An Integer Programming Based Decision Making Framework for Application Engineering in Software Product Line Development

Mohammad Fajar[†1] Tsuneo Nakanishi[†2]
Kenji Hisazumi[†3] Akira Fukuda[†4]

Abstract: Software product line (SPL) is emerging as an important paradigm for software development. In the SPL paradigm, development of each variant product, *i.e.* a member of the product line, is performed by configuration of core assets, which are artifacts shared and managed among all the members of the product line. Although this configuration must be done to satisfy requirements and constraints of the product, it can be time consuming and expensive for the large scale product line. Hence, in this paper, we propose a decision making framework with integer linear programming (ILP) for configuration process. In this paper we first present the framework in details and then we apply it to a real case study of the wireless sensor network (WSN) system family for field monitoring in order to find an optimal configuration in feature selection process.

Keywords: Software Product Line, Application Engineering, Variability Modeling, Feature Model, Integer Linear Programming

1. Introduction

Software product line (or *SPL* for short) is emerging as a paradigm of software reuse among products in the product line. It has been reported that SPL development achieves impressive time-to-market and cost reduction as well as productivity and quality improvement of software intensive products [9, 10]. SPL reuses not only software components but also software artifacts of various abstraction levels such as requirements, designs, components, test cases *etc.* These artifacts are constructed, shared and maintained among product line members, namely products in the product line, as *core assets*. Reuse in the SPL paradigm is not opportunistic and bottom-up, rather well-planned and top-down (or architecture centric).

SPL development consists of two parallel series of development processes: *domain engineering* and *application engineering*. Domain engineering is a series of processes to construct core assets. Application engineering is a series of processes to reuse the core assets in a prescribed manner and derive product line members. Domain engineering requires more costly and complicated construction works of reusable artifacts than single product development, while application engineering becomes costless reuse works. In the best scenario, the application engineer just selects or deselects features for his/her product; composes and configures the artifacts in the core assets based on the selected features in a prescribed way; and derive the product. Therefore, SPL pays off if we develop a number of similar but different product variants.

Commonality and variability modeling is a key concept in SPL. Feature modeling with the feature diagram [6] is a *de-facto* standard technique for this purpose. The feature diagram is a tree form diagram that visualizes commonality and variability of the product line in terms of the features equipped by its members. Features are categorized based on their reusability in different members of the product line. In the original feature diagram, the feature is categorized into mandatory, optional, alternative, and or-features. So far, the feature diagram has been extended to represent more complicated variability. The orthogonal variability model (OVM) [10] employed in this work is an instance.

In principle, application engineering must be performed at inexpensive cost within a sufficiently shorter period to earn return more than investment for the core asset. However, if the product line has a number of features and their relations are too complicated, it is often tough and time-consuming to find an optimal selection and configuration of features with satisfying both product requirements and constraints.

Therefore, the authors present an integer linear programming (or *ILP* for short) [1] based decision making framework using the feature model to help application engineers in feature selection and configuration activity. The framework provides a formulation and a process to find an optimal feature selection and configuration with the ILP to derive a preferable product of the product line. We use ILP as a method for optimization problem in feature selection. The work in this paper is an extension to the authors' previous work [3].

The remainder of this paper is organized as follows: Section 2 explains the variability modeling in SPL. Section 3 gives the related works. Section 4 proposes the framework and presents evaluation using real data. Finally, Section 5 concludes this paper.

2. Variability Modeling

Commonality and variability modeling is an important activity in SPL development. Feature modeling (FM) [6] is a well-known and widely used technique to represent commonality and variability of the product line. In feature modeling, commonality and variability among members of the product line are described in terms of the *features* that each member of the product line equips. The feature is a prominent or distinctive concepts or characteristics that are visible to various stakeholders of the product line [8]. Therefore, each distinctive function in the product line can be a single feature.

