Carnegie Mellon
Software Engineering Institute

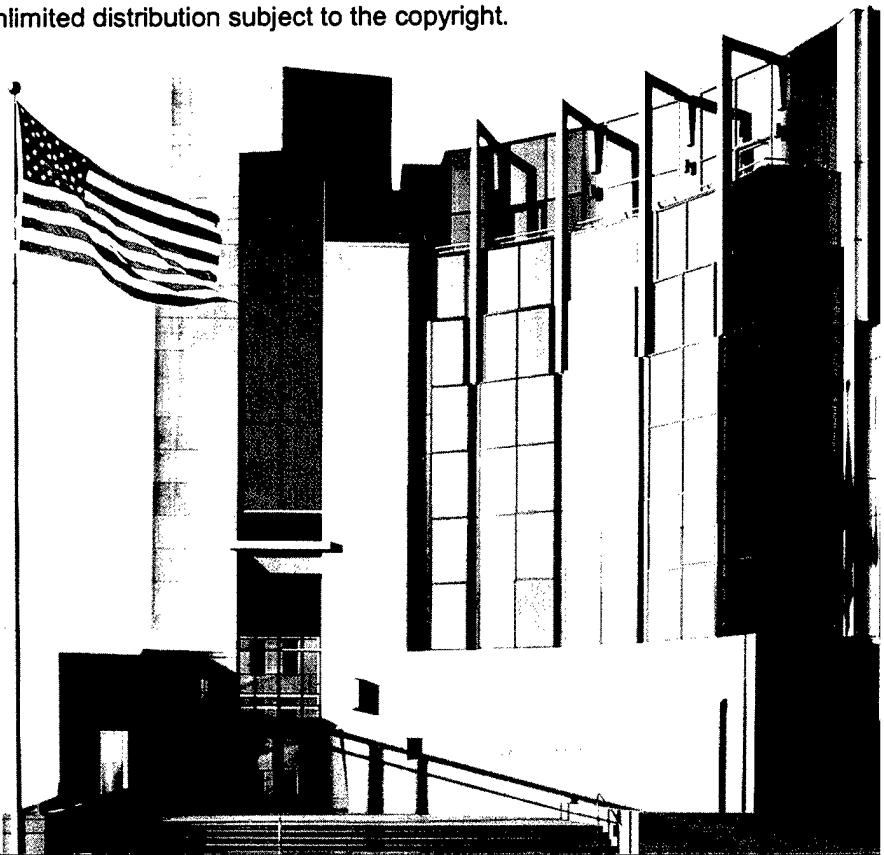
Software Process Improvement and Product Line Practice: CMMI and the Framework for Software Product Line Practice

Lawrence G. Jones
Albert L. Soule

July 2002

Product Line Practice Initiative

Technical Note
CMU/SEI-2002-TN-012



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Software Process Improvement and Product Line Practice: CMMI and the Framework for Software Product Line Practice

Lawrence G. Jones
Albert L. Soule

July 2002

Product Line Practice Initiative

Technical Note
CMU/SEI-2002-TN-012

Contents

List of Figures

List of Tables

Abstract

Many organizations report dramatic benefits from the adoption of software product line practice. Organizations that have established software engineering process discipline are better poised to succeed with product lines. While we acknowledge that there are different paths to successful process discipline, in this technical note, we concentrate on approaches based on the Capability Maturity Model Integration (CMMI) models. We describe practices that are most crucial to product line success. While some of these relate directly to the CMMI models process areas, others are uniquely important to product lines.

In this technical note, we first present fundamental concepts of software product lines. We then describe important product line practices as they have been documented in *A Framework for Software Product Line Practice* (framework). We next present an overview of the CMMI models, followed by a description of the general relationships between the framework and CMMI models. We amplify this comparison with a detailed example showing the relationship between configuration management practices in CMMI and in the framework. We conclude by describing the ways in which organizations can build upon their process improvement efforts to achieve success with product lines and realize additional benefits through the use of both technologies.

1 Introduction

New methods and technologies are often developed along different paths. This is natural since they frequently have different purposes and emphases. Additionally, newer technologies can naturally incorporate older technologies, while the reverse will not happen without an explicit revision cycle including this focus. As new technologies prove useful and gain acceptance, there is a need to compare and relate them to other accepted technologies. This technical note relates two important software technologies: software engineering process discipline and software product line practice. Product line practice includes process discipline at its heart, but the reverse is not generally true. This technical note will focus on the relationship of two SEI models—one for product line practice and one for software engineering process discipline—and the benefits of applying both to software development.

Motivating product line technology is the increasing realization among organizations that they can no longer afford to develop multiple software products one product at a time. They are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain a market presence, compensate for an inability to hire, leverage existing resources, and achieve mass customization. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. These organizations are adopting a product line approach for their software systems.

