This paper has been accepted by the
European Reuse Workshop '98 (ERW'98)
to take place in November 1998

Applying PuLSE for Software Product Line Development

Jean-Marc DeBaud, Peter Knauber
Fraunhofer Institute for Experimental Software Engineering (IESE)
Sauerwiesen 6
D-67661 Kaiserslautern, Germany
+49 (0) 6301 707 251
{debaud, knauber}@iese.fhg.de

Abstract
Software product lines are considered one of the most promising approaches for efficient software development and reuse [1, 4]. Nevertheless, practical methodologies covering the complete spectrum of construction, usage and evolution of product lines in enterprises are scant.

This paper presents a high-level overview of PuLSE (Product Line Software Engineering), a methodology developed at the IESE. A running example from one project in which we started to apply PuLSE illustrates its components. Further experience from our industrial projects concludes the paper.

1 Introduction

The PuLSE methodology for the engineering of software product lines (PL) consists of three main elements: the deployment phases, the technical components, and the support components [3] (see Figure 1).

The *deployment phases* are logical stages a PL goes through: Initialization of PuLSE, construction, usage and evolution of the PL infrastructure. They describe the activities performed to set up and use the PL.

The *technical components* provide the technical know-how needed to operationalize the PL development. As Figure 1 denotes, they are used throughout the deployment phases. A different facet of each component is often used in each of the phases – though some components directly correspond to phases.

The *support components* package guidelines on how to introduce, deploy and evolve product lines in specific contexts.

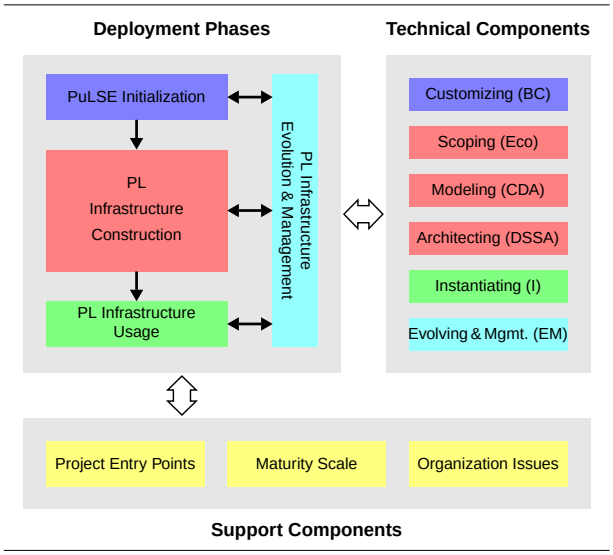

Figure 1. PuLSE Overview

These components are used by the other PuLSE elements.

Section 2 of this paper illustrates major aspects of the technical components and their interrelationships using a running example taken from one of our project. Our partner there is developing a PL of merchandise information systems. Their primary customer is an enterprise that distributes goods to over 400 supermarkets. Each system within the PL has to support processes related to buying, storing, and selling supermarket goods. Among the main variants are systems for different kinds of points of sale and types of goods, as well as systems for different kinds of distribution centers (e.g., conventional wholesale stock, cross-docking stock).

2 The PuLSE Technical Components

2.1 Baselining and Customization: PuLSE-BC

PuLSE-BC's role is to instantiate the PuLSE methodology and tailoring it to an enterprise context. It is split into a *baselining* and a *customization* phase.

Specific PL situations are characterized by *customization factors*. These have an impact on the process for developing the PL and thus are used for tailoring PuLSE. The information required to determine these customization factors is gathered during the *baselining* phase. *Customization* of PuLSE entails deriving a complete process, including the definition of workproducts used, their relations, and their representations depending on these factors.

Example: In the merchandise information system example, the investigation of some customization factors ("relevant abstraction types in the domain", "application type", etc.) during baselining implied the capture of business processes as workflows. To support these, business rules, rationales, and a glossary were also identified as required workproducts. As a consequence of this, PuLSE-CDA (see section 2.3) was customized to use the respective notations as well as the appropriate processes for their elicitation during domain analysis.

2.2 Scoping the Domain: PuLSE-Eco

PuLSE-Eco is the first of three components used in the construction phase of the PL infrastructure. PuLSE-Eco aims at determining an economically viable scope for the PL.

