Modeling and Analysis of Behavioral Variability in Product Lines

Hongxia Zhang\textsuperscript{a,b,*}, Hua Zou\textsuperscript{a}, Fangchun Yang\textsuperscript{a}, Rongheng Lin\textsuperscript{a}

\textsuperscript{a}State Key Laboratory of Networking and Switching, Beijing University of Posts and Telecommunications, Beijing 100876, China
\textsuperscript{b}School of Computer and Communication, China University of Petroleum, Qingdao 266580, China

Abstract

Software product lines have been proven to be an effective way to develop domain-oriented software systems efficiently. In product line engineering, formal modeling and verification are critical for managing the behavioral variability since the number of products can be exponential. Therefore, two major challenges are the scalable modeling and the efficient verification of system behaviors. In this paper, we propose a behavioral model, namely Adaptable Featured Petri Net (AFPN) based on analysis of variability mechanism for the change of requirements. In particular, an AFPN is able to define a unified behavior model for a family of products by leveraging on establishing relationships between features and transitions. And transition priorities are considered in. Then, the derivation algorithm is presented to obtain the behavior model of a particular product. Based on this model and algorithm, verification behavior properties of AFPN and products can be carried out by means of automatic tools. Finally, we apply our approach to a vending machine product family, which shows that our approach helps to modeling and analyzing product families effectively in the early phases of development.

Keywords: Software Product Lines; Features; Behavioral Models; Petri Nets; Variability

1 Introduction

With the command of innovation, increasingly shorter time-to-market and efficiency in business environments, the need to tailor software applications to specific requirements is growing. If each application variant is maintained individually, the management of all variants quickly becomes infeasible [1]. Software Produce Line Engineering (SPLE) is considered as a novel solution.

A Software Product Line (SPL) is traditionally defined as “a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market...
segment or mission and that are developed from a common set of core assets in a prescribed way.” [2] Feature Diagrams (FDs) are typically used to modeling commonality and variability in product families functions.

Many notations and description techniques have been recently proposed for the behavioral variability modeling of product line development, such as Model I/O automata [3], Modal Transition System and variants [4-8], PL-CCS [9], Petri net [10], etc, which only consider variability modeling of SPL behavior through introducing variable operators on transitions. However, these works ignore the relationship between feature models and behavior models. Moreover, they do not provide methods for verifying behaviors rely on features or products. Featured Transition System (FTS) [19] is proposed to model variable behaviors based on feature sets, which overcome problems mentioned above, however the corresponding verification algorithms should also be proposed, which is complex for engineers.

To address these challenges, we recently launch a research effort to, on the one hand, investigate the most promising existing modeling structures that allow to describe behavioral variability, meanwhile provide a solid formal basis for modeling and verification, on the other hand, we try to find a proper method to realize the relationship between feature models and behavior models. Our aim is the development of rigorous modeling techniques as well as analysis and verification approach for SPLs.

In this paper, we analyze characteristics of requirement changes firstly. In order to manage the variability of SPLs effectively, we propose Adaptable Featured Petri Net (AFPN), which specifies variable behaviors and resource usages in SPLs based on Petri Nets (PNs), and constructs the relationship between features and transitions. Then a derivation algorithm from feature sets to a behavior model is proposed which permits to obtain the behavior of each product of the SPL through features. Finally, we apply our approach to model and analyze the behavior model of a vending machine product family.

The reminder of the paper is organized as follows. Section 2 presents a vending machine case study that we use to show the related issues of the proposed approach and recalls the necessary background on FDs and PNs. We formally define AFPNs and project algorithm in Section 3. In Section 4, we discuss how to verification the behavior model of a product family and products. Section 5 surveys related work. Finally, in Section 6 we conclude the paper.

2 Background

2.1 Motivating Example

For easy comparison with others, we use a beverage vending machine, which is widely used in other articles, as a running example. Beverage vending machine is a family of (simplified) beverage machines, for which we consider the following requirements in the basic version:

(1) The vending machine is a charge or a free of charge machine exclusively.

