# A Comparison of Software Product Family Process Frameworks

Tuomo Vehkomäki, Kari Känsälä

Nokia Research Center
Helsinki, Finland
tuomo.vehkomaki@nokia.com, kari.kansala@nokia.com

**Abstract.** A number of product family process frameworks has been published recently. These frameworks focus on different aspects of product family based development. We have investigated a variety of publicly available product family frameworks and chosen four of the variants for maximum coverage of different viewpoints. We first propose a reference product line process framework. With the help of the reference framework, the chosen source frameworks are correlated and compared at the level of individual activities. Both in the reference framework and in the comparison, we stress domain engineering as one of the most essential activities.

## 1. Introduction

The objective of this study is to create a generic software product line process framework that can be used as a reference model to compare product line approaches known by today's industry. The objective has not been to create another process framework, but a benchmark of existing frameworks. The generic framework is best used to organize references to the actual information sources, such as the compared product family process frameworks, or proprietary process descriptions within a specific industry.

The product line approaches of interest to us represent rather different viewpoints. Therefore the generic framework needs to be comprehensive enough to allow mapping between product line approaches with different coverage.

In our terminology, the Generic Product Line Process (GPLP) covers the actual software development cycle for all levels of granularity: systems, products, platforms, and components. The term Generic Product Line Process Framework (GPLPF) includes GPLP plus supporting process categories i.e. the transition to product line, product portfolio management, and third party product acquisition and subcontracting.

The section 2 of this text introduces the source frameworks that contribute to the generic product line process framework and the comparison. Section 3 describes the generic product line process framework and section 4 compares the source frameworks with the proposed reference framework.

## 2. Source Frameworks

We have initially investigated traditional software and systems engineering frameworks. With emergence of frameworks that explicitly deal with product lines, we have included those frameworks in our comparison.

Using SPICE v2.0 [SPICE96] as a skeleton, an extensive comparison of existing software and systems engineering frameworks was presented in 1997 [Nyström97]. The compared frameworks are listed in Table 1. Based on the comparison and existing software processes in Nokia Business Units, a customized version of SPICE v2.0 called NRC Software Process Framework was developed at Nokia Research Center [Känsälä99]. A typical model of industrial product process categories based on SPICE is illustrated in Figure 1.

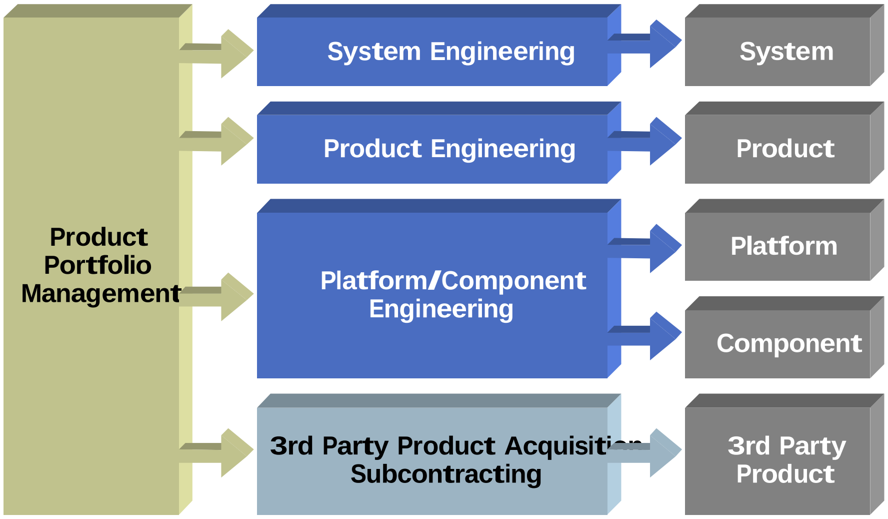
Product Portfolio Management
System Engineering
Product Engineering
Platform/Component Engineering
3rd Party Product Acquisition Subcontracting
System
Product
Platform
Component
3rd Party Product

**Fig. 1.** Typical product process framework. The software processes support development of systems that are composed of four layers: system, product, platform, and component. The depicted framework does not yet include product-line specific activities.

