Towards Visualisation and Analysis of Runtime Variability in Execution Time of Business Information Systems based on Product Lines.

Towards Visualisation and Analysis of Runtime Variability in Execution Time of Business Information Systems based on Product Lines[*]

Ildefonso Montero, Joaquín Peña, Antonio Ruiz-Cortés
Departamento de Lenguajes y Sistemas Informáticos
Av. Reina Mercedes s/n, 41012 Seville (Spain)
University of Seville
{monteroperez, joaquinp, aruiz}@us.es

Abstract

There is a set of techniques that build Business Information Systems (BIS) deploying business processes of the company directly on a process engine. Business processes of companies are continuously changing in order to adapt to changes in the environment. This kind of variability appears at runtime, when a business subprocess is enabled or disabled. To the best of our knowledge, there exists only one approach able to represent properly runtime variability of BIS using Software Product Lines (SPL), namely, Product Evolution Model (PEM). This approach manages the variability by means of a SPL where each product represents a possible evolution of the system. However, although this approach is quite valuable, it does not provide process engineers with the proper support for improving the processes by visualising and analysing execution-time (non-design) properties taking advantage of the benefits provided by the use of SPL.

In this paper, we present our first steps towards solving this problem. The contribution of this paper is twofold: on the one hand, we provide a visualisation dashboard for execution-traces based on the use of UML 2.0 timing diagrams, that uses the PEM approach; on the other hand, we provide a conceptual framework that shows a roadmap of the future research needed for analysing execution-time properties of this kind of systems. Thus, due the use of SPL, our approach opens the possibility for evaluating specific conditions and properties of a business process that current approaches do not cover.

1 Introduction

The development of *Business Information Systems* (BIS) is focused on providing techniques and mechanisms for designing software systems based on the business processes of the companies. One of the implementation approaches is based on deploying business processes, defined graphically, on process engines that execute the specification.

Variability in average-size business processes is high enough for motivating the use of tailored mechanisms to be managed. For that purpose, there exists only one approach devoted to managing business processes variability using *Software Product Lines* (SPL)[11]. In this approach, A. Schnieders *et al.* explore the idea of applying *Software Product Lines* (SPL) for managing runtime variability of a unique BIS in an approach called *Process Family Engineering* (PFE) [11]. In PFE, each *product* represents an *evolution* of the system (at runtime). In this context, the term *product* is defined as a set of features that are enabled/running at a certain moment, and the term *evolution* is defined as a transition from one product to another. See Section 2 for a detailed definition of these concepts.

However, although PFE may be the solution to managing the evolution of the business process of a company, the proposed models, namely feature models (equivalence between *feature* and *business process* is defined in Section 2), are not expressive enough for documenting this evolution. The main problem is that these kind of models are devoted to model static variability, and not runtime variability [9]. See Section 3.2 for details on this problem.

The *Product Evolution Model* (PEM) [8], which is shown in Section 6, complements PFE for properly representing runtime variability in BIS. For that purpose, it integrates PFE with several proposals for modeling runtime variability in SPL, namely [5][6][7] (see Section 6 for a discussion of these approaches). This approach is oriented to provide a set of artifacts able to represent properly runtime

[*]This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web-Factories (TIN2006-00472) and under a scholarship from the Education and Universities Spanish Government Secretariat given to the author Ildefonso Montero.

variability at design time and trigger events that drives these changes. PEMs are defined in two layers: (i) an abstract formal description of business evolutions, presented in Section 3.1 and (ii) a proposal for representing it based on a state-based notation where each state represents a product and each evolution between two or more states is represented by means of an inclusion or exclusion of features, presented in Section 3.2. PEM uses the *Business Process Model Notation* (BPMN) [3] for representing this, but the proposal is open to other notations. The main benefits of this approach are that it provides sufficient expressiveness for representing runtime variability in BIS, and events or conditions that fire business evolutions can also be represented.

Although this approach presents a valuable solution for representing runtime variability in BIS at design time, process engineers need to visualise and analyse properties of the business at execution-time, for example: how long each product is active or which is the percentage of benefits obtained in each product at a certain moment. This kind of evaluation is studied in the *Business Process Mining* field [4][13][10]. However there not exists any approach in this field that manages variability using SPLs.

The main motivation of this paper is that, to the best of our knowledge, there does not exist any approach, that takes advantage of the extra information than a SPL approach provides. This extra information allows process engineers to visualise the execution of the process showing how the business evolves between several configurations. See Section 6 for a discussion on the approaches that inspires us and their deficiencies for the BIS field. This kind of visualisation is valuable since it allows process engineers to focus on higher level properties of the business, such as which product is more profitable, than those proposed in the Business Process Mining field.

The contribution of this paper is twofold: on the one hand, we provide a visualisation dashboard of execution traces based on the use UML 2.0 timing diagrams and its integration with current approaches, detailed on Section 4. On the other hand, we have studied the problems related with the analysis of execution traces properties providing a conceptual framework that shows a future research, which is shown in Section 5. Thus, our ideas open the possibility to reason about products, evolutions, triggers, etc., which, to the best of our knowledge, is not present in current approaches based on SPL, neither in the process mining field.