In feature modeling, features are organized in a tree form. Each tree edge of the model means: i) the parent feature consists of the child feature(s); ii) the parent feature generalizes the child

†1,2,3,4 Kyushu University

feature(s); or iii) the parent feature is implemented by the child feature(s). Moreover, each feature can be classified into four kinds of variability classification: *mandatory*, *optional*, *alternative* and *or*. A mandatory feature is one equipped by each product line member if its parent feature is equipped in the member. An optional feature is one equipped optionally by each product line member if its parent feature is equipped in the member. An alternative feature is one equipped alternatively (of other features in the set) by each product line member if its parent feature is equipped in the member. An or feature is a set of features such that each product line member equips one or more out of them.

In order to represent variability more powerfully, the authors introduce notation of the orthogonal variability model (or *OVM* for short) [10] in the traditional feature diagram. In OVM notation, variability is represented by the variation point and possible variants bound to the variation point as shown in Figure 1. The solid edge between the variation point and the variant means the variant must be bound to the variation point, if the variation point is active in the product. The dotted edge between the variation point and the variant means the variant may or may not be bound to the variation point, if the variation point is included in the product. The multiplicity annotation [*min..max*] of the variation point means the variation point can bind at least *min* and at maximum *max* variants. Therefore, OVM of Figure 1 means the product can have one to three power sources out of batteries, solar cells, wind power generators and the battery power source is mandatory.

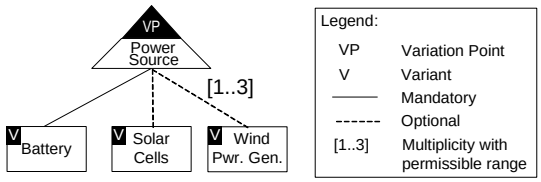
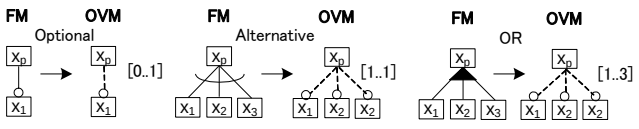


Figure 1: Example of Orthogonal Variability Modeling

In this paper the authors introduce the multiplicity concept of OVM into the traditional feature model with keeping the optional feature representation by circular decoration and three kinds of feature relationship representation (composition, generalization and implemented-by) of the original feature diagram. Note that feature selectability of the traditional feature diagram can be equivalently represented as below:



3. Related Works

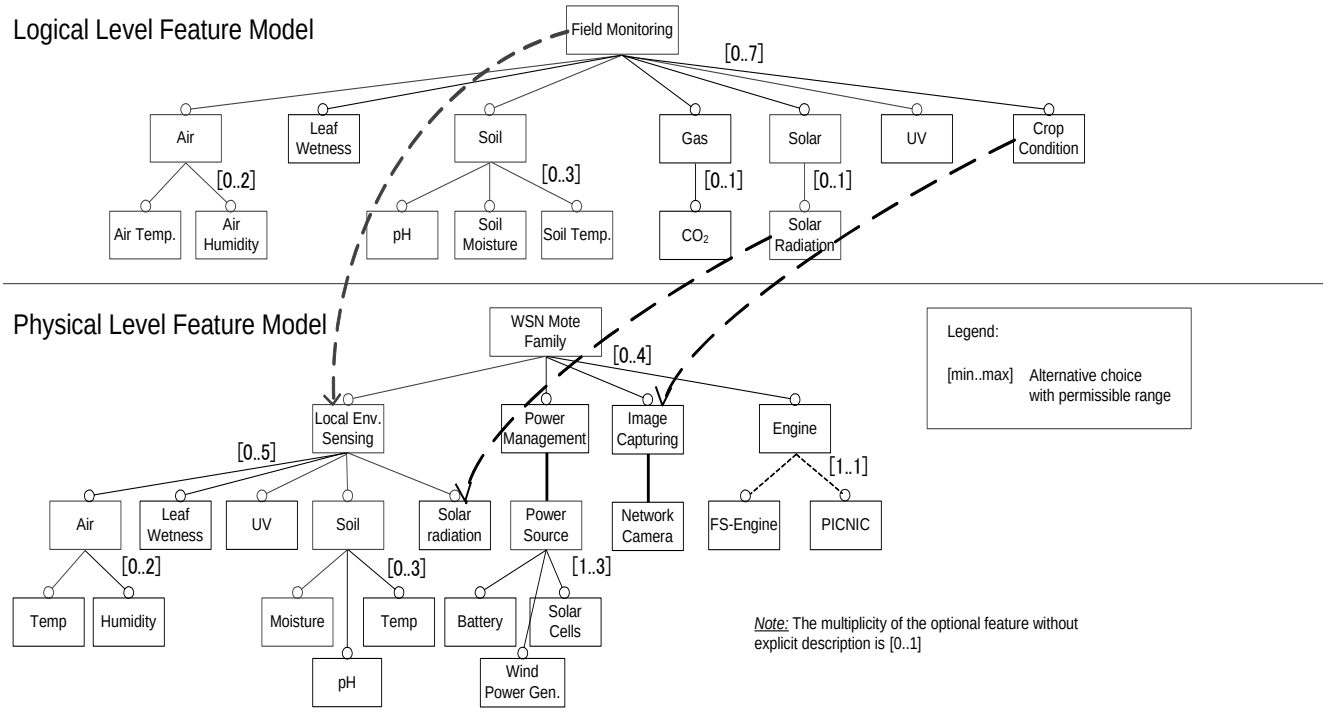
Products derivation can be performed using two kinds of process. First is configuration process, by selecting desired features directly from the feature model appropriate with user's requirements. And the second one is specialization process, by performing specialization of the feature model iteratively until a specialized feature model that represent the user requirements is obtained. Specialization process takes a feature diagram and yields another feature diagram [2].

