Integrating Theory and Practice: Applying the Quality Improvement Paradigm to Product Line Engineering

Michael Stark
Goddard Space Flight Center

Introduction

My assertion is that not only are product lines a relevant research topic, but that the tools used by empirical software engineering researchers can address observed practical problems. Our experience at NASA has been there are often externally proposed solutions available, but that we have had difficulties applying them in our particular context. We have also focused on return on investment issues when evaluating product lines, and while these are important, one can not attain objective data on success or failure until several applications from a product family have been deployed.

The use of the Quality Improvement Paradigm (QIP) [1] can address these issues by

1. Planning an adoption path from an organization's current state to a product line approach.
2. Constructing a development process to fit the organization's adoption path
3. Evaluation of product line development processes as the project is being developed.

The QIP consists of the following 6 steps:

1. Characterize the project and its environment
2. Set quantifiable goals for successful project performance
3. Choose the appropriate process models, supporting methods, and tools for the project
4. Execute the process, analyze interim results, and provide real-time feedback for corrective action
5. Analyze the results of completed projects and recommend improvements
6. Package the lessons learned as updated and refined process models.

Figure 1 shows the QIP in detail. The iterative nature of the QIP supports an incremental development approach to product lines, and the project learning and feedback provide the necessary early evaluations.

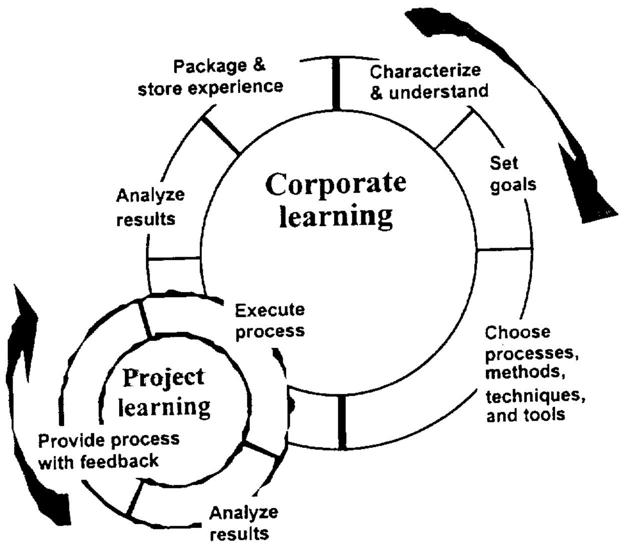

Figure 1. The Quality Improvement Paradigm

My current research is into tailoring the QIP to fit the needs of product line development. It carries out the same steps as are defined for the QIP, but each of these steps is specialized to address product line issues. My tailoring of the QIP, Model-based Analysis for Product Line Engineering (MAPLE), proposes four stages:

- Adoption requirements, which encompasses the first two steps of the QIP,
- Process development , which corresponds to the third step,
- Process evaluation, which corresponds to the next two steps, and
- Process deployment, which encompasses the packaging step of the QIP.

The original intent of this research is to address how to construct or tailor a product line development approach to fit a particular organization's needs. However, the first two steps of the QIP can be adapted to the larger issue of organizational change.

Any planned change should start with a characterization of the current organization and of the goals to be attained by the change, as a precondition for planning an adoption path. In principle, the same characterization that drives product scoping should be able to set requirements on the processes that are used to develop and exploit a product line.

The ultimate goal of my research is to provide an approach to developing and refining processes that operates concurrently with product development, as shown below in Table1. This has already been proposed for developing application engineering processes [2], I am proposing that this be done for all aspects of product line engineering.

Process Engineering	**Product development**
Adoption Planning	
Process development	Product Scoping
Process Evaluation • Project learning • Post-analysis	Product Development • Domain engineering • Architecture • Implementation • Test
Process Deployment • As core assets	Product deliveries • Core assets • Applications

Table 1. Concurrent Product Line Engineering

I will focus the rest of this position paper on how to characterize an organization' current state and its goals. These characterizations will drive the development of adoption strategies, the requirements for and constraints on development processes, and the determination of which features will be included in the product line.