2 Adaptation of the SPL terminology to the context of the paper

In this paper, we use concepts of the SPL field with a slightly different meaning. These changes in the meaning occur because we applied these concepts to assets that are processes and not executable software pieces. In the following paragraph, we clarify the meaning of these concepts in our context:

Features* and *Business Processes: the *Product Evolution Model* (PEM) approach [8] establishes an equivalence between *business processes* and *features* as follows: (i) a *feature* represents a *subprocess* (part of a complete process that starts and ends), and (ii) *child features nodes* of a feature model, also considered *variants* in [9], corresponds with *concrete processes* (processes that do not present abstract or complex activities). In this paper, we also use a direct correlation between a *feature* and a *business subprocess*. Hereafter, we use the term *feature* to refer to both terms.

Predictable Evolutions* and *Products: Businesses evolve to adapt to environmental changes. This is done by including or excluding features or modifying existing ones. As shown previously, we use a SPL to manage these changes. Thus, we define a *product* as the set of features (subprocesses) that are enabled/running at a certain moment. In addition, we define the term *evolution* to denote the changes or transitions from one *product* to another. Note that we only take into account the *evolutions* that can be predicted at design time, called *predictable evolutions* (hereafter, *evolutions* for shortening).

Design-time*, *runtime* and *execution-time: It is called *design-time* as an interval of time in which we build the business process model and represent its variability, including *runtime* variability. *Runtime* is defined as an interval of time which starts when the business processes modeled are deployed on process engines. Thus, this term can be also named as *deployment time* or *configuration time*. Finally, *execution-time* is defined as the interval of time which starts when the business processes deployed are executed in the process engine. Thus, *runtime* variability is modeled at *design-time*, by means of PEM, and visualised and analysed at *runtime*. This analysis is based on the observation of *runtime* properties at *execution-time*. This observation is performed analysing the traces produced by the system.

Predictable* and *Unpredictable Triggers: *Triggers* act as stimulus of an evolution from a product to another. An *Unpredictable Trigger* is defined as something happening in the environment that fires an evolution that cannot be predicted at design time. A *Predictable Trigger* is defined as a condition that can be defined at design time that fires an evolution. See Section 4 for an example of *predictable* and *unpredictable triggers*.

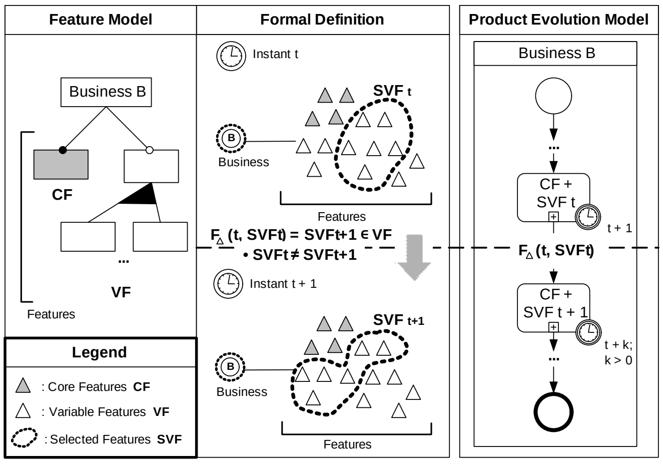


Figure 1.a. Rigorous Description Figure 1.b. Graphical Notation

Figure 1. Product Evolution Model approach defining an evolution of a business by the $F_\triangle$ function in t and $t+1$.

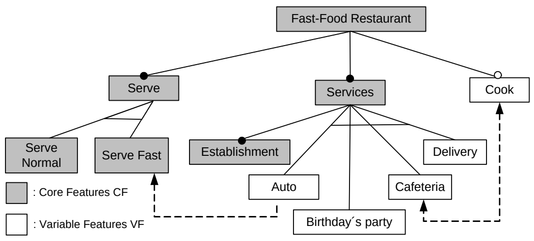


Figure 2. Case Study: Fast Food Restaurant

3 Representing Runtime Variability of BIS using the Product Evolution Model Approach

Product Evolution Model (PEM) is focused on providing a sufficiently expressive design-time model for representing runtime business properties. PEM provides in [8] an abstract rigorous description and a proposal for representing it by means of an extension of BPMN using stereotypes, including a case study. We show that description in the following sections.

3.1 PEM Rigorous Description

Let B be a business. Each business can be defined as a set of processes (denoted with P). Thus, B can be defined as follows:

$$B = \{P_1, P_2, ..., P_k\}; k > 0$$

Let *CF* be the set of common features, and let *VF* be the set of variable features, thus B is defined formally as a tuple containing all the CF and a subset of VF denoted as SVF:

$$B = (CF, SVF \in VF)$$

As shown before, in PFE, each set of features enabled at a certain moment represents a product. Thus, we can say that the CF of a B are always enabled at runtime, but the set of features in VF is not fixed at runtime.

Thus, we can set up a product line that takes into account this runtime variability. For formalizing these concepts we should redefine each business B as:

$$B = (CF, SVF \in VF, F_\triangle :$$
$$: t, \{Feature \times ... \times Feature\} \mapsto$$
$$\mapsto \{Feature \times ... \times Feature\})$$

where $F_\triangle$ is a function that given an instant t transforms the set of SVF_t into the new set of variable features of the following time instant t+1, that is to say SVF_{t+1}, formally:

$$F_\triangle(t, SVF_t) = SVF_{t+1} \in VF$$
$$\bullet SVF_t \neq SVF_{t+1}$$

Figure 1.a sketches a graphical representation of $F_\triangle$, where it is represented the transformation of SVF_t into SVF_{t+1}. In an instant t there exists a specific set of SVF_t for business B that evolves in instant $t+1$ to a different set SVF_{t+1}.