First, anticipated PL members are determined and their characteristics are mapped out. This is done by augmenting business objectives identified by PL stakeholders with *characterization and benefit functions* (see Figure 2). The former evaluate characteristics for each product. This information is added to the product map. During the following *benefit analysis*, the gathered information is used to identify the best products for the PL and thus its scope: First the values of the characterization functions are used to assign values to the benefit functions, then the different benefit functions need

to be balanced in order to come up with a single scope definition. This is a classical *multi-objective decision problem*. A large number of techniques for addressing this type of problem have already been described in the literature (e.g., [5]) and can be used directly.

Example: The table in Figure 2 is a very simplified documented product map for the domain of merchandise information systems. It rows shows two top-level subdomains ("Goods Reception" and "Ordering") with some subtasks. Many rows containing subtasks and further subdomains have been omitted for reasons of space, but the hierarchical structure of the map can be identified. The columns present existing, future, and potential products within the PL. These products are analyzed according to the characterization functions. The rows at the bottom summarize the columns and thus enable an evaluation of the products under situation-driven points of view. The rightmost columns provide an analysis of the significance of the different characteristics. These summaries are determined by the benefit functions. As a result of the evaluation, the characteristics in gray table areas have been determined to be part of the PL scope. Note that the same type of analysis can also be done from a product viewpoint as shown at the bottom of the table.

2.3 Modeling the Product Line: PuLSE-CDA

In PuLSE-CDA, information about the PL scoped by PuLSE-Eco is elicited and modeled using the workproducts and process stemming from PuLSE-BC. We distinguish *storyboards* and other (domain-specific) workproducts. Storyboards are used to capture relevant types of action sequences (workflow diagrams, message sequence charts, etc.) in the domain. Other workproducts (e.g., a data model) capture additional views on the PL.

Populating the models combines eliciting information about single systems and then consolidating this knowledge into generic storyboards and other workproducts that capture variability. To derive PL member specifications from a generic PL model, a *decision model* is created that contains a structured set of decisions. Each decision corresponds to a variability in a workproduct together with the set of possible resolutions. To define a specific PL member, the decisions must be resolved.

Example: Figure 3 shows a generic storyboard representing the task "Process Goods Reception" that models one of the key areas identified in the product map in Figure 2.
The storyboard is captured in DIVERSITY/CDA, a tool we developed to support PuLSE-CDA. Automated support is necessary to manage realistic domains.

2.4 Developing a Reference Architecture: PuLSE-DSSA

PuLSE-DSSA supports the definition of a domain-specific

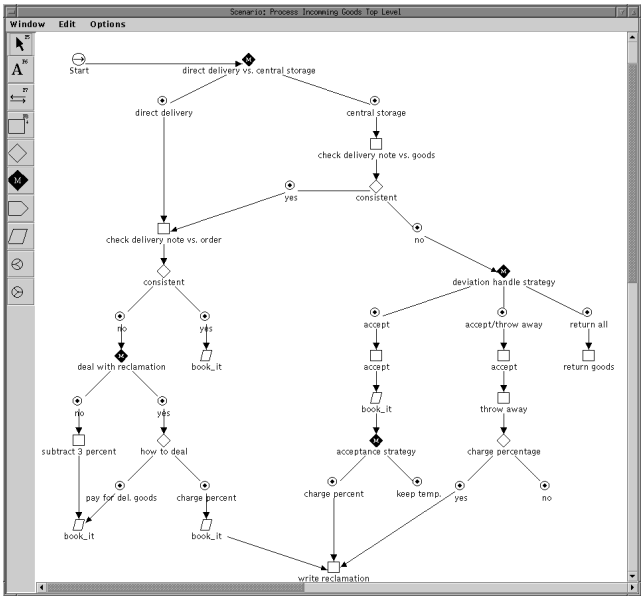

Figure 3. Example - Generic Storyboard

Characterization Functions

s_1: Need to be present
s_2: Competitor products
s_3: Market-share gainer
e_1: Conceptual complexity (1–4)
e_2: Cost estimate (1–4)

X = yes / completely
O = maybe / partially
= no

Benefit Functions

$$^1: \left\lfloor \frac{\sum_p e_2(c,p) \cdot s_1(c,p)}{p_{max} \cdot e_2(c,p)_{max}} \cdot 10 - 1 \right\rfloor$$

$$^2: \left(\sum_p s_3(c,p)\right) / p_{max}$$

$$^3: \sum_c s_1(c) \vee s_2(c) \vee s_3(c) \quad (1\text{–}8)$$

$$^4: \sum_c s_2(c) \vee s_3(c) \quad (1\text{–}8)$$

$$^5: \left(\sum_{c \in scope} e_2(c)\right) / \left(\sum_c e_2(c)\right) \quad (1\text{-}8)$$