**Table 1.** Software and systems engineering models compared by Nystöm [Nyström97].

| **PF** | **Full name** | **Status / Version** | **Released** |
|---|---|---|---|
| **SPICE** | Software Process Improvement and Capability dEtermination | V2.0 | 1996 |
| **CMM** | SW Capability Maturity Model/ SEI | V1.1 | 1993 |
| **ISO 9000-3** | Guidelines for the Application of ISO 9001 to the Design, Development, Supply, Installation and Maintenance of Computer Software | Draft International Standard | 1996 |
| **ISO 12207** | Software Life Cycle Processes | | 1995 |
| **IEEE 1074** | Standard for Developing Software Life Cycle Processes | | 1995 |
| **J-STD-016** | Standard for Information Technology, Software Life Cycle Processes, Software Development and Acquirer-Supplier Agreement | Interim Standard | 1995 |
| **SE-CMM** | Systems Engineering Capability Maturity Model/ SEI | V1.1 | 1995 |
| **IEEE 1220** | Standard for Application and Management of the Systems Engineering process | Trial-use | 1994 |
| **EIA/IS-632** | Systems Engineering | Interim Standard | 1994 |

The 1997 comparison is used as a background for the comparison of software product line process related models. Starting in the early 90's and more frequently since 1997, several frameworks related to software product lines have been published. A representative set of product line frameworks is listed in Table 2 and summarized in the rest of this section. The listed frameworks are included in the comparison of Section 4.

**Table 2.** Software product line process frameworks of our comparison.

| Framework | Full name [reference] | Status / Version | Released |
|---|---|---|---|
| **GPLP** | Generic Product Line Process (see section 3.) | Initial | Mar-00 |
| **SEI FSPLP** | Software Engineering Institute: Framework for Software Product Line Practice / [Clements99] | V2.0 | Jul-99 |
| **Synthesis, DsE** | Domain-specific Engineering (DsE) [Campbell99] based on Synthesis [RSP93] | Presented in Reuse'99 | Apr-99 |
| **RSEB** | Reuse-driven SW Engineering Business [Jacobson97] | Book / ACM Press | Jun-97 |
| **SPICE, NRC SPF** | Nokia Research Center Software Process Framework [Känsälä99] based on SPICE v2.0 [SPICE96] | V1.1 | May-98 |

### SEI Framework for Software Product Line Practice

The approach used by SEI is to identify foundational concepts underlying software product lines and activities to be considered when creating a product line [Clements99]. The listed practice areas comprise an extensive set of competencies and issues necessary to consider for successful adoption of product line based reuse. The viewpoint supports product line planning and management, rather than gives concrete instructions on implementing specific engineering tasks.

### Synthesis and Domain-specific Engineering

Synthesis [RSP93] by Software Productivity Consortium is an extensive description of processes related to domain engineering. Synthesis also includes creation of process support for the application engineering. Synthesis does not explicitly address transition to product line based reuse but describes two process variants for different levels of organizational reuse capability.

Domain-specific Engineering (DsE) continues from the basis of Synthesis and relies on parallel domain engineering and application engineering activities in the traditional way of domain engineering. In addition to plain domain engineering, Domain-specific Engineering has explicit activities of domain management, process engineering, and project support [Campbell99].

### Reuse-driven Software Engineering Business

Reuse-driven Software Engineering Business (RSEB) [Jacobson1997] describes a systematic model for implementing reuse. The description is tightly coupled with object-oriented analysis and design, the Unified Modeling Language (UML) and layered software architecture. Instructions on how to do analysis, design, implementation, and validation are given and less effort is put on management issues. For a mature organization, the approach may be used as a guide to implement reuse.

The actual process is a derivative of the traditional Domain Engineering/Application Engineering split and has separate activities for

- Application Family Engineering
- Component System Engineering
- Application System Engineering.

Application Family Engineering and Component System Engineering can be considered two separate variations of accustomary domain engineering. Application Family Engineering works at a high level of abstraction to develop a conceptual model and a common layered architecture for all product line members. Component System Engineering works at lower level of abstraction to develop functional building blocks for the layered product platform.

In addition to engineering activities, RSEB also includes an explicit set of activities that support the transition to reuse.

### NRC Software Process Framework

Being based on SPICE, NRC Software Process Framework [Känsälä99] is a traditional software engineering process framework which does not cover the product family dimension i.e. it deals with single systems only. The framework consists of 29 processes partitioned to five categories:

- Customer-supplier process category
- Engineering process category
- Support process category
- Management process category
- Organization process category

Being comprehensive also beyond engineering activities, it complements the product line approaches presented above. The customer-supplier process category supports transition of the software to the customer and its correct operation and use. Together with various maintenance activities, these processes are not covered well by the product line approaches.

## 3. Generic Product Line Process Framework

The comparison of software product family process frameworks is based on a Generic Product Line Process Framework that is described in this section. The generic framework consists of process categories for product line management, domain engineering, application engineering, and third party product acquisition.