3.2 PEM Graphical Notation

As shown previously, a business that evolves can be represented by $B = (CF, SVF \in VF, F_\triangle)$, where the evolution is defined by the $F_\triangle$ function in t.

In PFE, feature models are used to represent which features are variable and which are not. From this, the set of common (CF) and variable (VF) features can be obtained [1]. Thus, CF and VF can be represented by means of a feature model.

However, the feature model cannot establish the order of activation of features at runtime. This order is represented using $F_\triangle$, but as feature models are not devoted for representing runtime variability [5], they cannot be used for representing the variable t needed in the $F_\triangle$ function. For solving this problem, the *Product Evolution Model* (PEM) approach proposes a graphical notation that covers t and $F_\triangle$. This model is defined by means of a BPMN state machine where each state represents a product and each evolution between two or more states, is represented by means of a transition that is an application of the $F_\triangle$ function. In Figure 1.b, we show an evolution of a business from time t to

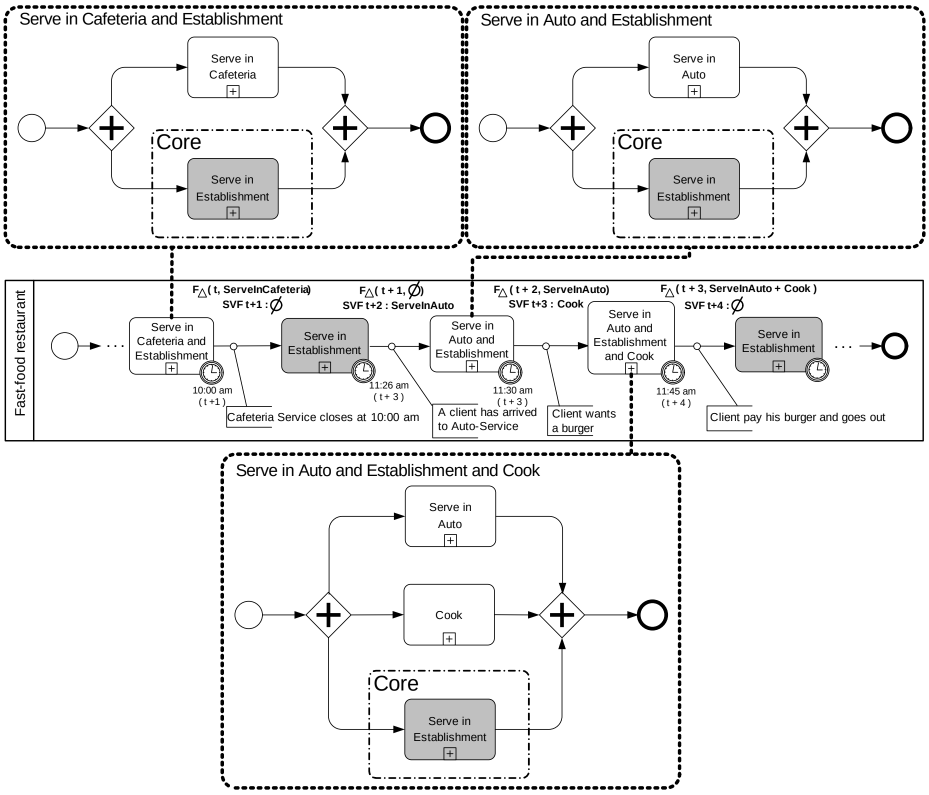


Figure 3. Fast-food restaurant Product Evolution Model BPMN Compositions

time $t+1$ by means of applying the $F_{\triangle}$ using a PEM model. As shown in the figure, there exists two different products. The first product is composed by the set of features CF and SVF_t. $F_{\triangle}$ in t fires an evolution at $t+1$ which implies the creation of the second product. This product is also composed by CF, since it never changes, and SVF_{t+1} which is different than previous SVF_t.

In order to illustrate PEM and the rest of the paper, we use a case study of a fast-food restaurant. Figure 2 depicts a simplified set of features pertaining to a fast-food restaurant: *Serve Normal*: which is defined as the normal activities for serving products in the restaurant, *Serve Fast*: which is defined as the activities needed for serving products in the restaurant when there exists a higher demand, and *Serve in Establishment*: which is defined as the activities for serving products performed only into the establishment. These features are CF and the rest are VF. In Figure 3 we present the PEM of this case study. Each state contains a BPMN state chart that represents how all the features are performed. It defines the evolution of the business at runtime showing that in every runtime instant t there exists a different SVF selected. For example, on a time instant t the restaurant opens its cafeteria service. In this moment, there exists two different processes running in parallel: *Serve in Cafeteria* and CF (*Serve in Establishment Normal/Fast*). When the restaurant closes its cafeteria service on time instant $t+1$,e.g. 10:00 am, the $F_{\triangle}$ function is applied and an evolution is performed to another state, that represent a different product, composed only by CF. After that, the restaurant opens its Auto-Service, because a client has arrived with his car at $t+2$. When this client orders a burger, the *Cook* subprocess is enabled, what happens in time instant $t+3$. When the burger is served, the system evolves to time instant $t+4$.