(2) After paying or press free button, the user has to select either soda or tea. The choice of beverages (soda, tea) varies between the products.

(3) After delivering the beverage, optionally, a ring is rung. However, a ring must be rung
whenever a tea is delivered.

(4) The machine returns to its idle state when the beverage is taken by the user. For charge machine, there is a compartment, which will be open so that the user can take her beverage, before it closes again.

This list contains both static requirements, which identify the features that constitute the different products (see requirements 1, 4, and partially, 2 and 4) and behavioral requirements, which describe the admitted sequences of operations (requirements 2, 4 and partially, 3). Therefore, we will first discuss the feature model of the above family and provide a formal representation. Then, we will show how behavioral requirements of the family can be described using an adaptable featured petri net. Finally, we will show how to combine two approaches and verify the family. We recall basic concepts and definitions of feature models and Petri nets that will be used throughout the rest of the paper in the rest of this section.

2.2 Feature Model

Feature Model is the first level abstraction of requirements, which provides help for understanding stakeholders’ requirements effectively. Features have been defined as “a distinguishable characteristic of a concept (e.g., system, component, and so on) that is relevant to some stakeholder of the concept” [11]. In the basic form, a feature model is a tree of features which are divided in grouped feature and solitary feature; Types of solitary feature contain mandatory (e.g., beverages) and optional (e.g., ring). The relationship of a grouped feature includes and-groups or-groups (e.g. soda or tea) and xor-groups (e.g. pay xor free). The relationship between features includes imply and exclude relationships. Here, we only focus on the function and configured products described by feature model. Skipping the details, we informally recall the definition of FDs as below.

**Definition 1 (Feature Diagrams)** A feature diagram is a 3-tuple $FD = (F, r, DE)$, where $F$ is a set of features, $r$ is the root which is a mandatory feature, $DE \subseteq D \times F$ is the set of decomposition edges between features.

A complete formal definition of FDs can be found in [12].

The semantics of a FD $d$, denoted $[d]_{FD}$, is its valid feature groups, named products, i.e., a set of features: $[d]_{FD} \subseteq P(F)$. Based on the above requirements and the definition of feature diagram, the FD of the vending machine is obtained in Fig. 1. For convenience, we use the character next to solidus (/) to express the feature.

![Fig. 1: The feature diagram of the vending machine](image-url)

Based on configuration methods of a feature model [13], the semantics of the vending machine FD from Fig.1 is obtained, which contain 16 products, that is, \{v, b, s, pc, p\}, \{v, b, t, pc, p, r\},
\{v, b, s, t, pc, p, r\}, \{v, b, s, pc, f\}, \{v, b, t, pc, f, r\}, \{v, b, s, t, pc, f, r\}, \{v, b, s, pc, p, r\}, \{v, b, s, pc, f, r\}, \{v, b, s, pc, p, c\}, \{v, b, t, pc, p, r, c\}, \{v, b, s, t, pc, p, r, c\}, \{v, b, s, pc, p, r, c\}, \{v, b, s, pc, f, r, c\}, \{v, b, s, pc, p, c\}, \{v, b, t, pc, p, r, c\}, \{v, b, s, t, pc, f, r, c\}, \{v, b, s, pc, f, c\}, \{v, b, t, pc, f, r, c\}, \{v, b, s, t, pc, f, r, c\}, \{v, b, s, pc, f, r, c\}.

\section{Petri Nets}

Since we want to address the behavioral aspects of a product family, we will base on the concept of PN [14], which provide a solid formal basis for system modeling. They have been studied and applied widely, and come with a wealth of formal analysis and verification techniques.

\textbf{Definition 2 (Petri Net)} A Petri Net is a 4-tuple $PN = (P, T, R, M_0)$, where

1) $P = \{p_1, p_2, ..., p_n\}$ $(n \geq 0)$ is a set of places, $T = \{t_1, t_2, ..., t_m\}$ $(m \geq 0)$ is a set of transition, $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$;

2) $R \subset (P \times T) \cup (T \times P)$ is a set of arcs;

3) $\text{dom}(F) \cup \text{cod}(F) = P \cup T$, where, $\text{dom}(F) = \{x \in P \cup T \mid \exists y \in P \cup T : (x, y) \in F\}$, $\text{dom}(F) = \{x \in P \cup T \mid \exists y \in P \cup T : (y, x) \in F\}$;

4) $M_0 : P \rightarrow \mathbb{N} \cup 0$ is called an initial marking of petri net.