Reference [6] presents a method of finding a valid feature set by following the four level feature hierarchy, *i.e.* first considering capabilities, then operating environments, and finally domain technologies and implementation techniques, because this hierarchy corresponds to increasing level of details in the artifact space, and therefore, reflects stepwise refinement. In [11] researchers proposed COVAMOF, is a framework facilitating product derivation using four steps: product definition, product configuration, product realization, and product testing. These steps are performed iteratively until a final configuration meets with user's requirements. In order to select the right features and components, the engineer binds variation points in the feature layer. However, product configuration in the COVAMOV involves a very complicated task. While reference [2], proposes staged configuration to find a form of configuration achieved by successive specialization followed by deriving a configuration from the most specialized feature diagram in the specialization sequence. The process of specifying a family member is performed in stages, where each stage eliminates some configuration choices and yields a specialized feature model. This method is a useful and important mechanism for software supply chains based on product lines. Another method is presented in Gears [7], where the domain engineers create and declare a global model of a product line and application engineers create the definition of the product by selecting a value for each of the feature declarations from the previous step.

However, in the studies, they are focus on satisfying the functional requirements without considering non-functional requirements. There is no clear mechanism of how to decide selected features have satisfied user's requirements and considered constraint aspects of the system.

In this study, we provide a systematic process for configuration and apply optimization technique to find the optimal features with satisfying user's requirements and constraints aspects or non-functional requirements of the system.

4. An Integer Linear Programming Based Decision Making Framework for Application Engineering



4.1 Problem Formulation

ILP [1] is a class of the maximization or minimization problem to find a value assignment to a set of integer variables, denoted by $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, subject to a set of linear equalities or inequalities of $\boldsymbol{x}$, denoted by $\boldsymbol{p}(\boldsymbol{x}) = \{p_1(x_1, x_2, \ldots, x_n), p_2(x_1, x_2, \ldots, x_n), \ldots, p_m(x_1, x_2, \ldots, x_n)\}$, such that maximizes or minimizes the given linear objective function of $\boldsymbol{x}$, denoted by $f_{obj}(\boldsymbol{x}) = f_{\text{obj}}(x_1, x_2, \ldots, x_n)$.

In application engineering, the engineer selects or de-selects the features required for the product to be derived on the feature diagram. The features which the engineer does not select nor de-select should be selected or de-selected with keeping the constraints posed by the product and its environment to maximize preferable properties.