4 Visualisation of Runtime Variability in BIS

Process engineers need support for improving the processes by means of visualising and analysing the execution-time traces of business evolutions. For that purpose we provide a single view that illustrates all the transition from one product to another in certain moment. We use UML *Timing Diagrams* to represent this information. Timing diagrams are one of the new artifacts added to UML 2.0 which are used when the goal of the diagram is to reason about time. We call this view the *Business Dashboard*.

UML provides two different representations of timing diagrams: (i) *State* or (ii) *General value*. Both representations contain events and constraints that represent stimuli for an evolution. In Figure 6, we have included an example of each

Rigorous Description of PEM	**UML 2.0 Timing Diagrams**
Product	State
F∆	Transition / Stimulus
Trigger X	Predictable Trigger {X}
	Unpredictable Trigger X

Figure 4. Rigorous Description of PEM and Timing Diagram Correspondence

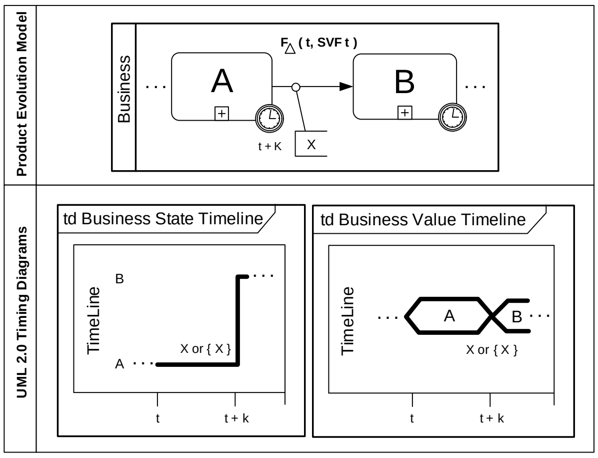


Figure 5. Obtaining Timing diagrams from Product Evolution Model

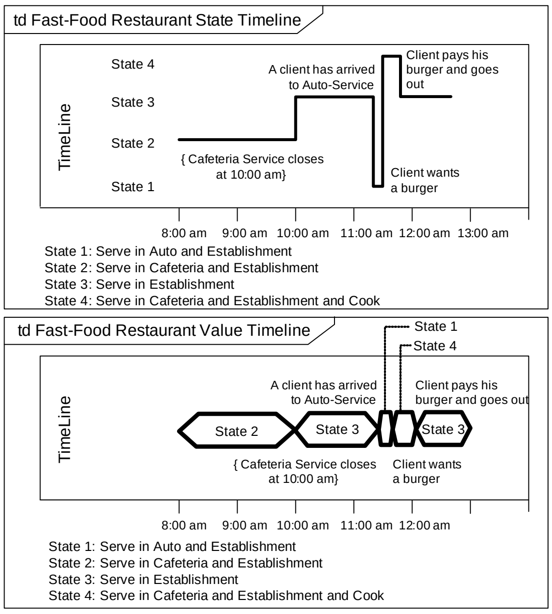

td Fast-Food Restaurant State Timeline
TimeLine
State 4
State 3
State 2
State 1
A client has arrived to Auto-Service
Client pays his burger and goes out
{ Cafeteria Service closes at 10:00 am}
Client wants a burger
8:00 am
9:00 am
10:00 am
11:00 am
12:00 am
13:00 am
State 1: Serve in Auto and Establishment
State 2: Serve in Cafeteria and Establishment
State 3: Serve in Establishment
State 4: Serve in Cafeteria and Establishment and Cook
td Fast-Food Restaurant Value Timeline
State 1
State 4
TimeLine
A client has arrived to Auto-Service
Client pays his burger and goes out
State 2
State 3
State 3
{ Cafeteria Service closes at 10:00 am}
Client wants a burger
8:00 am
9:00 am
10:00 am
11:00 am
12:00 am
State 1: Serve in Auto and Establishment
State 2: Serve in Cafeteria and Establishment
State 3: Serve in Establishment
State 4: Serve in Cafeteria and Establishment and Cook

Figure 6. Visualising fast-food restaurant evolutions by means of UML 2.0 Timing diagrams

view. As shown in figure, the representation called *State* focuses on showing every evolution, while the representation called *General value*, focuses on each product instead of an implicit representation of an evolution. Given the characteristics of each view, the second representation, *General value*, is more adequate for software product lines where the number of products is high, while the first, *State*, is more adequate for software product lines where the number of products is low since evolutions are shown graphically.

Using the rigorous description defined previously in Section 3.1, we provide the correspondence between the information managed in PEM and timing diagrams. Figures 5 and 4 show the equivalence between a PEM and a timing diagram. As shown, each product modeled, using PEM, obtained from the application of the $F_\triangle$ function is equivalent to a state in a timing diagram. Notice that each $F_\triangle$ is performed in a time instant $t + k; k \geq 0$ when a trigger *X* holds. Notice that in timing diagrams, *X* denotes an unpredictable trigger, and $\{X\}$ a predictable trigger. See Figures 4 and 5 for an example of both kind of triggers. In PEM there is no difference between unpredictable and predictable triggers, since unpredictable only appears at execution-time and PEM is a design-time model.