Meanwhile, the 1990s saw the widespread application of manufacturing process principles to the development of software. Software engineering process discipline, based on the quality concepts pioneered by Crosby [Crosby 79], Deming [Deming 86], and others, has resulted in dramatic benefits. Organizations typically report return on investment (ROI) figures of between 5:1 and 8:1 resulting from successful software process improvement (SPI) programs. Additional quantified benefits include productivity gains, improved time-to-market gains, significantly improved project planning estimations, and reduced defect rates. Other observed benefits include improved employee morale, less employee turnover, and increased customer satisfaction.

Software engineering process discipline has a significant relationship to product line practice. Product line practice is strategic in nature. A strategic effort requires more coordination, discipline, and commonality of approach than a more independent effort. Dependencies within an organization are greater, and predictability and quality become even more critical. Process discipline can provide the basis for a strategic effort and has proven that it can provide better predictability and quality. Thus, an organization with a culture of process discipline is much better poised for product line success.

In this technical note, we explore the relationship between software product line practice, as defined by the Framework for Software Product Line Practice (framework), and software engineering process discipline, as defined by the Capability Maturity Model® Integration℠ (CMMI℠) models. In Section 2, we present fundamental concepts of software product lines. Then, we describe important product line practices and how they have been documented in the framework. In Section 3, we present an overview of the CMMI models and their structure and contents. Given this basis, in Section 4, we describe the way software engineering process discipline and software product line practice complement each other in practice. Section 5 highlights general relationships between the framework and CMMI models and the degree to which CMMI process areas support framework product line practice areas. In Section 6, we amplify this general comparison with a detailed example showing the relationship between configuration management practices in CMMI and in the framework. We conclude by describing the ways in which organizations can build upon their process improvement efforts to achieve success with product lines and realize additional benefits through the use of both technologies.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.
℠ CMMI and CMM Integration are service marks of Carnegie Mellon University.

2 Product Line Practice

2.1 What Is a Product Line?[1]

A *software product line* is a set of software-intensive systems sharing a common, managed set of features that satisfy the needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [Clements 02].

According to Clements and Northrop, this definition is consistent with the definition traditionally given for any product line, but it adds more; it puts constraints on the way the systems in a software product line are developed because substantial production economies can be achieved when the systems in a software product line are developed from a common set of assets in a prescribed way. The product line architecture is central to the set of core assets used to construct and evolve the products in the product line. This common, product line software architecture[2] capitalizes on commonalities in the implementation of the line of products and provides the structural robustness that makes the derivation of software products from software assets economically viable.

Each product in the product line is formed by taking applicable components from the base of common assets, tailoring them as necessary through preplanned variation mechanisms such as parameterization or inheritance, adding any new components that may be necessary, and assembling the collection according to the rules of the product line architecture.

By product line practice, we mean the systematic use of software assets to assemble, instantiate, generate, or modify the multiple products that constitute a product line. Building a new product (system) becomes more a matter of assembly or generation than creation. For each software product line, there is a predefined guide, called a production plan, which specifies the exact product building approach[3].

Product line practice involves strategic, large-grained reuse as a business enabler. The key concepts are

---

1 This section draws substantially from the work of Clements and Northrop [Clements 02].

2 The software architecture of a computing system is the structure or structures of the system that consist of software components, the externally visible properties of those components, and the relationships among them [Bass 98].

3 Chastek, G. & McGregor, J. *Guidelines for Developing a Product Line Production Plan.* Pittsburgh, PA: Software Engineering Institute, to be published.

- **the use of a common asset base** (with the architecture being the pivotal asset)
- **in the production** (according to a predefined and documented production plan)
- **of a set of related products** (whose scope has been clearly defined and validated with a business case).

According to Clements and Northrop, at its essence fielding a software product line involves *core asset development, product development* from the core assets, and *management* to staff, orchestrate, and coordinate the entire product line effort. Figure 1 shows these essential activities. The arrows signify the high degree of iteration involved and the fact that there is no prescribed order as to how these activities take place.