Each of these products has its behavior model, such as, in its basic version, the behavior model of vending machine which only provides soda for charge $\{v, b, s, pc, p\}$ (called “pt1”) is shown in Fig. 2 (a). Another variant which provides soda and tea for free $\{v, b, s, t, pc, f\}$ (called “pt2”) is shown in Fig. 2 (b). And the behavior of providing tea for charge with ringing and cancel function $\{v, b, t, pc, p, r, c\}$ (called “pt3”) is shown in Fig. 2 (c).

\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\centering
\begin{tikzpicture}[node distance=2cm, auto]
  \node (p) [circle, draw, initial, initial text=Pay] {$\ast$};
  \node (c) [circle, draw, right of=p] {Choose soda};
  \node (s) [circle, draw, right of=c] {Serve soda};
  \node (o) [circle, draw, right of=s] {Open};
  \node (c1) [circle, draw, below of=s] {Close};
  \draw[->] (p) -- (c);
  \draw[->] (c) -- (s);
  \draw[->] (s) -- (o);
  \draw[->] (c1) -- (p);
\end{tikzpicture}
\caption{The behavior model of the product pt1}
\end{subfigure}
\begin{subfigure}{0.45\textwidth}
\centering
\begin{tikzpicture}[node distance=2cm, auto]
  \node (p) [circle, draw, initial, initial text=Free] {$\ast$};
  \node (c) [circle, draw, right of=p] {Choose tea};
  \node (s) [circle, draw, right of=c] {Serve tea};
  \node (c1) [circle, draw, below of=s] {Close};
  \node (r) [circle, draw, right of=s] {Ring};
  \node (o) [circle, draw, right of=r] {Open};
  \draw[->] (p) -- (c);
  \draw[->] (c) -- (s);
  \draw[->] (s) -- (r);
  \draw[->] (r) -- (o);
  \draw[->] (c1) -- (p);
\end{tikzpicture}
\caption{The behavior model of the product pt2}
\end{subfigure}
\begin{subfigure}{0.45\textwidth}
\centering
\begin{tikzpicture}[node distance=2cm, auto]
  \node (p) [circle, draw, initial, initial text=Pay back] {$\ast$};
  \node (c) [circle, draw, right of=p] {Choose tea};
  \node (s) [circle, draw, right of=c] {Serve tea};
  \node (c1) [circle, draw, below of=s] {Close};
  \node (r) [circle, draw, right of=s] {Ring};
  \node (o) [circle, draw, right of=r] {Open};
  \draw[->] (p) -- (c);
  \draw[->] (c) -- (s);
  \draw[->] (s) -- (r);
  \draw[->] (r) -- (o);
  \draw[->] (c1) -- (p);
\end{tikzpicture}\caption{The behavior model of the product pt3}
\end{subfigure}
\caption{The behavior model of variants of the vending machine}
\end{figure}

Through analyzing, we found that, behaviors of these products have not only distinctions but also some similarities, and modeling behavior for each product will be time-consuming and laborious. Therefore, we will discuss unified behavior modeling for a whole product family.
3 Behavior Models for Product Families

In order to describe the behavior model of product families concisely, we propose Adaptable Featured Petri Nets, which is an extension of petri nets by introducing the features. We will formalize the variability of behaviors by means of combining features with transitions in a petri net, and verification of model will discuss in Section 4.

3.1 Adaptable Featured Petri Net

The purpose of Adaptable Featured Petri Nets (AFPNs) is to model the behaviors of the whole product families. This means that the AFPN of a product family has to accommodate all the possibilities desired for each derivable product, predication on the choices that make a product belong to that family.