Characterizing the organization

The MAPLE approach to characterizing the organization's current and goal states is to define a hierarchical classification scheme that incorporates key issues. The same scheme is used for both the current and goal states. The top level components of the hierarchy are the organization's *assets*, its *stakeholders*, its *business model*, and its *processes*.

Each of these top level areas can be decomposed in turn, and assigned characterization functions that are similar in concept to those found in PuLSE-Eco. [3].

Assets

The assets discussed here are candidate work products that can either be incorporated into a new product line, or may inform the development of a product line. These can be decomposed into domain assets such as requirements and algorithm documents, architecture assets such as existing system designs and available COTS products, implementation assets such as source code and make files, and verification assets such as test plans, procedures and results.

Each of these types of assets may be characterized on an ordinal scale that measures a level of trust in its reusability:

0 – asset would need to be replaced

1 – asset may be salvageable

2 – asset is part of a tested reuse library

3 – asset is part of an already existing product line.

As a product line evolves, the core assets developed in previous cycles can be factored into the decisions for what to do next – hence the inclusion of product line assets in this scale.

A second characteristic that one can start to assess is the extent of modification or tailoring needed to reuse an asset. However, enough details to complete this characterization may not be available until the scoping is done, or even until commonality analysis is done during product development.

Stakeholders

Characterizing the stakeholders may be the most difficult of the 4 areas, as the focus is on people, rather than products or documented processes. However, this is a critical area to the success or failure of a product line. For example, the Generalized Support Software (GSS) project at GSFC met all its stated goals, yet has fallen into disuse. One of the key reasons for this is that one set of stakeholders, the flight dynamics analysts who would develop mission requirements using core assets, were not sufficiently considered by the GSS development team and their management [4].

The first issue to be addressed is to identify *all* the stakeholders. In the case of GSS, the analysts were known to be important, but in principle it is possible to omit a stakeholder.

There are ways to directly characterize stakeholders that are currently used in empirical software engineering. One common characterization is a set of ordinal scales measuring experience: total software development experience, experience in a role, or experience with a particular organization.

However, there are some more complex questions to answer. One is how does one characterize the interactions between stakeholder roles? Does the current organizational structure support the interactions that are needed to build and exploit a product line?

Another issue is to assess the gap between current stakeholder roles and what would be necessary to succeed in a product line environment. To do this would require substantial understanding of how stakeholders carry out their work product development.

All of these questions are areas where empirical research may provide very practical answers.

Business Model

The business model can be split into two main components: *product metrics* such as size, complexity, cost, and cycle time that quantify the development process; and *market drivers* such as cost, time-to-market, or quality that indicate priorities perceived by the organization.

The product metrics can be used to estimate return on investment. Ideally, an organization already has a metrics baseline that can be used. If not, data may be scavenged from sources such as project status reports, or estimated through interviews with managers and other key personnel. In any case, the product metrics associated with the goal states will be estimates that must be validated as a product line project is being developed.

In the area of market drivers, quality can in turn be decomposed into factors such as reliability, maintainability, safety or portability. Each of these quality factors, along with cost and cycle time, can be characterized and aggregated into a benefit function, as in [3], to provide relative weights to market drivers.

Process

There are two aspects of process that should be measured. The first is process maturity in general, the second is the degree of product line technology usage.

Software process maturity is often measured using the SEI's CMM or CMMI model. If an organization has already done this, the results can be used in this context.

If an organization has not performed a full process assessment using one of these models, one can still construct a model that is similar but not nearly as exhaustive. For example, the CMMI continuous model [5] defines the broad categories of engineering, project management, process management, and support (configuration management, quality assurance,...). One can also define an ordinal scale similar to CMMI maturity levels as follows:

1. reliance on the abilities of individuals
2. reliance on project based processes
3. reliance on organizationally defined processes
4. quantifiably predictable processes
5. continuously improving processes

Each of the 4 broad categories can be characterized on this scale through informal interviews. The results will not be as accurate as a full CMM or CMMI assessment, but the goal here is to gain enough information to plan a product line adoption strategy. A full assessment would be appropriate if overall process improvement is considered as part of the strategy, but this may not always be the case.