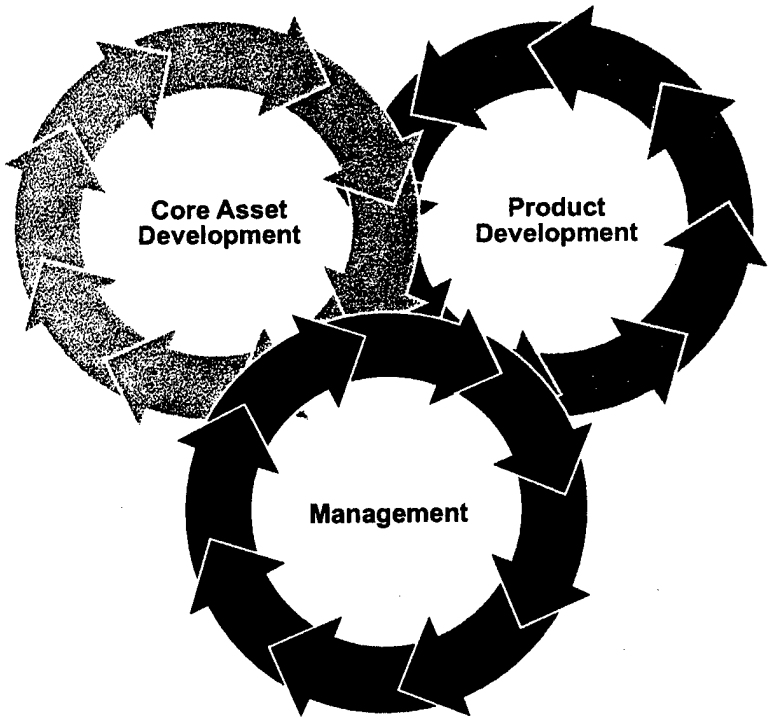


Figure 1: Essential Activities for Product Line Practice [Clements 02]

The goal of the core asset development activity (the left side of Figure 1) is to establish a production capability for products. Inputs to the development of core assets are product constraints found by analyzing the similarities of and differences between current and projected products; the production constraints such as those found in a technical architecture; a production strategy for the assets; an inventory of preexisting assets; and styles, patterns, and architectural frameworks. The outputs are the core assets, a preliminary list of the products they will support (the product line scope), and a production plan for how the core assets will be used in the development or acquisition of products.

On the right side of Figure 1, individual products are developed from the core assets using the production plan that has been established. Product requirements are developed and

refined with the existing core assets in mind, and products that systematically reuse the core assets are output.

There is a strong feedback loop between the core assets and products. Core assets are refreshed as new products are developed, and in fact, the earliest products may well be the source of the core assets to begin with. In addition, the value of the core assets is realized through the products that are developed from them. As a result, core assets are made more generic by considering potential new products on the horizon. There is a constant need for strong and visionary management to invest the resources in the development of the core assets and to develop the cultural change needed to view new products through the filter of the core assets.

2.2 The Software Product Line Practice Framework

The Software Engineering Institute has captured the essential product line activities and practices in *A Framework for Software Product Line Practice*. This framework is available as a Web-based, evolving document and is targeted primarily at members of organizations who are in a position to make or influence decisions regarding the adoption of product line practices. Version 4.0 of the framework is published in the book titled *Software Product Lines: Practices and Patterns* [Clements 02]. Version 3.0 [Clements 00] can be found on the SEI's Web site, and Version 5.0 will be available on the Web in fall 2002.

There are essential practices in a number of specific areas that are required to produce the core assets and products in a product line and to manage the process at multiple levels. The framework describes the essential practice areas for software engineering, technical management, and organizational management, where these categories represent disciplines rather than job titles. A *practice area* is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line. For individual practice areas, the framework provides

- an introductory description of the practice area
- aspects of this practice area that are peculiar to product lines
- how this practice area is applied to core asset development
- how this practice area is applied to product development
- specific practices in this practice area
- risks in this practice area
- references

The software engineering practice areas include those practices necessary to apply the appropriate technology to create and evolve both core assets and products as follows:

- Architecture Definition
- Architecture Evaluation
- Component Development
- COTS Utilization
- Mining Existing Assets
- Requirements Engineering
- Software System Integration
- Testing
- Understanding Relevant Domains

The technical management practice areas include those management practices necessary to engineer the development and evolution of the core assets and products as follows:

- Configuration Management
- Data Collection, Metrics, and Tracking
- Make/Buy/Mine/Commission Analysis
- Process Definition
- Scoping
- Technical Planning
- Technical Risk Management
- Tool Support

Organizational management refers to the management of the business issues that are visible at the enterprise level, as opposed to those at the project level. Organizational management includes those practice areas necessary to position the enterprise to take fullest advantage of the product line capability. The organizational management practices include

- Building a Business Case
- Customer Interface Management
- Developing an Acquisition Strategy
- Funding
- Launching and Institutionalizing
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management

- Structuring the Organization
- Technology Forecasting
- Training

3 CMMI

3.1 Capability Maturity Models

Many organizations have had success by basing their software engineering process discipline efforts on the Capability Maturity Model (CMM) for Software [Paulk 95]. The CMM (hereafter referred to as the SW-CMM to differentiate it from subsequent CMMs) is a model that contains the essential elements of effective processes for software development. The wide acceptance of this model is evidenced by large annual Software Engineering Process Group (SEPG) conferences in North America, Europe and Asia. In addition, there are more than 100 Software Process Improvement Network (SPIN) chapters worldwide. While the SW-CMM is not the only model for guiding software process improvement based on process discipline,[4] its widespread acceptance makes it a de facto standard, and we will focus on CMMI-based process discipline in this technical note.