Characteristics \ Products			Existing Applications										Future Applications												Potential Applications										Char.-based		
			Product 1					Product 2					Product 3					Product 4					..		Product 5					…						Effort Saving[1]	Market advantage[2]	...
			s_1	s_2	s_3	e_1	e_2	s_1	s_2	s_3	e_1	e_2	s_1	s_2	s_3	e_1	e_2	s_1	s_2	s_3	e_1	e_2	…	s_1	s_2	s_3	e_1	e_2	…								
Goods Reception	Central storage reception		X	X		3	3	X	X		3	3	X	X		3	2	X	X		2	2		X	X		2	3							8	1	
	Branch local reception		X			2	2	X			2	2		X		1	2			X	2	3		X	X		1	2							5	3	
	Acceptance of non-ordered goods		X	X	O	2	3	X	X	O	2	3	X	X		1	2	X	X	O	2	3		X	X	O	2	3							6	6	
	...																																			
Ordering	Reclamation strategy	write reclamation	X	X		2	3	X	X		2	2	X	X		2	2	X	X		3	3		X	X		2	3							7	2	
		charge certain percentage		X		3	2		X		3	4							X	O	3	2			O	O	3	2							6	3	
	Ordering strategy	order at point of sale		X	X	3	3						X			1	2	X	X	O	3	3		X	X		3	3							7	2	
		order at central storage						X	X	0	2	2												X	X		2	1							2	0	
	…																																				
...																																				

Product-based b.a.	Contribution to domain coverage[3]		5			7			4			7			7																					
	Competitive Advantage[4]		3			4			6			3			3																					
	Reference Arch. Effort Saving[5]				7			6			6			6																							
	…																																			

Figure 2. Example - Simplified Product Map

software architecture, which covers current and future applications as described by the PL model produced by PuLSE-CDA.

Task scenarios drive the incremental development of the architecture. These scenarios are derived from the generic storyboards and the other workproducts produced by PuLSE-CDA; they represent major functional requirements and quality-related aspects. First they are ranked according to their architectural significance. The most important ones are used to create an initial architecture. Then, using an iterative process, the others are applied one by one to the current architecture, which is refined or extended accordingly to support the scenarios.

During the creation of the reference architecture, implementation-specific decisions are collected that will have to be resolved during reference instantiation. These decisions and their possible resolutions are captured in the *configuration model* that extends the decision model.

Example: Figure 4 presents a high-level logical view on part of the reference architecture for merchandise information systems. The component “goods reception”, two of its subcomponents, and some neighboring components are shown. The generic architecture includes four variations of the subcomponents “goods reception control flow” and two of “ordering control flow” and “booking” each.

2.5 Instantiating Product Line Members: PuLSE-I

PuLSE-I aims at specifying, instantiating, and validating one member of the PL. This encompasses the instantiation of the PL model and the reference architecture, the creation and/or reuse of products that constitute the instance, and validation of the resulting product.

The customer’s requirements are used to plan the development process for the new instance. Driven by the decision model from PuLSE-CDA, the new PL model instance is specified.

Example: Figure 5 shows one specific instance of the generic storyboard in Figure 3 which defines the workflow for a specific product supporting “direct delivery”. The decision model implemented in DIVERSITY/CDA supports the specification of PL model instances.

Next, driven by the configuration model from PuLSE-DSSA, the architecture for the PL member is defined and instantiated from the reference architecture.

Example: Figure 6 shows an instance of the reference architecture in Figure 4 which supports the workflow defined by the storyboard in Figure 5.

For the implementation, existing PL assets are reused, new assets that are within the scope of the PL but have not been designed and implemented yet are developed and assets are created to fulfill specific requirements of the current instance only.