Moreover, features are often used for compact representations of a family’s products. To model behaviors of product family, a “translation” from features to behaviors is needed. Each transition is labeled with a feature. Besides, a priority relation over alternative transitions was introduced. Then, the definition of Adaptable Featured Petri Net is given, which define the relationship between features and transitions and priorities between transitions.

**Definition 3 (Adaptable Featured Petri Net)** An Adaptable Featured Petri Net is a 4-tuple $\text{AFPN} = (PN, FD, \gamma, \succ)$, where

1) $PN = (P, T, R, M_0)$ is a petri net;
2) $FD = (F, r, DE)$ is a feature diagram;
3) $\gamma : T \to F$ is a total function, labeling transitions with features;
4) $\succ \subseteq T \times T$ is a partial order, defining priorities between transitions.

The semantics of an AFPN afpn, denoted, is the set of its valid behavior model. Based on the definition, the AFPN of the vending machine example is given in Fig. 3, where the feature label of a transition is shown next to its action label, separated by a solidus (/).

![Fig. 3: The feature diagram of the vending machine](image)

AFPN is a behavior model of a product family, which become complicated for accommodating all the possibilities. This will cause conflicts between transitions. For example, in the behavior...
model of vending machine family (Fig. 3), Open and Skip are alternative transitions which cause the execution in marking $P_6$ is indeterminate when features vending machine and Free are all selected. For this reason, we define transition priorities which offer an intuitive way to model cases in which one transition overrides the behavior of another. Here, we define transition Skip has priority over transition Open, this means that products containing both Free and Vending machine only have transition Skip. Therefore, AFPN model constraints by explicitly referring to features and priorities.

Different from Petri Nets, transitions are constraint by features and priorities change dynamic properties in AFPNs. Then, we discuss behavior properties of AFPNs for a given product according to a feature selection which characterize the behavior of AFPN.

**Definition 4 (Enabled)** Given an AFPN $afpn = (P, T, R, M_0, FE, \gamma, \triangleright)$ and a product $pt \in \langle d \rangle_{FD}$, a transition $t$ is enabled in a state with marking $M$ iff $\forall p \in p \cdot t \cdot \gamma (t) \in pt : (M(p).pt) \geq \bullet (t)\mid .$

In the above definition, the state of the AFPN is denoted by a tuple consisting of a marking $M$, the behavior of AFPN exhibits by passing through a sequence of markings $M_0 \cdot M_1, \ldots, M_{k+1}$, each change of marking is triggered by a transitions occurrence $(m_i, pt)[t_i \cdot (M_{i+1}.pt)],$ is called a trace over $pt$, that is, $\exists (M_0, pt), (M_1, pt), \ldots, (M_{k+1}, pt): (M_0, pt) \xrightarrow{t_1} (M_1, pt) \xrightarrow{t_2} \cdots \xrightarrow{t_k} (M_{k+1}, pt)$ written as $(M_0, pt)[\sigma > (M_{k+1}, pt),$ where $\sigma$ is the transition sequence $t_1, t_2, \ldots, t_k$.

Given an AFPN, the set of traces for all products represent the behavior of a product family, denoted as $\lbrack afpn \rbrack_{AFPN}$. And the behavior of a particular product is a set of trace related to the product.

**Definition 7 (Product Behavior)** Given an AFPN $afpn = (P, T, R, M_0, FE, \gamma, \triangleright)$ and a product $pt \in \langle d \rangle_{FD}$ the behavior of $afpn$ over $pt$ is the set of all traces over $pt$ from the initial marking $M_0$, noted as $afpn (pt)$.

Each product behavior of the SPL is a subset of the family behavior, and the semantics of an AFPN is thus the union of the behaviors of all valid products.
Definition 8 (Semantics of an AFPN) Given an AFPN \( afpn = (P, T, R, M_0, FE, \gamma, \succ) \) the semantic of \( afpn \) is union of the set of all valid product behaviors.

\[
[afpn]_{AFPN} = \bigcup_{pt \in [d]_{FD}} afpn(pt)
\]

Through this way, the behavior of product families can be formally modeled by AFPN. An important observation is that, except for trivial cases, the AFPN semantics we just defined is not equal to the ordinary PN semantics. Formally, there exists an AFPN \( afpn \) for which \( [afpn]_{AFPN} \neq [PN(afpn)]_{PN} \), where \( PN(afpn) \) is the PN obtained by removing \( FD, r \) and \( \succ \) from \( afpn \). For example, in the AFPN of vending machine SPL, when the customer insert a coin ask a cup of tea, sometimes the machine ring and sometimes not. Both of traces are accepted by \( PN(afpn) \), however, the trace with skip ring is not the behavior of \( afpn \). The reason is that \( afpn \) is a PN with feature and priority constraints. Moreover, we have the following theorem.

Lemma 1 (Relationships Between AFPN and PN Semantics)

\[
[afpn]_{AFPN} \subseteq [PN(afpn)]_{PN}
\]

This theorem illustrates that the classical analysis method on PN cannot be simply used on an AFPN for the SPL directly. While this verification might be sound, it is not always complete: by ignoring feature and priority constraints, it would find counter-example. Therefore, though analyzing, we find two ways to analyze the AFPN for the SPL. The first way is presenting a new analytical method which we will do in the future work, and another way is analyzing all product behaviors of the SPL which may cause the state-explosion problem, but for small systems, it is effective and simply. In this paper, we adopt the second method. Besides, the behavior of a particular product is also needs for developers. In the next section, we will discuss how to obtain the product behavior model through AFPN of a product family.

3.2 Derivation Product Behavior from AFPN

It is a critical issue for business developers to acquire the behavior model based on a configured feature model, which will greatly reduce development time and cost. From the AFPN, one can obtain the behavior of a particular product through projection. Intuitively, the diagrams of Fig. 2 can be obtained by removing selected transitions and states from Fig. 3.

Formally, in order to obtain the behavior of one particular product, one projects the AFPN on the corresponding set of features, namely \( pt \in [d]_{FD} \). This transformation is entirely syntactical and consists in removing (i) all transitions linked to features that are not in \( pt \), (ii) all transitions that are overridden by higher priority transitions, and (iii) all states that are isolated. The result of the projection is an ordinary PN. The specific mapping is shown below.

Definition 9 (Projection) The projection of an AFPN \( afpn = (P, T, R, M_0, FE, \gamma, \succ) \) to a product \( pt \in [d]_{FD} \) denoted as \( afpn|_{pt} \) is the PN \( pn = (P', T', R', M_0) \), where \( T' = \{ t \in T \mid \gamma(t') \in pt \land (t', t) \in \succ \} \), \( P' = \{ p \in P \mid p \in \bullet t \lor t \in T' \} \), \( R' = \{ (p, t) \in R \mid p \in P' \land t \in T' \} \cup \{ (t, p) \in R \mid t \in T' \land p \in P' \} \).

Through projection, the behavior model of a particular product is obtained which is a classical petri net, it coincides with the definition of product behavior, as stated by the following theorem.
Lemma 2 Given an Adaptable Featured Petri Net \(afpn = (P, T, R, M_0, F E, \gamma, >)\) and \(pt \in \llbracket d \rrbracket_{FD}\) \nafpn(pt) = \afpn\mid pt

Proof Firstly, the initial markings in \(afpn\mid pt\) is the same as \(afpn\mid pt\), which is the initial marking in the AFPN. The initial marking \(M_0\) coincide in both petri nets.

Then, we discuss traces in both behavior models.

(\subseteq) We show that every trace \(\sigma \in afpn\mid pt\) is also a trace in \(afpn\mid pt\) at first. If \((M_i, pt)[t > (M_{i+1}, pt)\) in \(afpn\mid pt\), then feature constraints and priority constraints in occurrence rule (Definition 5) is satisfied, which is the same as the transition project conditions in Definition 9, the transition \(t\) is also in \(afpn\mid pt\). Hence, \(M_i[t > M_{i+1}\).

(\supseteq) Following a similar reasoning as before, we show that every trace \(\sigma \in afpn\mid pt\) is also a trace in \(afpn\mid pt\). Observe that, if \(M_i[t > M_{i+1}\), then \(t\) is a transition of \(afpn\mid pt\) which satisfies \(\gamma(t) \in pt \land \gamma(t') \in pt \land (t', t) \in >\), it is the same as the feature constraint and priority constraint in Definition 5, that is transition \(t\) is also in \(afpn\mid pt\). Hence, we conclude that \((M_i, pt)[t > (M_{i+1}, pt)\).

As we just showed, the projection of an AFPN for a product is equal to the product behavior as defined in Definition 9, that is to say, the AFPN behavior is also equal to the union of projections of the AFPN for all products. Therefore, the result of an AFPN analysis is the same as the result of all product behaviors. Then, we discuss the derivation steps.

A projection steps derivation() can be applied to obtain a behavior model from the product family model AFPN:

Step 1. Configure the feature model based on stakeholder’s requirements, then, acquire the valid feature set, that is product \(pt\);

Step 2. Obtain the transition sets related to features in product, \(trans_{pt} = \{t \in T \mid (\forall t)\gamma(t) \subseteq pt\}\);

Step 3. Traversing the AFPN, terminate traversing when a transition is not in \(trans_{pt}\), delete it and trace back to the previous place node, continue traversing;

Step 4. Determining whether transitions satisfy priority constraints, if not, delete the lower-priority transitions;

Step 5. Delete isolated places, that is, without predecessors and successors, satisfies \(\bullet p \cup p \bullet = \emptyset\).

Based on derivation(), the behavior model of each particular product is obtained.

4 Analysis of Product Behavior Models

The previous sections provide a mathematical foundation for behavior modeling of the whole SPL, and define the mapping method through which the behavior of a product is obtained. In this section, the verification of ADPN and PN for a certain product will be discussed.

Model analysis aims at checking the constructed model for system inconsistencies and deriving statements on structural and dynamic system properties, which reflect correctly the behavior of
the system in reality. In the following we introduce those model properties and apply them to our case study.

The three orthogonal dynamic properties of a Petri net are liveness, reversibility and boundness. A transition is dead at a given system state, if no state can be reached anymore, where the transition is enabled. A transition is live at a given state, if no state can be reached, where the transition is dead. A net is live, if all its transitions are live in the initial marking. A net is reversible, if the initial system state can be reached again from each reachable state. A net is bounded, if there is a positive integer number, k, which represents an upper bound for the number of tokens on each place in all states of the net. If the net is bounded in every initial marking, it is said to be structurally bounded. If a net is bounded, the number of reachable states is finite.

In this paper, we use PIPEv4 [15] to verify the dynamic properties of product behaviors. We take a particular product (pt3 in Section 3) for example, analyze its dynamic properties. Through analyzing the requirement, the feature set is vending machine, beverages, tea, purchase, pay, ring, cancel. Based on the mapping method and BFS algorithm, the behavior model of the product can be obtained from the AFPN, which is shown in Fig. 2 (c).

Firstly, we use invariant analysis method to analyze the behavior model for the product. T-invariant is calculated and the result is shown in Table 1, which demonstrates that the model is covered by positive T-invariant, therefore it might be bounded and liveness. And the result of P-invariant is \((P_0, P_1, P_2, P_4, P_5, P_6, P_7) = (1, 1, 1, 1, 1, 1, 1)\), which demonstrates that the model is covered by positive P-invariant, therefore the model is sound, and its P-invariant equation is \(M(P_0) + M(P_1) + M(P_2) + M(P_4) + M(P_5) + M(P_6) + M(P_7) = 0\).

<table>
<thead>
<tr>
<th>Cancel</th>
<th>Choose tea</th>
<th>Close</th>
<th>Open</th>
<th>Pay</th>
<th>Return</th>
<th>Ring</th>
<th>Server tea</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Then, we analyze the liveness of the AFPN, which is based on the construction of the reachability graph or just the set of reachable states of the net. The result depends generally on the initial marking. Due to space constraints and possible state space explosion problem, we calculate the reachability graph with the initial marking only contains one token, and 7-tuple \((P_0, P_1, P_2, P_4, P_5, P_6)\) is used to express each reachable state, the reachability graph is shown in Fig. 4.