The success of the SW-CMM spawned the development of several maturity models for other disciplines, for example, systems engineering, software acquisition, workforce practices, and integrated product and process development. While these models proved valuable to many organizations, the application of multiple models became expensive and complicated. To address this problem, the Capability Maturity Model Integration (CMMI) project was initiated [Phillips 02]. This project resulted in a complete product suite including three models [SEI]:

- CMMI for Systems Engineering/Software Engineering (CMMI-SE/SW, V1.1)

  This model addresses the development of products and services (in particular software-intensive systems) and provides the foundation for the other two models.

- CMMI for Systems Engineering/Software Engineering / Integrated Product and Process Development (CMMI-SE/SW/IPPD, V1.1)

  This model builds upon CMMI-SW/SE by introducing integrated product teams and the context they need to operate effectively toward achieving a systematic, timely collaboration of relevant stakeholders throughout the life of the product.

- CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1)

  This model builds upon CMMI-SW/SE/IPPD with additional focus on proactively acquiring products and services from external sources.

[4] See Zahran for several examples including ISO 15504, ISO 9001, and BOOTSTRAP [Zahran 97].

Because the CMMI models are scheduled to eventually replace the SW-CMM model, the rest of this note will focus on the CMMI models.

3.2 CMMI Models

A major organizing element for all CMMI models is the *process area*. A process area is a group of related activities that are performed collectively to achieve a set of goals. In the context of these models, processes refer to "what to do" rather than "how to do it." A process area specifies goals that describe the result of successful application and practices that describe required (and expected) activities to achieve those goals. Some goals and practices are specific to the process area; others are generic and apply across all process areas. These *generics* describe essential ways in which a process can be *institutionalized.* Institutionalization refers to a process's degree of repeatability, standardization, and sophistication of control.

Structurally, each CMMI model comes in two representations: a *staged representation* and a *continuous representation.* Each representation organizes process areas and the application of the generics to them differently. These two representations are really just different views into the same content. A staged representation may be said to focus on the organization's processes as a whole, to provide a roadmap for process improvement with proven predefined groupings of process areas, and to provide an easy migration path from the SW-CMM. A continuous representation may be said to focus on improvement to individual process areas chosen to align with specific organizational needs and to provide an easy migration path from Electronic Industries Alliance Interim Standard (EIA/IS) 731 [Menezes 02].

Unique to the staged representation is the major organizing element of the *maturity level.* A maturity level is an indicator of the extent to which a set of processes is implemented and institutionalized. Maturity levels recognized by the CMMI are

1. Initial: The organization has informal process control; no process areas are institutionalized.
2. Managed: Here, relative to basic project management, processes are standardized within individual projects.
3. Defined: This level is characterized by process standardization across projects.
4. Quantitatively Managed: Quantitative management of processes is the hallmark of this level.
5. Optimizing: Continual process improvement occurs at this level.

Maturity levels also provide a recommended order for improving processes within an organization. Maturity levels and their process area groupings for CMMI-SE/SW are shown in Figure 2.

Level	Focus	Process Area
5 Optimizing	Continuous Process Improvement	Organizational Innovation and Deployment Causal Analysis and Resolution
4 Quantitatively Managed	Quantitative Management	Organizational Process Performance Quantitative Project Management
3 Defined	Process Standardization	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management for IPPD Risk Management Integrated Teaming Integrated Supplier Management Decision Analysis Resolution Organizational Environment for Integration
2 Managed	Basic Project Management	Requirements Management Project Planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management
1 Initial	N/A	N/A

Figure 2: CMMI-SE/SW/IPPD/SS Staged Representation Process Areas

The continuous representation uses the concept of capability level to measure process improvement within individual process areas. Capability levels represent the application of the generics to a single process area and indicate the degree of institutionalization of the process area. Apart from the application of generics to an individual process area, continuous representation models do not recommend a particular implementation order. Also, though they recognize relationships within general CMMI categories (see Figure 3), the models generally treat process areas as independent. While in theory this implies freedom of implementation order when using a continuous representation, key associations among the process areas preclude totally arbitrary ordering or implementations.

Category	Process Areas
Process Management	Organizational Process Focus Organizational Process Definition Organizational Training Organizational Process Performance Organizational Innovation and Deployment
Project Management	Project Planning Project Monitoring and Control Supplier Agreement Management Integrated Project Management for IPPD Risk Management Integrated Teaming Integrated Supplier Management Quantitative Project Management
Engineering	Requirements Management Requirements Development Technical Solution Product Integration Verification Validation
Support	Configuration Management Process and Product Quality Assurance Measurement and Analysis Decision Analysis and Resolution Organizational Environment for Integration Causal Analysis and Resolution

Figure 3: CMMI Process Area Categories

Experienced implementers often take advantage of the strengths of both representations. For example, while relying on a staged ordering as a "first cut" prioritization, you might vary the basic implementation ordering based on business needs or "where it hurts most."