2.6 Evolving and Managing the Product Line over Time

The purpose of PuLSE-EM is to monitor and control the evolution of the PL infrastructure as well as to coordinate the activities of the other PuLSE components over time. It gets various workproducts from the different components, consolidates them, and takes care of their evolution and mainte-

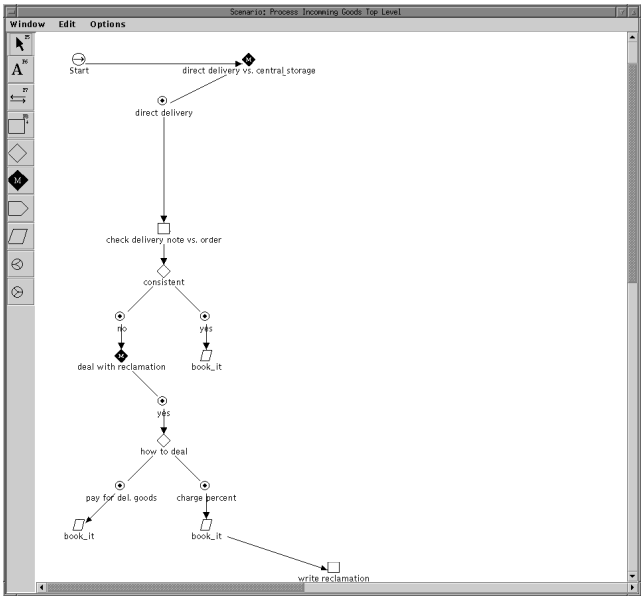

Figure 5. Example - Instantiated Storyboard

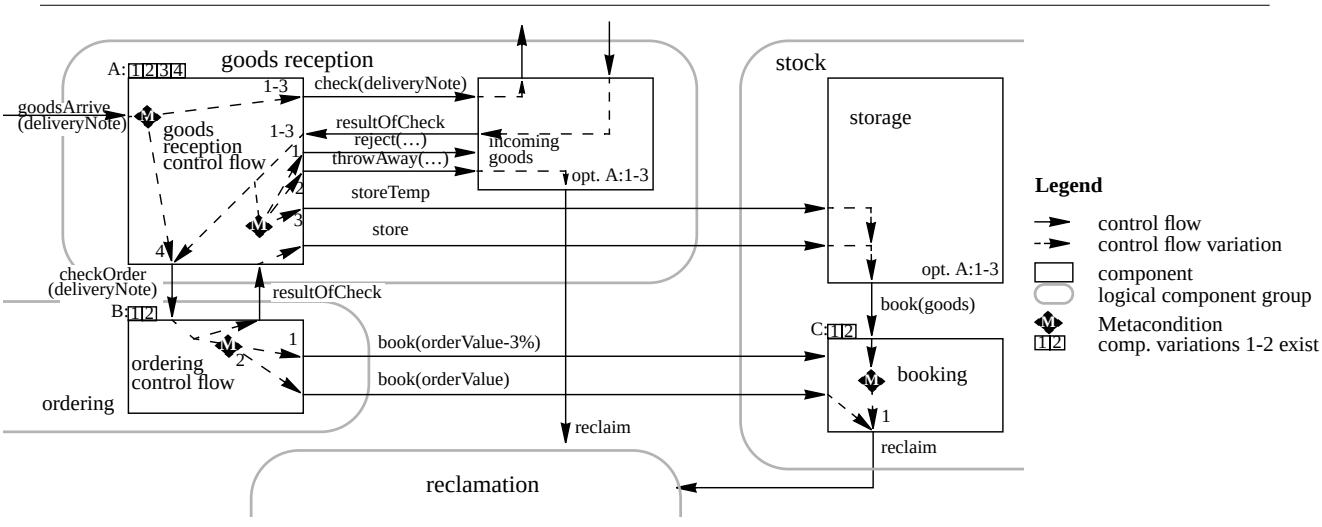

Figure 4. Example - Partial Reference Architecture

nance.

During the application of the other components problems may arise that cause changes to the specifications set by previous components. These changes have ramifications, not necessarily limited to the preceeding component as the following example demonstrates.

Example: During the application of PuLSE-DSSA problems to integrate a required COTS component may turn up. These problems may affect the scope (handled by PuLSE-Eco), which in turn causes changes to the generic storyboards and other CDA workproducts. PuLSE-EM takes care of all these kinds of change requests, consolidates and evaluates them, determines their (potential) ramifications, and restarts the appropriate steps of the respective components where they are handled.