Figure 6 shows the timing diagrams of an execution trace of our case study. Each product is denoted by a state number. As shown in Figures 3 and 6, there are four evolutions: (i) from product denoted by State 2 to another denoted by State 3 ($F_\triangle$ in t), which represents the predictable trigger: *Cafeteria Service closes at 10:00 am*. This implies an exclusion of *Serve in Cafeteria* feature from our product; (ii) from State 3 to State 1 products ($F_\triangle$ in $t + 1$) that is performed when a client arrives to Auto-Service. This unpredictable trigger fires the second evolution that implies that feature *Serve in Auto* must be added or enabled in the new product; (iii) from State 1 to State 4 products ($F_\triangle$ in $t + 2$) when a client wants a burger, that implies that feature *Cook* must be added in the new product; and finally (iv) from State 4 to State 3 ($F_\triangle$ in $t + 3$) when client pays his burger and goes out.

In order to validate our approach, we have developed an automated transformation from a PEM execution trace to a timing diagram, concretely to *State* representation, using `gnuplot`[1], a command-driven interactive function and data plotting software. In Appendix we present an screenshot of the timing diagram of our case-study obtained using this transformation.

[1]http://www.gnuplot.info/

5 Roadmap for Research on Analysis

As shown previously, once runtime variability is visualised by means of timing diagrams, process engineers need to evaluate execution-time properties of the business. There are many basic analysis questions that can be performed, for example:

- Find constraints and events that fire a subprocess and calculate its relative frequency, i.e: *How many times does a client arrive in Auto-Service?*
- Calculate relative frequency of the activation of a subprocess, i.e: *How many times is Serve in Cafeteria subprocess executed?*
- Analyse processes bottlenecks, i.e: *Which is the activity with the lowest level of performance?*

These kinds of questions are usually supported by current software tools for business process management and by the Process Mining approach [4][13][10]. They are focused only on analysing single/isolated subprocesses. However, given that PEM and PFE are based on SPL, there are other analysis questions that may be supported providing higher level views for analysing the features, as for example:

- Analyse for each product: cost, risk and benefits.i.e: *Which is the percentage of benefits of product "State 1"?*
- Compare the performance of a certain feature when running in different products (dependencies with other features, events and/or constraints may affect the performance). i.e: *earning rate of product defined by state 1 is less than earning rate of product defined by state 2 on Fridays when it is executed in parallel with the Serve in Auto-Service feature.*

For arranging this research problem we propose two artifacts: (i) a metamodel for arranging and determining the needed information for supporting the analysis questions presented previously, which includes business process management support for current analysis questions, and (ii) a conceptual framework for future research on analysis which specifies how future research lines are related and may be conducted.

5.1 Analysis Metamodel

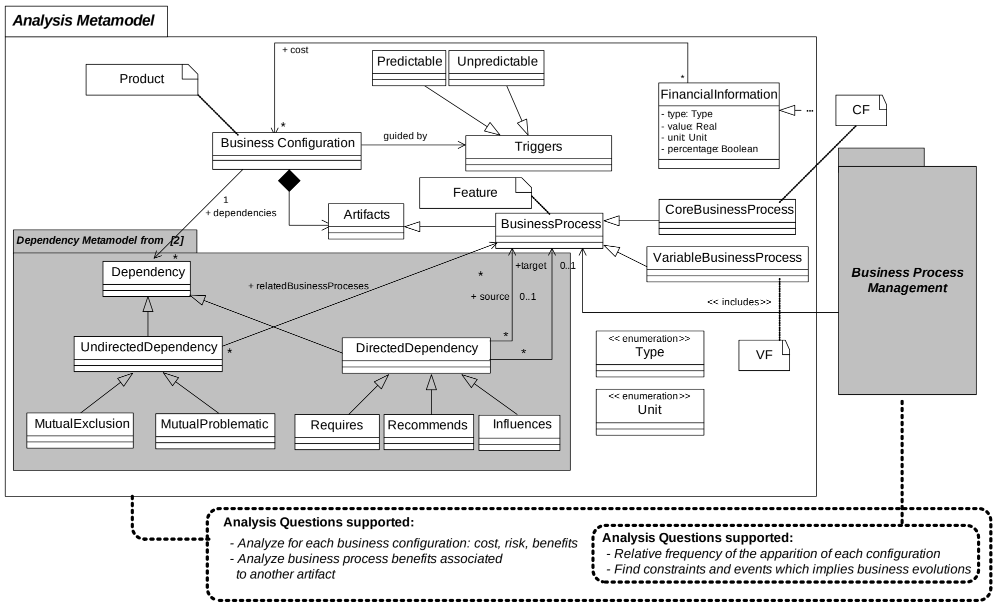
In this section we show the metamodel for arranging and determining needed information for supporting analysis questions presented previously. Figure 7 shows the metamodel that contains the following elements:

- **Business Process Management package**: it provides business process definition and represents the support for basic analysis questions provided by current tools for business process management.
- **Analysis Metamodel package**:
  - **Business Configurations**: states in timing diagram are considered business configurations represented by the *Business Configuration* metaclass. Each configuration contains a set of business processes which are modeled by means of the *Business Process* metaclass. It can be specialized to the *Core Business Process* or *Variable Business Process* metaclasses, previously denoted as CF and VF in the PEM definition.
  - **Predictable and Unpredictable Triggers**: these elements drive the evolutions of business configuration. They are modeled by the *Predictable*, and *Unpredictable* metaclasses.
  - **Financial Information**: Each business configuration has an associated cost, represented by the *Financial Information* metaclass, where we may add additional information about it; i.e: "*Serve in Establishment* process has an associated human resources cost of two employees" statement can be modeled by an association between the *Business Process* and *Financial Information* metaclasses instances, which attributes of second metaclass *type*, *value* and *unit* are initialised to "*human resource*", "2", and "*employees*" values respectively.
- **Dependency Metamodel package**: Business processes has associated a set of dependencies between them which are modeled by means of the *Dependency* metaclass. As shown in figure 7, the metaclasses in the *Dependency* package are based on Botterweck *et al.*'s metamodel for supporting feature configurations by interactive visualisation [2].