A measure of the degree of product line technology use can be built from the following three product line characteristics:

- The product line represents an investment to be amortized across multiple products
- There is an asset development process that is separate and distinct from the product development process
- There is a well-defined procedure for assembling products from core assets.

An ordinal scale for reuse technology level can be constructed as follows:

1. no significant reuse approach
2. scavenging from previous projects
3. First of above characteristics is satisfied: there is multiple project investment.
4. First two characteristics are satisfied
5. All three characteristics are satisfied.

The Software Engineering Institute defines a product line as a "A group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission area." [6] Using this definition, level 3 and above would represent the use of a product line approach. There may be organizations where a level 3 approach might be more appropriate than a level 5 one; in other words this scale measures the satisfaction of a set of properties without an assumption that satisfying more of them is better.

Using the classification hierarchy

Figure 2 below shows a partial decomposition of the classification hierarchy. The above discussion has discussed some of the theoretical modeling considerations for constructing the hierarchy, this section discusses some of the practical reasons for such a hierarchy.

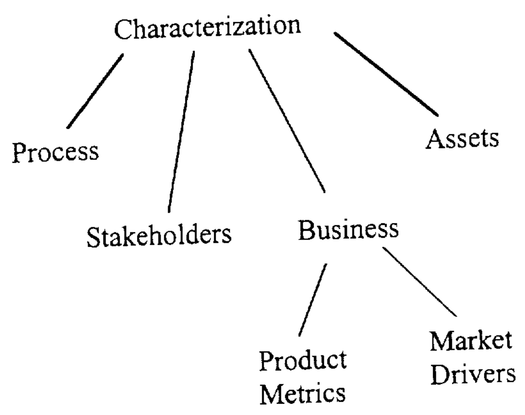

Figure 2. Classification Hierarchy

First, the hierarchy can be decomposed to whatever level is needed. A small organization might write an informal description of both its current state and its goal state, allocating a page to each of the four top level areas. A larger organization in a safety critical area might use the full hierarchy, and add more detailed information on safety issues to the market drivers, the process and the stakeholders.

Second, a tree hierarchy is easy to understand and to edit. This is useful when applying an iterative learning approach such as the QIP. As more is learned during product line development, the key issues can be elaborated in more detail and the less relevant parts of the model can be either removed or represented in less detail. The trees representing an organization's current and desired state may also help visualize the areas that need to be addressed in an adoption plan.

A practical approach to using such hierarchies would be to provide an example hierarchy for organizations to use as a starting point for their planning, and allow the organization to tailor the example to fit its needs.

Future Research

There are two areas of research that follow from this work, both of which have extremely practical application. The first is the application of a classification hierarchy to adoption planning. Some of the issues to be addressed here are:

- How do the current and desired organizational characteristics map into an adoption strategy?
- Which of the characteristics have the greatest impact on process and product development?

The second research area is the development of MAPLE as an approach to concurrent engineering of processes within a product line approach. My approach here is to develop each MAPLE step in turn, analyze the plausibility of the approach through a retrospective analysis applying it to GSS, then validating it through use on a product line project.

References

1. Basili, V. and S. Green, "Software Process Evolution at the SEL", IEEE Software, pp. 58-66, July 1994.
2. Weiss, David M. and Chi Tau Robert Lai *Software Product-Line Engineering: A Family Based Software Development Approach*, Addison-Wesley, 1999.
3. DeBaud, J-M. and K. Schmid, "A Systematic Approach to Derive the Scope of Software Product Lines," pp. 34-43. *Proceedings of the 21st ICSE*. Los Angeles, CA, May 1999.
4. Condon, S., et al., "Evolving the Reuse Process at the Flight Dynamics Division (FDD) Goddard Space Flight Center", Proceedings of the 21st Annual Software Engineering Workshop, Greenbelt, MD, December 1996.
5. *CMMI^{SM} for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1) Continuous Representation* CMU/SEI-2002-TR-011, March 2002
6. Software Engineering Institute: "Glossary, A Framework for Software Product Line Practice - Version 3.0", http://www.sei.cmu.edu/plp/frame_report/glossary.htm.