Finally, when we talk about CMMI-SE/SW/IPPD, V1.1 and CMMI-SE/SW/IPPD/SS, V1.1, we need to consider that the model implementation now extends beyond the engineering organization to more overtly include other corporate functions such as procurement, marketing, human resources, and support in the product or system development effort. As in the characterization of the organizational implementation of the framework described above, the addition of these domains requires a strategic perspective on process improvement having a perspective across these functions within the organization. Therefore, most of the same attributes that underpin a strategic effort such as product line management (coordination, discipline, commonality of approach, etc.) are supported by a robust set of cross-functional process best practices that help organizations better manage dependencies and provide for improvements in predictability and quality.

4 Applying Product Line Practice and Software Process Improvement

Clements and Northrop explain the fundamental connection between product line practice and software engineering process discipline [Clements 02] as follows:

> An essential aspect of software engineering is the discipline it requires for a group of people to work together cooperatively to solve a common problem. Defined processes set the bounds for each person's roles and responsibilities so that the collaboration is a successful and efficient one. ... If software engineering is about a group of people working together to solve a problem cooperatively, then product line software engineering requires cooperation in spades. ... In fact, organizations that do not have a strong process culture will find deploying a successful product line a perilous proposition.

This strong, complementary connection has been proven in practice by leading companies in various domains. It is not an overstatement to say that successful product line practice requires a significant degree of process discipline. We will provide two examples: the Boeing Company, the world-class aircraft manufacturer, and Cummins Engine Inc., the world's largest manufacturer of commercial diesel engines above 50 horsepower.

The benefits of SPI based on software engineering process discipline are well documented and accepted [Ferguson 99, Goldenson 95, Zahran 97]. Organizations successful in moving from SW-CMM Maturity Level 1 to Level 2 and from Level 2 to Level 3 typically report ROI figures of 5:1 to 8:1. In particular, John Vu of the Boeing Company has substantial data supporting this from a number of projects [Vu 97, Vu 00b]. Recently Vu has studied the improvements in high maturity organizations [Vu 00a]. His studies show that the benefits of SPI applied to a single-product focus tend to level off between SW-CMM Maturity Level 3 and Maturity Level 4. However, when this improvement includes a shift to a product line approach, the productivity increase is significant, as much as 70% improvement. A key point is that this also resulted in a shift to a focus on business benefits rather than merely technical benefits.

Another case in point is Cummins Engine Inc. [Clements 02]. The role of process discipline as an enabler is a pervasive theme in Cummins' product line approach. For example, some of the first steps that the Cummins product line champion undertook were to

- establish a standard Controls Software workflow process
- form a standard hardware process group

- launch a team to establish common development and configuration management tools and processes

The results of this successful product line approach included

- dramatically reduced product cycle time
- an all time high for software quality
- high customer satisfaction
- an increased number of successful projects
- productivity improvement of 360%

While it is evident that process discipline and product line practice go hand in hand, Cummins estimates that process improvement alone resulted in a benefit-to-cost ratio of between 2:1 and 3:1. It further estimates that software product line practice, applied in addition to software process discipline, resulted in a benefit-to-cost ratio of 10:1.

Figure 4 summarizes the complementary nature of software engineering process discipline and software product line practice. It also illustrates a "multiplier" effect, namely that the two technologies can operate in concert to achieve business goals through a complementary focus on both process and product. This focus makes it natural to extend process discipline beyond just the engineering processes. This explicitly brings in non-technical processes and organizational aspects emphasized in the framework but not in the CMMI models.

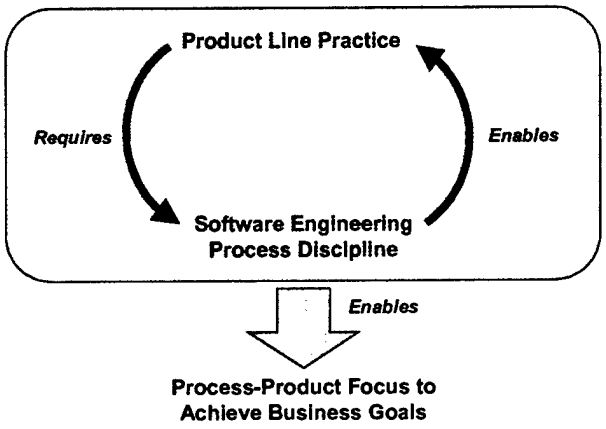


Figure 4: Process and Product Line Relationships

Given the complementary nature of software engineering process discipline and software product line practice, the question is, "how do we coordinate them and maximize the benefits of each?" The bodies of knowledge in the CMMI models and the framework provide one basis for a coordinated approach. We next look at how they relate.

5 CMMI Models and the Framework for Software Product Line Practice

There are several ways to compare the framework with CMMI models. In this section, we draw both broad and detailed comparisons of framework practice areas and CMMI process areas.