5.2 Conceptual Framework for Research on Analysis

For materializing these analysis operations we propose a conceptual framework for research on analysis based on filtering and analysing evolutions to perform queries using the information on the metamodel presented previously. Figure 8 shows it using a stereotyped association, *«uses»*, between the framework and our analysis metamodel. The framework also takes into account a representation for a Product Evolution Model and timing diagrams.

We have divided the elements included in the framework into those that can be implemented using our current results,



Analysis Questions supported:
- Analyze for each business configuration: cost, risk, benefits
- Analyze business process benefits associated to another artifact

Analysis Questions supported:
- Relative frequency of the apparition of each configuration
- Find constraints and events which implies business evolutions

Figure 7. Proposed metamodel for analysing runtime variability in BIS (partially based on [2] metamodel)

labeled as *'implementation'*, and those that require for some research effort, labeled as *'research'*:

- **For visualising variability**:
  - **Artifact Factory** *(Implementation)*: Process engineers need to visualise evolutions. Evolutions are represented by means of *Timing Diagram* components. The component called *Artifact Factory* allows process engineers to generate timing diagrams from a business process modeled by means of Product Evolution Model using the information shown in the metamodel presented in Figure 7. PEM is represented by BPMN Product Evolution Model component. Notice that for obtaining PEM we need the core features which are obtained using FAMA [1]
- **For analysing variability**:
  - **Filter** *(Research)*: Process engineers can be interested in performing analysis questions about only one part of the timing diagram. For that purpose, the *Filter* component must provide query operations on BPMN Product Evolution Model and *Timing Diagrams*. The definition of these operations could be based on formalisms, such as Constraint Satisfaction Problem (CSP), Temporal Logic, Petri nets, etc. This is one of the possible future research lines, as shown in Figure 8
  - **Analyser** *(Research)*: As shown previously, process engineers need to perform analysis questions in order to improve their company. For that purpose the *Analyser* component should perform all possible analysis questions or operations from artifacts represented by *Timing Diagram* and BPMN Product Evolution Model components (the information shown in the metamodel). These operations can be grouped as basic operations, i.e:*obtain business process dependencies* and complex operations obtained by means of basic operations combinations, i.e: *obtain financial information about all possible business processes dependencies*. This represents another future research line.

6 Related Work

The *Business Process Mining* field, or *Process Mining* for short, is focused on extracting information about processes using execution traces [4][13][10]. For that purpose, these approaches provides some visualisation artifacts and frameworks for the automated analysis. Although this field is the most realted with the topic of this paper, there not exist any approach that support BISs based on product lines. Thus, these approaches cannot address the analysis questions provided by our approach and cannot represent the information on evolutions provided by our approach.

As shown in Section 3.2, feature models (FM) are one

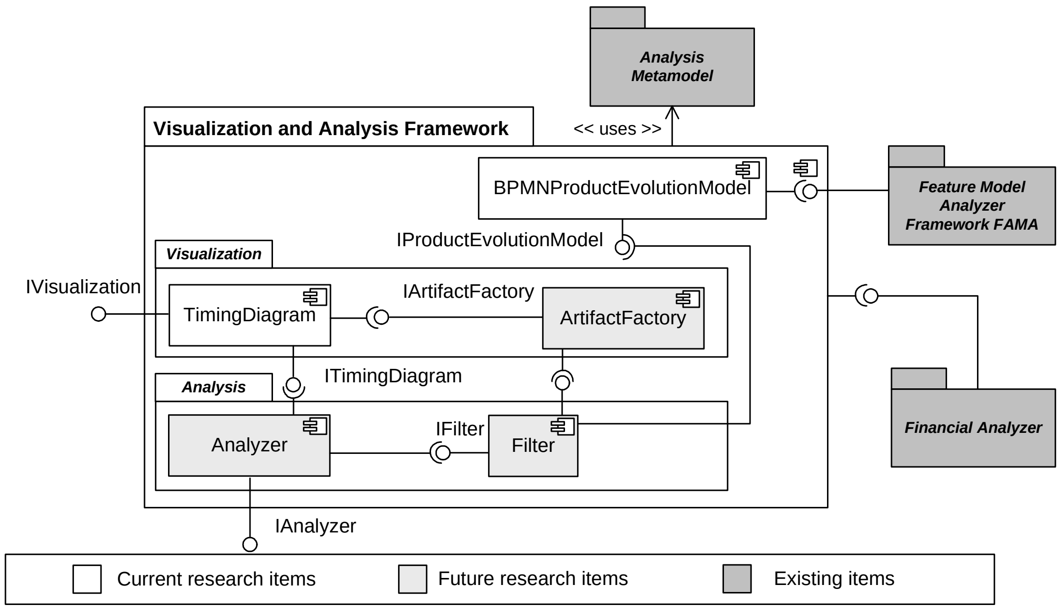


Figure 8. Runtime Variability Visualisation and Analysis Framework

of the most used artifacts for modeling variability. Unfortunately, as shown in Section 3.2, FM are devoted to design variability, and not to runtime variability [5]. There exists three approaches, to the best of our knowledge, that describe how to represent runtime variability in SPL.