Corresponding to the traditional product process framework of Figure 1, the Generic Product Line Process Framework reflects creation of systems that are composed of four layers: system, product, platform, and component.

Components and 3rd party products are parts of a whole. They may be used as building blocks of any of the upper layers. Platforms have a double role: from a product viewpoint, they are components as they are integrated with some application functionality to build products. From the component viewpoint, platforms are similar to products as they typically consist of several components that have been integrated together. Finally, systems are solutions that consist of several products.

The process categories and their relations to each other and to created work products are illustrated in Figure 2. Compared to the previous model of Figure 1, this model replaces the Component/Platform Engineering process category with domain engineering, which may produce reusable assets for all levels of the layered systems. As domain engineering builds competence on the application area, domain engineering can give input to the portfolio management and management of 3rd party products. Domain engineering also interacts directly with the application engineering process groups.

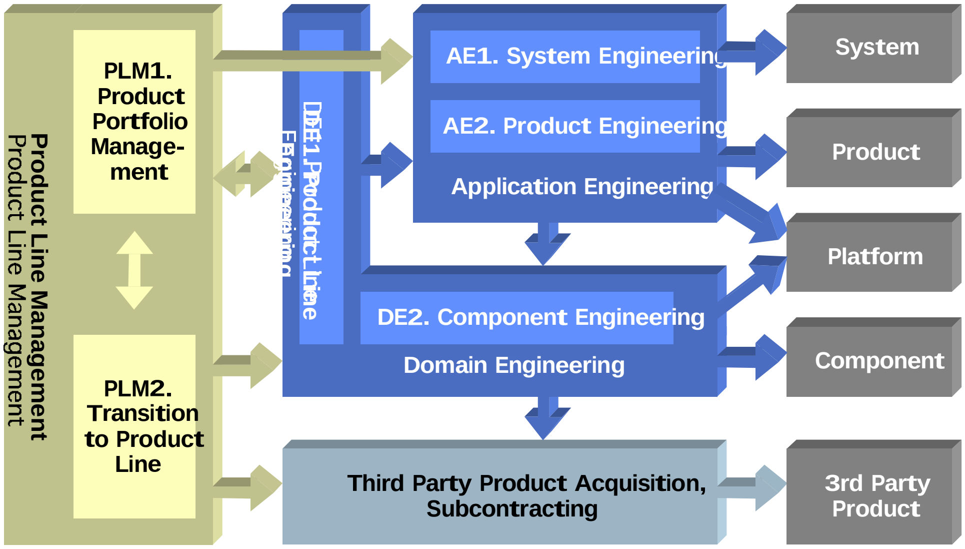
Product Line Management
PLM1. Product Portfolio Manage-ment
PLM2. Transition to Product Line
DE1. Product Line Engineering
AE1. System Engineering
AE2. Product Engineering
Application Engineering
DE2. Component Engineering
Domain Engineering
Third Party Product Acquisition, Subcontracting
System
Product
Platform
Component
3rd Party Product

**Fig. 2.** Process categories of the proposed Generic Product Process Framework. The central part of the figure represents the actual Software Product Line Process.

### 3.1 Activity Groups

| | |
|---|---|
| **Product Line Management (PLM)** | The product line management process category contains activities related to establishing and managing the product line. |

PLM1. Product Portfolio Management

Creates visions and requirements for new products. This includes gathering requirements from the customers, market research, and technology research..

PLM2. Transition to Product Line

The transition process is temporary and not necessary after the product line infrastructure has been established. The transition process includes organizational planning and planning for competence creation. Reuse maturity assessment may be used to determine the organization's current reuse capability [Adoption93].

**Domain Engineering (DE)**

Domain Engineering is the activity to produce reusable assets. Domain Engineering is essentially orthogonal to the layered system-product-platform architecture and supports producing reusable components for all of the layers.

Domain Engineering can play two roles: Product Line Engineering and Component Engineering. In the comparison, however, Domain Engineering is treated as a single activity group.

DE1. Product Line Engineering

Product Line Engineering is the variation of Domain Engineering for the entire product family. Product Line engineering concentrates on analysis of concepts common to all applications and design of common architecture for the complete product line.

DE2. Component Engineering

Component Engineering is the variation of domain engineering for a specific area of functionality or knowledge. Typically these areas represent the organization's technical core competencies. The resulting assets may be reused in all levels of the system-product-platform hierarchy.