Table 1 contains some broad comparisons between the CMMI models and the framework. First, they each have a different focus. CMMI models support generic process improvement in a product development environment and are intended to be independent of any particular development methodology. This contrasts with the framework, which is specifically focused on a product line technical approach. A natural consequence of this difference is that CMMI models try to avoid providing "how to" information, while the framework contains specific examples of how to implement the product line practice areas.

Next, consider the areas of coverage. While it will be more apparent when we compare process areas and practice areas, we may say that the CMMI models contain much greater emphasis on process and project management, and the framework contains much greater emphasis on organizational management.

Both the CMMI models and the framework are supported by diagnostic methods to determine the state of organizational practice; CMMI uses the Standard CMMI Appraisal Method for Process Improvement (SCAMPI[SM]); the framework uses the Product Line Technical Probe [Clements 02].

While acceptance of the framework is growing, it is relatively new and, thus, has not had time to become a de facto standard. While the same may be said about the CMMI models, a direct ancestor, the SW-CMM, is certainly a de facto standard.

Finally, the staged versions of CMMI models incorporate the concept of a maturity level to provide a broad roadmap for implementation. The framework has no such concept. Experience has shown that organizational contexts (as may be revealed by a probe) differ so greatly that implementation priorities are best determined using the concept of *software product line practice patterns* [Clements 02]. Patterns are a way of expressing common context and problem-solution pairs.

[SM] SCAMPI is a service mark of Carnegie Mellon University.

Table 1: General CMMI-Framework Comparisons

Area of Comparison	CMMI	Framework
Focus	Generic – process improvement	Prescriptive for a specific technical approach
Contains "how to" information	No	Yes
Coverage	Process Management Project Management Engineering Support	Software Engineering Technical Management Organizational Management
Diagnostic method	Appraisal	Probe
De facto standard	Yes (SW-CMM)	No
Maturity Levels	Yes (staged)	No
Capability Levels	Yes (continuous)	No

Having made these broad comparisons, we now proceed to a more detailed look. The most appropriate units for detailed comparison are between the CMMI process areas and the framework's practice areas. How do the CMMI process areas[5] compare to the 29 framework practice areas? While both cover similar subjects, the emphases are different. Roughly speaking, a CMMI process area describes where an organization should have *processes*, whereas a product line practice area describes where an organization should have *expertise* (which sometimes includes process expertise). Having said that, Table 2 draws some high-level associations between practice areas and process areas.

In Table 2, process area names in bold provide fairly direct support for the corresponding practice areas, while others are less strongly related. The CMMI process areas of Process and Product Quality Assurance, Organizational Process Focus, Organizational Process Performance, Quantitative Project Management, Causal Analysis and Resolution, and Organizational Innovation and Deployment do not correspond to any software product line practice areas.

[5] There are 22 process areas in CMMI-SE/SW, 24 process areas in CMMI-SE/SW/IPPD, and 25 in CMMI-SE/SE/IPPD/SS.

Table 2: Associations Between Software Product Line Practice Areas and CMMI Process Areas

Software Product Line Practice Areas	CMMI Process Areas
Software Engineering Practice Areas	
• Architecture Definition	Technical Solution
• Architecture Evaluation	Verification
• Component Development	Technical Solution
• COTS Utilization	Supplier Agreement Management Technical Solution Integrated Supplier Management
• Mining Existing Assets	(none)
• Requirements Engineering	Requirements Development
• Software System Integration	Product Integration
• Testing	Verification Validation
• Understanding Relevant Domains	(none)
Technical Management Practice Areas	
• Configuration Management	**Requirements Management** **Configuration Management**
• Data Collection, Metrics, and Tracking	**Measurement and Analysis** Project Monitoring and Control Integrated Project Management for IPPD
• Make/Buy/Mine/Commission Analysis	**Decision Analysis and Resolution** **Technical Solution** Supplier Agreement Management Integrated Supplier Management
• Process Definition	Organizational Process Definition
• Scoping	(none)
• Technical Planning	**Project Planning**
• Technical Risk Management	**Risk Management**
• Tool Support	(none)
Organizational Management Practice Areas	
• Building a Business Case	(none)
• Customer Interface Management	Integrated Project Management for IPPD Integrated Teaming
• Developing an Acquisition Strategy	Supplier Agreement Management Integrated Supplier Management
• Funding	(none)
• Launching and Institutionalizing	(none)
• Market Analysis	(none)
• Operations	(none)
• Organizational Planning	Project Planning
• Organizational Risk Management	Risk Management
• Structuring the Organization	Organizational Environment for Integration Integrated Teams
• Technology Forecasting	Organizational Innovation and Deployment
• Training	**Organizational Training**

This table notwithstanding, bear in mind that any comparison between a CMMI process area and a software product line practice area is weak. Practice areas and process areas are fundamentally different. Even, when at first glance, they appear to cover the same topic, similar names do not mean they cover the same ground. Practice areas also extend the realm of their coverage into the situation where product lines are the goal, and this is not a focus of

the process areas. Just because your organization has institutionalized the CMMI process area of Configuration Management, this does not mean that you have mastered the practice area of "Configuration Management" for software product lines. We will illustrate this point in detail in the next section.