First, J. Bosch *et al.* [7] introduce an extension of FM for representing runtime variability. Bosch's notation is slightly different from FODA's or FORM's notation. They introduce a new kind of feature for representing features that vary at runtime, called *external feature*, represented by means of a dashed rectangle. Figure 9 depicts an example of a feature model using this notation that represents the plugin support provided by the *Firefox* web browser. It represents that there exists one feature called *Website Debugger*, that can be enabled/disabled at runtime. As can be observed, the trigger events or conditions that fire this variability can not be represented with this approach, i.e: *plugin Website Debugger is enabled at runtime only in websites with domain* US.ES.

Sinnema *et al.* [12] propose a framework for modeling variability in SPL, called COVAMOF [2], which proposes a language for describing variation points named *COVAMOF Variability View Language* (CVVL) that takes into account enabling/disabling time. It is similar to the previous approach for representing runtime variability using in CVVL the tag *bindingtime*. The CVVL code for *Firefox* web browser example is the following:

```
<variationpoint id=Plugin>
   ...
   <variants>
        ...
      <variant id=Website Debugger>
         ...
         <bindingtime>runtime</bindingtime>
      </variant>
     </variants>
     ...
</variationpoint>
```

[2]www.covamof.com

H. Gomaa *et al.* [6][5] propose a set of models for representing runtime variability based on evolutionary reconfigurable software architectures. The different versions of an evolutionary system are considered a software product line, where each version of the system is a product and the reconfiguration is defined by a state machine that, for each component, represents the steps that have to be performed to evolve from a normal operation state to an inactive state. Once inactive, the component can be removed and replaced with a different version. Figure 10 depicts trigger events in the state machine. It represents how an optional feature named *Beeper* from a *Microwave System* feature model is enabled or disabled at runtime.

For runtime variability management in BIS, the focus of this paper, we have discussed in Sections 1 and the following proposals: *Process Family Engineering* (PFE) [11] and *Product Evolution Model* (PEM) [8] as a complement of PFE for properly representing a design model of runtime variability in BIS. However, none of these approaches provide any visualisation or analysis artifact for execution-time traces.

Given this state of art, to the best of our knowledge, there does not exist any approach for visualising and analysing runtime variability in execution-time of BIS using SPL techniques. This situation motivates us to propose a future

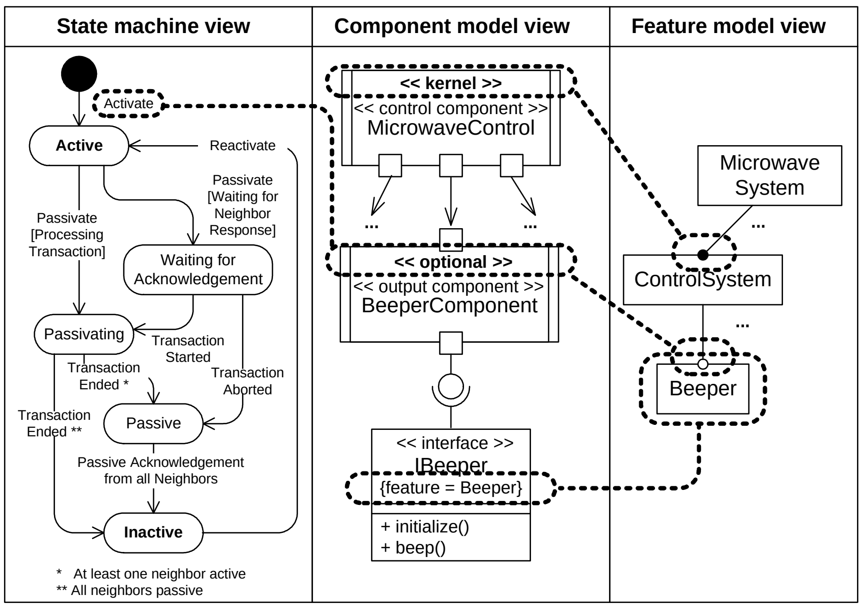


Figure 10. Gomaa approach (Figure taken from [6])

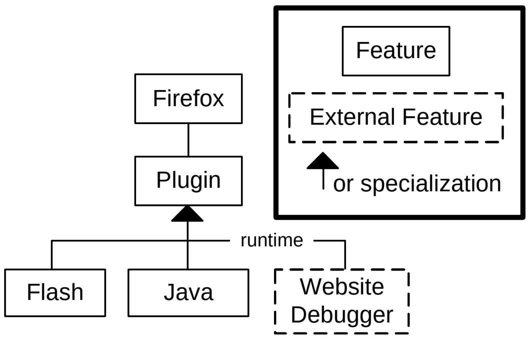


Figure 9. J. Bosch approach

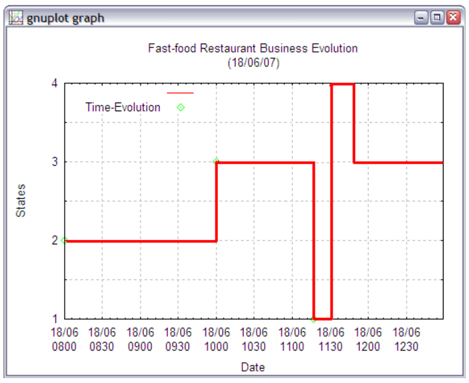


Figure 11. UML 2.0 Timing diagram obtained by gnuplot

research roadmap agenda and an approach for visualisation.