**Application Engineering (AE)**

The term Application Engineering refers to the activities that generate new applications utilizing the assets created by Domain Engineering. In our terminology, there are two types of applications that have different creation processes: products and systems.

AE1. System Engineering

System Engineering is the activity to create systems utilizing reusable assets. Systems are solutions that consist of several products. Based on system requirements, System Engineering develops systems by integrating products.

AE2. Product Engineering

Product Engineering is the activity to create products utilizing platforms, components, and other reusable assets. Products may be supplied directly to the end-customers or integrated to compose systems.

**Third Party Product Acquisition and Subcontracting (TPS)**

This activity group creates 3rd party product by acquisition of COTS components or through subcontracting. As the components produced by the Component Engineering activity, 3rd party products may be used in all levels of the layered systems.

## 4. Comparison

This comparison illustrates the coverage of the source frameworks compared to the Generic Product Line Product Process Framework. The comparison also maps the activities of the compared frameworks to the common terminology defined by the generic framework. The activity groups listed above are refined to consist of individual activities that make the rows of the comparison matrix. The columns represent different product line process frameworks. Their individual activities are distributed within the column to match the activities of the generic framework on the right column.

Table 3 shows an overview of the mapping without the names of the individual activities from the compared frameworks. The purpose of this overview is to illustrate which activities of the generic framework have been addressed by each of the compared frameworks.

Table 4 is an extract of the complete mapping to further illustrate details related to the Domain Engineering activity group. Note that RSEB defines two variations of domain engineering: Application Family Engineering and Component System Engineering. This separation corresponds to Product Line Engineering and Component Engineering activities of Figure 2.

For further details on domain analysis techniques, comparisons of plain domain analysis techniques have been published by Arango [Arango93] and by DeBaud and Schmid [DeBaud98].

**Table 3.** Coverage of compared SW product line process frameworks. One asterisk indicates some correspondence and two asterisks indicate good match with the activity named in the left column.

| | **SEI FSPLP** | **Synthesis, DsE** | **RSEB** | **SPICE, NRC SPF** |
|---|---|---|---|---|
| **PLM1. Product Portfolio Management** | | | | |
| Product Line Scoping | ** | ** | ** | |
| Domain Management | ** | ** | ** | |
| **PLM2. Transition to Product Line** | | | | |
| Develop Organizational Strategy | ** | | ** | * |
| Model Current Process | * | * | ** | |
| Develop Product Line Process | * | ** | ** | * |
| Implement Product Line Process | ** | | ** | * |
| Develop Metrics | | | ** | |
| **DE. Domain Engineering** | | | | |
| Domain Scoping | ** | ** | | |
| Domain Analysis | ** | ** | ** | * |
| Domain Verification | | ** | | |
| Mine Assets | ** | | | |
| Domain Design | ** | ** | ** | * |
| Architecture Evaluation | ** | | | |
| Domain Implementation | | ** | ** | ** |
| Integration and Testing | * | ** | ** | ** |
| Domain Support | ** | ** | * | * |
| **AE1. System Engineering** | | | | |
| Analyze Requirements | | | | * |
| Design | | | | * |
| Implement | | | | * |
| Integrate and Test | | | | * |
| Package | | | | * |
| Supply | | | | ** |
| Support | * | | | ** |
| **AE2. Product Engineering** | | | | |
| Analyze Requirements | * | * | ** | ** |
| Design | * | * | ** | ** |
| Implement | | * | ** | ** |
| Integrate and Test | ** | ** | ** | ** |
| Package | | | * | * |
| Maintain | | | | ** |
| **TPS. Third Party Product Acquisition, Subcontracting** | | | | |
| COTS Utilization | ** | | | * |
| Develop and Implement Acquisition Strategy | ** | | | |