3 Experience from the Application of PuLSE

We have started to apply PuLSE in various domains (text layout systems, CAD systems, human comfort simulation) with different industrial customers. One of them produces case based reasoning (CBR) tools for various purposes. Targeted towards a complete redesign of their software, we applied PuLSE-CDA to describe the application situations of current and anticipated future products. Based on these we started to apply PuLSE-DSSA to derive a common reference architecture for all of these. As the developed infrastructure seems to be promising PuLSE-Eco was introduced to plan more reliably for approaching new market.

Another customer whose civil engineering software leads the german market in the area of composite construction is on the Full Level too. We use PuLSE there with the final goal of leveraging their currently separate products dealing with different yet overlapping aspects of the same domain into an integrated solution with a CAD frontend.

The experience we have made so far is very promising. The following sections highlight some interesting issues encountered:

Different enterprises and domain contexts have strong impact on how a PL should be established and used. PuLSE-BC enables us to customize the other PuLSE components to the specific needs of the enterprise.

To improve the adaptation process in PuLSE-BC, we are tracking baselining profiles from different contexts. Based on these we are developing packages of PuLSE customizations that are pretailored to typical situations.

PuLSE-Eco provides a good basis for communication and it is easy to use. Yet, very large product maps are physically hard to manage. We need to make then hierarchical.

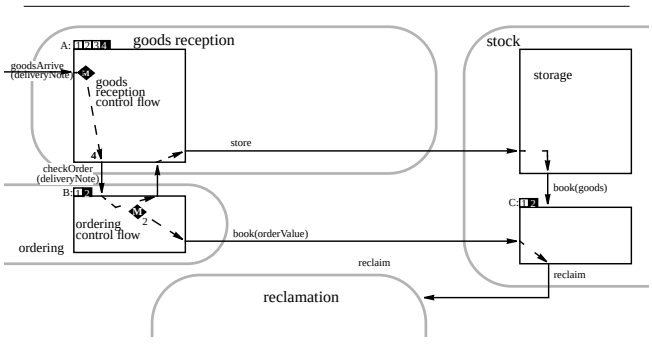

Figure 6. Example - Instantiated DSSA

Domains have different concerns and therefore, require different models to capture their key concepts. It appears to have been the right decision to design PuLSE-CDA to be customizable to the relevant domain abstractions and notations.

It proved to be an advantage of PuLSE-DSSA that it neither requires a specific design methodology nor a special architecture description language or technique. The framework can use whatever idiom established in an enterprise.

PuLSE-I works in a way, that PL assets can be implemented just-in-time when they are needed for a specific product. First applications show clearly that there has to be strong supervision to make sure that assets are developed considering the PL context (i.e., not only the current project).

PuLSE-EM provides an effective mechanism to propagate change requests turning up during the application of PuLSE to the components responsible for handling them. Moreover, we expect that, over time, PuLSE-EM will allow reasoning about the kinds of changes that occur and thus to more effectively evolve the PL infrastructure to future needs.

PuLSE is currently used in about eight enterprises. It is evolving constantly and much remains to be done before it could be considered stable.

4 Conclusions

In this paper, we present a high-level overview of PuLSE, our customizable methodology for the conception and deployment of software product lines. Major characteristics of the PuLSE technical components were demonstrated using an example from our industrial project experience.

Acknowledgments

PuLSE is currently being developed by the following people: Joachim Bayer, Oliver Flege, Roland Laqua, Dirk Muthig, Klaus Schmid, Tanya Widen, and the authors [2]. We thank them for their valuable contribution, especially Klaus for support on Figure 2; Joachim, Dirk, and Tanya for their work on DIVERSITY/CDA; and Oliver for help on the merchandise examples.

References

1. Brownsword, L. and Clements, P. *A Case Study in Successful Product Line Development*. Technical Report CMU/SEI-96-TR-016, Carnegie Mellon, 1996
2. DeBaud et al. *PuLSE — Product Line Software Engineering*. Technical Report, Fraunhofer Institute for Experimental Software Engineering (IESE), Sauerwiesen 6, 67661 Kaiserslautern, 1998.
3. DeBaud et al. *PuLSE: A Methodology to Develop Software Product Lines*. submitted to 21. International Conference on Software Engineering (ICSE) 1999.
4. Foreman, J. *Product Line Based Software Development - Significant Results*, Future Challenges. Proceedings of the Software Technology Conference. April 1996.
5. Mollaghasemi, M. and Pet-Edwards, J. *Making Multiple-Objective Decisions*. IEEE Computer Society, 1997.