Let us denote the set of all the features by F. For each feature $f \in F$, the decision making framework that the authors propose defines one binary integer variable x_f to represent if f is selected or de-selected; that is, 1 or 0 is assigned to x_f if f is selected or de-selected, respectively. These variables become ones of the ILP.

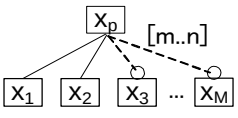
Non-functional properties such as cost, reliability, operating time, power consumption, *etc.* are dependent on the feature selection. In the decision making framework, these properties must be represented by linear functions of $\{x_f\}$ $(f \in F)$. Let us denote the function on quality q by $QF_q(\{x_f: f \in F\})$. The quality function may become an objective function or a constraint of the ILP, depending on the product line. If a quality should be maximized or minimized for the product, its quality function is used as the objective function of the ILP. If a quality should be at a certain value or within a certain range, its quality function should be used as a constraint equality or inequality of the ILP.

4.2 The Application Engineering Process of the ILP Based Decision Making

In this section the authors present the application engineering process of the proposing ILP based decision making framework.

Step 1: The application engineer defines constrains on feature selection by linear equalities and inequalities of $\{x_f: f \in F\}$.

The constraints can be defined in a straightforward manner according to the feature relationship and the multiplicity on feature selection. For any feature having child features shown in the fragment of the feature diagram with OVM multiplicity notation:



we define the following constraint equalities and inequalities:

$$x_{f_i} = x_{f_p} \quad \text{(for any mandatory feature } f_i\text{)}$$

$$\text{and } \mathrm{m} \times x_{f_p} \leq \sum_{i=1}^{M} x_{f_i} \leq n \times x_{f_p}.$$

Moreover, for any require dependency $f_p \rightarrow f_q$, mutually inclusive dependency $f_p \leftrightarrow f_q$, and mutually exclusive dependency $f_p \leftrightarrow f_q$ in the feature diagram, we define the following constraint equalities:

$$x_{f_p} \leq x_{f_q} \quad \text{(for require dependency } \quad f_p \rightarrow f_q\text{)}$$

$$x_{f_p} = x_{f_q} \quad \text{(for mutually inclusive dependency } \quad f_p \leftrightarrow f_q\text{), and}$$

$$0 \leq x_{f_p} + x_{f_q} \leq 1 \quad \text{(for mutually exclusive dependency } \quad f_p \leftrightarrow f_q\text{),}$$

respectively.

Step 2: The application engineer assigns 1 or 0 based on the feature selection for the product to be derived. 1 is assigned to x_f if the product needs a feature f, or 0 is assigned otherwise.

Step 3: The application engineer defines non-functional properties of the product, which will be optimized or constrained in application engineering, as linear functions of $\{x_f: f \in F\}$.

Step 4: The application engineer specifies constraints of the ILP as inequalities of the linear functions defined in Step 3.

Step 5: The application engineer adopts the linear function defined in Step 3 for the preferable property of the product as the objective function of the ILP.

Step 6: The application engineer solves the ILP. The ILP solver gives an optimized configuration for the suspended features subject to the feature selection and non-functional properties specified in Step 4.

4.3 Evaluation

The authors applied the proposed framework to the agricultural sensor network product line using FieldServer [4].

Figure 2 shows the feature diagram with OVM multiplicity notation for the product line. We define a binary integer variable representing if feature is selected or not for each feature. Moreover, we define linear equalities and inequalities representing constraints on feature selection as shown in Step 1 of the previous section. For example, we can define the following equality and inequality for the feature *power source*.

$$x_{\text{powermanagement}} = x_{\text{power source}},$$

$$x_{\text{power source}} \leq x_{\text{battery}} + x_{\text{solar cells}} + x_{\text{wind power gen.}} \leq 3x_{\text{power source}}.$$

Let us suppose the product to be derived needs features *WSN Mote Family*/*Local Env. Sensing*/{*Air*/{*Temp*, *Humidity*}, *Soil*/*Temp*, *Power Management*/*Power Source*/*Battery*} and does not need *WSN Mote Family*/{*Local Env. Sensing*/{*UV*, *Solar Radiation*}, *Image Capturing*}. Therefore, the variables corresponding to these features should be bound as follows:

$x_f = 1$ if f is *WSN Mote Family*/*Local Env. Sensing*/{*Air*/{*Temp*, *Humidity*}, *Soil*/*Temp*,*Power Management*/*Power Source*/*Battery*}

$x_f = 0$ if f is *WSN Mote Family*/{*Local Env. Sensing*/{*UV*, *Solar Radiation*},*Image Capturing*}

The other variables corresponding to the suspended features are not be bound at this step.

Cost is the most important concern for the agricultural sensor network product line. Based on data shown in Reference [4], we define linear functions of $\{x_f\}$ estimating the cost of each node denoted by Q_{Cost}.

Since the cost must be less than or equal to US$400, we add a non-functional constraint $Q_{\text{Cost}} \leq$ US$400 to the constraints of the ILP. At the same time, the product is wanted to equip more functions in order to execute unknown applications in future. Therefore, we adapt $Q_{\text{Cost}} \rightarrow \max$ as the objective function of the ILP.

We solve the above problem using LPSolve IDE [5] and obtained an optimal configuration such that the suspended features *WSN Mote Family*/{*Local Env. Sensing*/{*Air*/*Leaf Wetness*, *Soil*/{*Moisture*, *pH*}}, *Power Management*/*Power Source*/*Battery, Engine*/*PICNIC*} are selected.

5. Concluding Remarks

In this paper, the authors have presented an expressive way for configuration process in SPL. The proposed framework reduces the problem to find the best set of features satisfying requirements and constraints forced by the product to the ILP. In our case study of the agricultural wireless sensor network product line, we obtained an optimal configuration within the specified cost and power consumption. The proposed framework can be used to prepare best configurations for a new product or to optimize existing configuration for maintenance purposes.

Future works include further evaluation of the proposed framework with various non-functional requirements such as computation time and communication overheads.

Reference

1) D.-S. Chen, R.G. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*, John Willey & Son, 2010.

2) K. Czarnecki, S. Helsen, and U. Eisenecker, "Staged Configuration Using Feature Models, " *Proc.Software Product Line Conf. 2004*, Springer, pp.266-283, 2004.

3) M. Fajar, T. Nakanishi, K. Hisazumi, and A. Fukuda, "A Decision Making Framework for Developing Agricultural Wireless Sensor Network Systems," *Proc. 8th Asian Conf. for Information Technology in Agriculture (AFITA)*, Sep. 2012 (to appear)

4) T. Fukatsu and M.Hirafuji, "Field Monitoring Using Sensor-Nodes with a Web Server," *J. of Robotics and Mechatronics*, Vol.17 No.2, pp. 164-172, 2005.

5) H. Gourvest, W.Patton, and P. Notebaert, Linear Programming Solve (LPSolve IDE), available athttp://sourceforge.net/projects/lpsolve/files/lpsolve/5.5.2.0/

6) K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," *Technical Report CMU/SEI-90-TR-21*, SEI/CMU, 1990.

7) C.W.Krueger, "Software Mass Customization," white paper, 2001. available at http://www.biglever.com/learn/whitepapers.html

8) K.Lee, K.C. Kang, and J. Lee, "Concepts and Guidelines of Feature, Modeling for Product Line Software Engineering,"*Proc. 7thInt. Conf. on Software Reuse (ICSR)*, pp.62-77, 2002.

9) L. Northrop and P. Clements, *Software Product Lines: Practices and Patterns*,Addison-Wesley, 2001.

10) K.Pohl, G. Bockle, and F. v. d. Linden, *Software Product Line Engineering: Foundations, Principles, and Techniques*,Springer, 2005.

11) M.Sinnema, S. Deelstra, and P.Hoekstra, "The COVAMOF Derivation Process," *Proc. 9th Int. Conf. on Reuse of Off-the-Shelf Components*, pp. 101-114, 2006.