6 A Detailed Example: Configuration Management

Software systems and product line development organizations use configuration management practices to establish and maintain control of the work products, and changes to work products, throughout the product life cycle. From both the framework and CMMI perspective, configuration management is a key "infrastructure" activity that is fundamental to the project's success.

From the CMMI perspective, a configuration management process should achieve three specific goals:

1. Baselines of identified work products are established.
2. Changes to the work products under configuration management are tracked and controlled.
3. The integrity of baselines is established and maintained.

The CMMI model goes on to define specific practices[6] that an organization must perform in order to achieve each of these goals. For example, to establish baselines, the organization must

- Identify the configuration items to be placed under configuration management control.
- Establish and maintain a configuration management and change management system.
- Create or release baselines for internal use or delivery to the customer.

Accomplishing each of these specific practices requires an organization to undertake a detailed set of steps and produce specific work products. For example, identifying the configuration items involves a selection process, assigning an identifier, specifying important characteristics, and determining when the item is to be placed under configuration management. The output of these steps is a list of discrete configuration items that constitute the configuration baseline.

Of course, from the CMMI perspective, to accomplish an institutionalized configuration management process an organization must

[6] At this level of detail, there is a terminology clash between the CMMI models and the framework. In CMMI models, a specific practice is an activity that is considered important in achieving an associated specific goal. In the framework, a specific practice is an example of a particular way that organizations have accomplished the work associated with a practice area.

- Train people.
- Assign responsibility to perform these activities.
- Schedule the activities and provide funding.
- Provide other resources (e.g., tools and equipment).

The model also recommends a configuration management plan, as well as periodic management and quality reviews of the performance of the configuration management activities against the plan.

By comparison, because framework practice areas almost always describe activities that are essential for any successful software development, the framework *assumes* that the critical role of configuration management for software product line development must be fulfilled by the organization, then it identifies practices that an organization must adopt to successfully develop and manage a software product line. From a software product line perspective, the challenge for the organization with respect to configuration management is the ability to manage the complexity of the software product line.

So while the CMMI defines configuration management processes in terms of *what* to do, the framework provides guidance in the form of *how* to actually do configuration management for software product lines, and the attributes necessary for a solid configuration management system for software product lines. For example, the framework covers issues associated with the complexity of software product line development and characterizes them in terms of

1. versions of configuration items compared to versions of each item, for each product
2. separate management of configuration items compared to a single, unified configuration management process
3. control of the configuration while core assets are being developed and used by multiple team members simultaneously
4. the robustness of the configuration management tool and its ability to support product line development.

The configuration management mission as stated in the framework is " ...allowing the rapid reconstruction of any version of any product, which may have been built using various versions of the core assets and development/operating environment *plus* various product-specific artifacts" [Clements 02]. To elaborate, the tools, processes, and environments for product line configuration management must support the following capabilities:

- support for parallel development such that the same item can be worked on for different products by different team members
- support for distributed engineering so that the integrity of multiple libraries of the same configuration items can be controlled when the development and maintenance sites are not colocated

- build and release management that supports the team's need to release assets to product developers in addition to releasing completed products into the test environment and ultimately to the customer
- more robust change management that includes a greater degree of care because changes may affect the entire product line
- support for configuration and workspace management that carries development and test environment information as part of the attributes of each configuration item
- process management that defines the life cycle for each configuration item and the associated management and control "rules"

It is this last item that provides us the needed link between the framework and the CMMI models. A configuration management system robust enough to manage the complexity of a typical software product line initiative would be difficult to implement without a solid process capability to sustain it. So, while the CMMI addresses configuration management capability in a generic fashion (i.e., the ability to support any software development—product line or otherwise), the framework practices give "life" to the subject by providing the lens we need to see the other dimension of the issue—the "expertise" required to implement configuration management for software product lines.

Referring back to Table 2, it should now be evident why we say that the CMMI Configuration Management process area provides "fairly direct" support (indicated by being in bold type) for the framework's "Configuration Management" practice area. It should also be evident that the framework extends the realm of the CMMI models.

A similar analysis is possible for the other process-area-to-practice-area associations in Table 2. Readers wishing to make their own detailed comparisons should examine the "Aspects Peculiar to Product Lines" section for each practice area described by Clements and Northrop [Clements 02].

7 Conclusions

We have established that software engineering process discipline as specified in the CMMI models provides an important foundation for software product line practice. We have also shown that even with a solid process foundation, more work is required for ultimate success with software product lines. We will conclude with some general recommendations about which CMMI process areas provide a good basis for product line practice and how the two CMMI model representations support a software product line approach.