Analyzing the reachability graph shows that all states are reachable, therefore the model is reachability; the model is liveness for all transitions are included in graph; and the model is reversibility, because every path from initial state \(M_0\) can reach \(M_0\) again eventually. Therefore, the behavior model of the product is liveness, reversibility and boundness.

Then, other behavior models of products are been analyzed as the above method. And we can conclude that the AFPN of product family is also liveness, reversibility and boundness.
5 Related Work

Before we examine related approaches in detail, we note that they can be broadly categorized into two lines. On the one hand, there are approaches that just provide modeling languages [5, 6, 10, 16-18] and others also provide a modeling language with verification mechanisms [3, 7-9, 19], as we do. Among the former, there are a number of formal approaches that only provide modeling methods as well as UML-based approaches, where family models can be used to derive the model of a specific product, but not be verified. On the other hand, one can distinguish between approaches that consider features as a first-class citizen and those that express variability as part of the behavioral model.

A. Classen et al. [19] consider features as the first-class abstractions of system requirements. They model variability of functional requirement and behavior in a different model, which is a clear separation of concerns. And the relationship of features and behavior is constructed which makes a designer obtain the behavior model for a particular product conveniently.

A. Fantechi et al. [5, 6] consider the variable behavioral model of the entire product family as the starting point. They propose EMTS which extends Modal Transition System (MTS) to model the one-to-many relations among transitions. Further, they propose GEMTS to model the many-to-many relations among transitions by introducing explicit variability operators, which makes the properties of must, may, alternative, or and cardinality can be modeled. And ACTL temporal logic is used to checking models. Patrizia Asirelli et al. [7, 8] also propose to model behavior variability based on MTS, a function visit is defined to obtain the behavior of a product from SPL behavior model, and adopt MHML and vaCTL to verify the configured behavior. However, all these approaches lack the notion of feature and priority between transitions. Moreover, the relationship and mapping between feature model and behavior model do not discussed, which are considered in our paper.

Ziadi et al. [16] propose a UML profile for variability with stereotypes for optionality, alternatives and refinements. This profile is used to model product line behavior with UML sequence diagrams. But it neither provides verification mechanisms nor uses a first-class variability approach.

Radu Muschevici et al. [10] introduce Feature Petri net to model behavior of product families with a high degree of variability which is a lightweight extension to Petri nets. And present Dynamic Feature Petri Nets to support the modeling of dynamic SPLs. However, priorities among transitions are not considered which may lead to conflicts. Moreover, the model verification is...
6 Conclusion

Modeling the behavior model of the whole software product lines is infeasible, especially obtaining the behavior model of a particular product based on software requirements. This paper analyzes the advantages and disadvantages of the various modeling methods on the basis of previous research, presents Adaptable Featured Petri Nets (AFPN), lightweight Petri net extensions designed for modeling the behavior of software product lines.

AFPN capture the behavior of entire product lines in a single, concise model, opening the way for efficient analysis and verification. At first, in order to model detailed behavioral variations, AFPN leverages on establishing relationships between features and transitions, which can reflect variations of features to behavior models directly. And the relationship between AFPN semantics and PN semantics is discussed. Then, derivation steps are presented, through which the behavior model of a particular product can be obtained. Thereby, we can use analysis techniques on petri net theory to verify the derived behavior model. Finally, we have analyzed behavior of a vending machine family using a PN Editor using our methodology. As a result, our approach helps to modeling and analysis product families effectively in the early phases of development.

Several issues deserve a further investigation, as suggested in the previous sections. A major issue is developing an analysis method, which can verify behaviors of product families based on an AFPN effectively solving the problem of state-space explosion for large systems. Another issue is study a modeling language to enhance the ability of description to reduce modeling complexity.

References