| | | | | |
|---|---|---|---|---|
| Subcontractor Management | | | | ** |

**Table 4.** Detailed mapping of activities within Domain Engineering activity group.

| | SEI FSPLP | Synthesis, DsE | RSEB: Application Family Engineering | RSEB: Component System Engineering | SPICE, NRC SPF |
|---|---|---|---|---|---|
| **Domain Scoping** | TMP2. Product Line Scoping | DE.1. Domain Management | | | |
| | | DE.2.1. Domain Definition | | | |
| **Domain Analysis** | SEP1. Domain Analysis | DE.2.2. Domain Specification | AFE1: Analyzing requirements that have an impact on the architecture | CSE1: Capturing requirements focusing on variability | ENG.1 Develop product requirements and design |
| | | | AFE2: Performing robustness analysis | CSE2: Performing robustness analysis to maximize flexibility | ENG.2 Develop SW requirements |
| **Domain Verification** | | DE.2.3 Domain Verification | | | |
| **Mine Assets** | SEP2. Mining Existing Assets | | | | |
| **Domain Design** | SEP3. Architecture Exploration and Definition | DE.2.2.4 Product (Family) Design | AFE3: Designing the layered system coordination | CSE3: Designing the component system | ENG.3 Develop SW design |
| **Architecture Evaluation** | SEP4. Architecture Evaluation | | | | |
| **Domain Implementation** | | DE.3.1. Product (Family) Implementation | AFE4: Implementing the architecture as a layered system | CSE4: Implementing the component system | ENG.4 Implement SW design |
| | SEP5. COTS Utilization | | | | |
| **Integration and Testing** | SEP6. Software System Integration | DE.2.3 Domain Verification | AFE5: Testing the layered system coordination | CSE5: Testing the component system | ENG.5 Integrate and test SW |
| | | DE.4.1 Domain Validation | | CSE6: Final packaging of the component system for reuse | ENG.6 Integrate and test product |
| **Domain Support** | OMP3. Training | DE.4.2 Domain Delivery | | | |
| | OMP5. Launching and Institutionalizing a Product Line | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | TMP1. Data Collection, Metrics and Tracking | DE.3.2. Process Support Development | TRA6: Continuous process improvement | | |
| | TMP3. Configuration Management | | | | ENG.7 Maintain product and SW |

## 5. Summary and Outlook

We have presented a Generic Product Line Process Framework and compared four publicly available product process approaches with the help of this generic model. The developed framework reflects the product structure of our industry and the compared product family process frameworks represent viewpoints that we consider important.

The comparison shows that the coverage of actual software engineering activities is rather complete by all of the compared frameworks. Deficiencies exist in management and other supporting categories and in the acquisition of 3rd party products. The system engineering field is only covered by NRC SPF. The SEI FSPLP covers all the other categories well. The weaknesses of Synthesis are the transition process and the 3rd party product acquisition process but it has the best coverage of domain engineering activities. RSEB does not cover 3rd party product acquisition. NRC SPF has the best coverage of customer support and maintenance activities but lacks several of reuse-oriented activities.

The first public version of the comparison is based on the work at Nokia Research Center. Further developmend of the model is to continue in an European ESAPS (Engineering Software Architectures, Processes and Platforms for System-Families) project during 2000-2001 [ESAPS].

## Acknowledgements

The authors would like to thank prof. Jukka Paakki for his comments on the manuscript.

## References

[Adoption93] Reuse Adoption Guidebook, SPC-93051-CMC, Software Productivity Consortium, Herndon, VA, 1993.

[Arango94] G. Arango, Domain Analysis Methods, in Software Reusability (W. Shaefer, R. Prieto-Diaz, and M. Matsumoto, eds.), Ellis Horwood, 1994.

[Campbell99] Grady H. Campbell, Jr., Reuse-driven Process Improvement. The first European Annual Conference on Reuse, London, April 1999.

[Clements99] Clements P, Northrop L, et.al., A Framework for Software Product Line Practice – Version 2.0, SEI, July 1999, (http://www.sei.cmu.edu/plp/framework.html).

[DeBaud98] Jean-Marc DeBaud and Klaus Schmid, A Practical Comparison of Major Domain Analysis Approaches - Towards a Customizable Domain Analysis Framework, In Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering, 1998.

[ESAPS] Engineering Software Architectures, Processes and Platforms for System-Families, European EUREKA/ITEA project, (http://www.esi.es/esaps/).

[Jacobson97] Jacobson I, Griss M, and Jonsson P, Software Reuse. Architecture, Process and Organization for Business Success, ACM Press, New York, June 1997.

[Känsälä99] Känsälä, Kari . Practices for Managing a Corporate-wide SPI Programme, European SEPG Conference, Amsterdam, The Netherlands, 7-10 June 1999.

[Nyström97] Nyström T, Comparison of Software Reference Processes Definitions, Master's thesis, HUT, 1997. 66 p.

[RSP93] Reuse-driven Software Processes Guidebook, SPC-92019-CMC, Software Productivity Consortium, Herndon, VA, November 1993.

[SPICE96] SPICE, Software Process Assessment Part 5: An assessment model and indicator guidance ISO/IEC/JTC1/SC7/WG10/N111, V2.0, October 1996, (http://www.sqi.gu.edu.au/spice/).