7 Conclusions and Future Research Roadmap

The main motivation of this paper is to provide to process engineers a first step toward an automatised visualisation and analysis of runtime variability in BIS based on SPL. For that purpose, we have explored the feasibility of using PEM for visualising and analysing runtime variability. As a result of our work we have proposed: (i) integration between PEM and a visualisation model based on UML 2.0 timing diagrams; (ii) a metamodel for arranging the information needed for analysing runtime variability in BIS; and (iii) a roadmap for research on analysing that can be used as a research agenda for this topic.

We think that this field is quite interesting and future research should be conducted. Thus the main research lines that could be derived from our framework are the following:

- Visualisation: to perform alternative techniques to those proposed in this paper such as 3D representation, circle graphs, etc.
- Analysis: to explore possible basic and complex operations, obtained by means of basic operations combinations, for runtime business evolution execution-traces in order to perform queries, filters and analysis. As proposed in Section 5.2, the definition of these operations may be done using several formalism, such

as a *Constraint Satisfaction Problem* (CSP), Temporal Logic, Petri nets, etc.

In addition, due to our work is highly related to the *Process Mining* field, a survey of the techniques used in this field may help to clarify the first steps to be performed in the context of the future research lines identified

8 Acknowledgments

The authors would like to thank the reviewers of the Second International Workshop on Variability Modelling of Software-intensive Systems for their useful comments. We would like also to thank Patrick Heymans and David Benavides, whose comments and suggestions improved the presentation substantially.

References

[1] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, 2007.

[2] G. Botterweck, D. Nestor, C. Cawley, and S. Thiel. Towards supporting feature configuration by interactive visualization. In *VISPLE'07: Proceedings of the 1st International Workshop on Visualization in Software Product Line Engineering - collated with SPLC 2007.*

[3] BPMI. Business process modeling notation BPMN version 1.0 - may 3, 2004. *OMG.*

[4] A. K. A. de Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral. An outlook on semantic business process mining and monitoring. In R. Meersman, Z. Tari, and P. Herrero, editors, *OTM Workshops (2)*, volume 4806 of *Lecture Notes in Computer Science*, pages 1244–1255. Springer, 2007.

[5] H. Gomaa. Feature dependent coordination and adaptation of component-based software architectures. In *WCAT '07: Proceedings of the 4th Workshop on Coordination and Adaptation Techniques for Software Entities*, 2007.

[6] H. Gomaa and M. Hussein. Model-based software design and adaptation. In *ICSEW '07: Proceedings of the 29th International Conference on Software Engineering Workshops*, 2007.

[7] J. V. Gurp, J. Bosch, and M. Svahnberg. On the notion of variability in software product lines. In *WICSA '01: Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'01)*, 2001.

[8] I. Montero, J. Peña, and A. Ruiz-Cortés. Representing Runtime Variability in Business-Driven Development systems. In *Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS08)*, 2008.

[9] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, September 2005.

[10] A. Rozinat, A. A. de Medeiros, C. Günther, A. Weijters, and W. van der Aalst. The need for a process mining evaluation framework in research and practice. In *Proceedings of the Third International Workshop on Business Process Intelligence. (pp. 73-78). Brisbane, Australia: Queensland University of Technology.(2007).*

[11] A. Schnieders and F. Puhlmann. Variability mechanisms in e-business process families. In *Proceedings of BIS '06: Business Information Systems*, 2006.

[12] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. COVAMOF: A Framework for Modeling Variability in Software Product Families. In *Proceedings of the Third Software Product Line Conference (SPLC04)*, San Diego, CA, 2004.

[13] W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. A. de Medeiros, M. Song, and H. M. W. Verbeek. Business process mining: An industrial application. *Inf. Syst.*, 32(5):713–732, 2007.

9 Appendix: gnuplot Experiment

In order to provide an experiment of automated transition from PEM to timing diagrams for visualising runtime business evolution execution-trace, we have deployed our case study PEM modeled by BPMN to a business process execution engine and it has been translated to WS-BPEL. We have developed two basic web services for representing choreography interaction between business process actors and we have executed it obtaining a runtime execution trace that has been stored in a file denoted as "fast-food-restaurant.dat". The following gnuplot script takes this file as input for plotting the timing diagram shown in Figure 11.

```
#************************************************
# fast-food-restaurant.dem
# Author:
#   Ildefonso Montero Pérez - monteroperez@us.es
#   Dpto. Lenguajes y Sistemas Informáticos
#   Av. Reina Mercedes s/n, 41012 Seville (Spain)
#   University of Seville
# Description:
#   A gnuplot script to represent an UML 2.0
#   timing diagram of Fast-food restaurant
#   Product Evolution Model
#************************************************
set title "Fast-food Restaurant Business
           Evolution\n(18/06/07)"
set style data steps
set xlabel "Date"
set timefmt "%d/%m/%y\t%H%M"
set xdata time
set xrange ["18/06/07\t0800":"18/06/07\t1259"]
set ylabel "States"
set format x "%d/%m\n%H%M"
set grid
set key left
plot 'fast-food-restaurant.dat' using 1:3 t ' ', \
     'fast-food-restaurant.dat' using 1:3 t
     ' Time-Evolution' with points
pause -1 "Hit return to continue"
reset
```