As we have noted, it is not appropriate to prescribe a "one size fits all" adoption approach for software product line practice. However, we still may conclude that it would be "very useful" for an organization to achieve CMMI Capability Level 2 (continuous representation) in at least the following process areas:

- Requirements Management
- Project Planning
- Configuration Management
- Requirements Development

Recall that Maturity Level 2 and Capability Level 2 generally represent institutionalization at the *project* level. Because of the coordination required across traditional project boundaries, we may say that it would be even more useful to standardize these process areas at the organizational level. This implies achievement of Capability Level 3 for these process areas.

Does this imply that software product line practice is supported better by one representation over the other? Not really. The continuous representation supports software product line practice by allowing a focus on a minimum set of essential processes as determined by the organization's context and goals. Since broad process discipline is ultimately needed, an approach based on the staged representation provides an incremental way to achieve this broad foundation.

In summary, software engineering process discipline is a very helpful foundation for software product line practice. However, success in software product lines requires mastery of many other essential practice areas. Neither is a substitute for the other, but the CMMI models and the software product line practices together can help guide an organization to the state necessary to achieve product line success.

References

[Bass 98] Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice*. Reading, MA: Addison-Wesley, 1998.

[Clements 00] Clements, P. & Northrop, L. *A Framework for Software Product Line Practice, Version 3.0* [online]. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 2000. <http://www.sei.cmu.edu/plp/framework.html>.

[Clements 02] Clements, P. & Northrop, L. *Software Product Lines: Practices and Patterns*. Reading, MA: Addison-Wesley, 2002.

[Crosby 79] Crosby, P. *Quality is Free*. New York, NY: McGraw-Hill, 1979.

[Deming 86] Deming, W. *Out of the Crisis*. Cambridge, MA: MIT Center for Advanced Engineering, 1986.

[Ferguson 99] Ferguson, P. et. al. *Software Process Improvement Works!* (CMU/SEI-99-TR-027, ADA371804). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, November 1999. <http://www.sei.cmu.edu/publications/documents/99.reports/99tr027/99tr027abstract.html>

[Goldenson 95] Goldenson, D. & Herbsleb, J. *After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors that Influence Success* (CMU/SEI-95-TR-009, ADA302225). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, August 1995. <http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.009.html>

[Menezes 02] Menezes, W. "To CMMI or Not to CMMI: Issues to Think About." *Crosstalk 15*,2 (February 2002): 9-11.

[Paulk 95] Paulk, M. et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley, 1995.

[Phillips 02] Phillips, D. "CMMI Version 1.1: What Has Changed?" *Crosstalk 15*, 2 (February 2002): 4-6.

[SEI] CMMI Product Suite. <http://www.sei.cmu.edu/cmmi/products>

[Vu 97] Vu, J. "People, Process, Technology : Stuff that Works." *Proceedings of SEPG 97* (Software Engineering Process Group) (CD-ROM). San Jose, CA, March 17-20, 1997 Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[Vu 00a] Vu, J. "Findings of the Managing Software Innovation and Technology Change Workshop," *Proceedings of SEPG 2000* (Software Engineering Process Group)(CD-ROM). Seattle, WA, March 20-23, 2000. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[Vu 00b] Vu, J. "The SEPG from Level 1 to Level 5." Proceedings of SEPG 2000 (Software Engineering Process Group)(CD-ROM). Seattle, WA, March 20-23, 2000. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[Zahran 97] Zahran, S. *Software Process Improvement: Practical Guidelines for Business Success*. Reading, MA: Addison-Wesley, 1997.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2002	Final

4. TITLE AND SUBTITLE
Software Process Improvement and Product Line Practice: CMMI and the Framework for Software Product Line Practice

5. FUNDING NUMBERS
F19628-00-C-0003

6. AUTHOR(S)
Lawrence G. Jones, Albert L. Soule

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER
CMU/SEI-2002-TN-012

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited, DTIC, NTIS

12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

Many organizations report dramatic benefits from the adoption of software product line practice. Organizations that have established software engineering process discipline are better poised to succeed with product lines. While we acknowledge that there are different paths to successful process discipline, in this technical note, we concentrate on approaches based on the Capability Maturity Model Integration (CMMI) models. We describe practices that are most crucial to product line success. While some of these relate directly to the CMMI models process areas, others are uniquely important to product lines.

In this technical note, we first present fundamental concepts of software product lines. We then describe important product line practices as they have been documented in *A Framework for Software Product Line Practice* (framework). We next present an overview of the CMMI models, followed by a description of the general relationships between the framework and CMMI models. We amplify this comparison with a detailed example showing the relationship between configuration management practices in CMMI and in the framework. We conclude by describing the ways in which organizations can build upon their process improvement efforts to achieve success with product lines and realize additional benefits through the use of both technologies.

14. SUBJECT TERMS	15. NUMBER OF PAGES
CMMI, Product Line Practice	35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL