, it searches for an appropriate device driver for the installed sump pump and binds it to provide the pumping service. (See the lower left part of Fig. 5.)

Once product line components are developed, each component is annotated with binding information such as mappings to feature binding units, their binding time, and binding techniques, and a reuse process (See the notes attached to *FloodResponder* and *SumpPumpActuator* in Fig. 5.). For example, *FloodResponder* is mapped to the *FLOOD* binding unit, and the time it is included to the product and the time it becomes available for use are product development time and pre-operation time, respectively. Also, binding techniques used are described in parentheses. For the product development time inclusion of *FLOOD*, the macro processing is used and, for the pre-operation time availability, the load-table is used.

In the next section, the product line production plan documentation activity, which integrates work products of other activities, is described.

6 Product Line Production Plan Documentation

This activity starts with documenting the information from MPP. For instance, the target products in the product plan for each market segment are documented in the 'Products possible from available assets' section of Fig. 6. Also, product delivery methods in the marketing strategy, which describe how products will be delivered to customers, initially outline the 'Production strategy' section in the production plan. For example, products for the low-end market segment only have core features (i.e., *Fire* and *Intrusion*) and these features are prepackaged in all products; an automatic code generation approach is used to develop the low-end products. On the other hand, products for the high-end market segment include customer specific features (e.g., *Security*) and these products are developed for each customer; a custom-made approach, in which a product is developed for each customer, is used for the high-end products.

After the information from MPP is incorporated into the production plan, the 'Overview of available core assets' section is refined with binding units and identified product line core assets. In our approach, the product line asset identification results (i.e., Table 3 in section 4) are used to provide an overview, which is in the middle of Fig. 6.

Now the feature binding units are documented in the 'Detailed production process' section with information on commonality, asset type, associated binding units, a functional description, binding time, binding techniques, and reuse processes. For example, *FLOOD* is an optional binding unit and it has the *PUMPING* child binding unit. The reuse process describes the way product developers can include *FLOOD* and make it available for use. To include *FLOOD*, product developers should select the *Flood* feature when generating code for a product. Also, product developers should release the product with the HIS-Installer package, which configures the load table, so that the availability of *FLOOD* can be controlled at installation time.

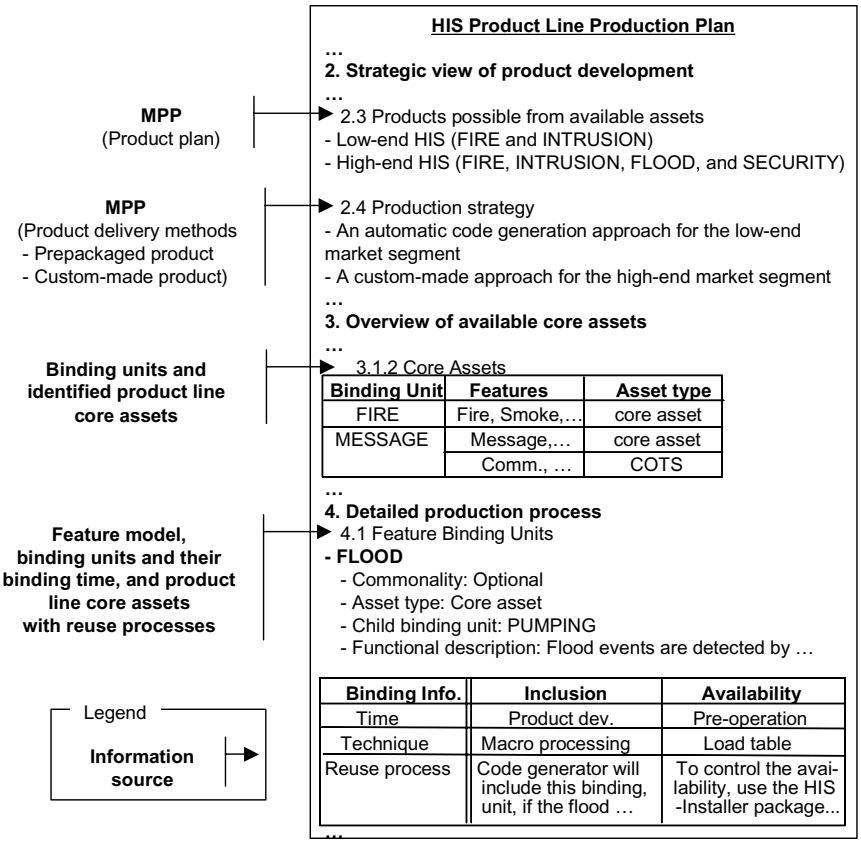


Fig. 6 An HIS Product Line Production Plan Example

Finally, the production plan should be validated to see if the products in MPP could be produced following the production process.

7 Conclusion

In this paper, we introduced a feature-based approach to product line production planning, and illustrated how a feature model and feature binding information are used to identify core assets and develop a product line production plan. A product line may be targeted for more than one market segment and each market segment may require a unique set of features, a certain feature binding time, and/or feature binding techniques that are different from others. Therefore, product line assets must be identified with consideration of not only the commonalities and variabilities of a product line, but also the feature binding requirements. The explicit identification of feature binding unit and binding time is essential for identifying and managing consistency among variation points, and selecting appropriate binding techniques.

We found that our approach provided asset developers with an explicit way to identify and organize core assets, and determine asset types with technical and business/management considerations. A product line production plan developed using our approach could be easily customized to a product-specific production plan, because it was developed with consideration of units of product configurations as well as their integration techniques (i.e., binding techniques) determined with consideration of required binding time and organizational production strategies.

Our future work includes a tool support to derive a product-specific production plan from a product line production plan. We believe that the feature orientation of our approach makes it easy to achieve this goal and we hope that this research will lead us to develop more detailed guidelines for developing a product line production plan.

References

1. Clements, P., Northrop, L.: *Software Product Lines: Practices and Patterns*, Addison Wesley, Upper Saddle River, NJ (2002)
2. Chastek, G., McGregor, J.D.: Guidelines for Developing a Product Line Production Plan, *Technical Report CMU/SEI-2002-TR-006*, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University (2002)
3. Chastek, G., Donohoe, P., McGregor, J.D.: Product Line Production Planning for the Home Integration System Example, *Technical Note CMU/SEI-2002-TN-029*, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University (2002)
4. Lee, J., Kang, K.: Feature Binding Analysis for Product Line Component Development. In: van der Linden, F. (eds.): Software Product Family Engineering. Lecture Notes in Computer Science, Vol. 3014. Springer-Verlag, Berlin Heidelberg (2004) 266-276
5. Kang, K., Lee, J., Donohoe, P.: Feature-Oriented Product Line Engineering. IEEE Software, **19**(4), July/August (2002) 58-65
6. Kang, K., Donohoe, P., Koh, E., Lee, J., Lee, K.: Using a Marketing and Product Plan as a Key Driver for Product Line Asset Development. In: Chastek, G. (eds.): Software Product Lines. Lecture Notes in Computer Science, Vol. 2379. Springer-Verlag, Berlin Heidelberg (2002) 366-382
7. Lee, K., Kang, K., Lee, J.: Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. In: Gacek, C. (eds.): Software Reuse: Methods, Techniques, and Tools. Lecture Notes in Computer Science, Vol. 2319. Springer-Verlag, Berlin Heidelberg (2002) 62-77
8. Czarnecki, K, Eisenecker, U.: Generative Programming: Methods, Tools, and Applications, Reading, MA: Addison Wesley Longman, Inc. (2000)
9. Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J. H., Pohl, K.: Variability Issues in Software Product Lines. In: van der Linden, F. (eds.): Software Product Family Engineering. Lecture Notes in Computer Science, Vol. 2290. Springer-Verlag, Berlin Heidelberg (2002) 13-21
10. Lee, K., Kang, K., Chae, W., Choi, B.: Feature-Based Approach to Object-Oriented Engineering of Applications for Reuse. Software Practice and Experience, **30**(9), (2000) 1025-1046
11. Basset, P. G.: Framing Software Reuse: Lessons From The Real World. Prentice Hall, Yourdon Press (1997)

12. Microsoft Developers Network (MSDN): Introduction to Internet Component Download (ICD), http://msdn.microsoft.com/workshop/delivery/download/overview/entry.asp
13. Sun Microsystems, Inc.: Java Authentication and Authorization Service (JAAS), http://java.sun.com/security/jaas/doc/api.html
14. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. MA: Addison Wesley Longman, Inc. (1995)