Capability Maturity Model Integration (CMMI) V1.3 and Architecture-Centric Engineering

SATURN Conference
May 17, 2011
San Francisco, CA

Dr. Lawrence G. Jones
Dr. Michael Konrad

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-2612

Software Engineering Institute | Carnegie Mellon

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
17 MAY 2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Capability Maturity Model Integration (CMMI) V1.3 and Architecture-Centric Engineering	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Capability Maturity Model Integration (CMMI) is a process improvement approach that provides the essential elements of effective processes that ultimately improve organizational performance. It is estimated that over 1,200,000 people worldwide work in organizations that have conducted process appraisals using CMMI. Early versions of the CMMI paid little attention to architecture, agility, product lines, and other modern engineering practices. A significant change to the CMMI V1.3 models (released in November 2010) is the new emphasis on the role of architecture in the design process. This presentation will address the basics of architecture-centric engineering and show where and how these practices are now reflected in the CMMI models. Topics will include ? process improvement and CMMI ? structure of the CMMI models ? essential architecture-centric engineering practices ? architecture-centric engineering practices in CMMI V1.3 ? additional modern engineering practices in CMMI V1.3

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**49**	

Presentation Learning Outcomes

After completing this presentation, attendees should

- know that a process discipline is a powerful enabler of product quality
- be familiar with the structure and purpose of CMMI models
- be familiar with essential architecture-centric engineering activities
- know where architecture-centric activities and work products are described in CMMI V1.3
- know where to find out more about architecture-centric engineering practices and CMMI V1.3

Presentation Outline

CMMI V1.3 – Overview and Context for Modern Engineering Practices Changes

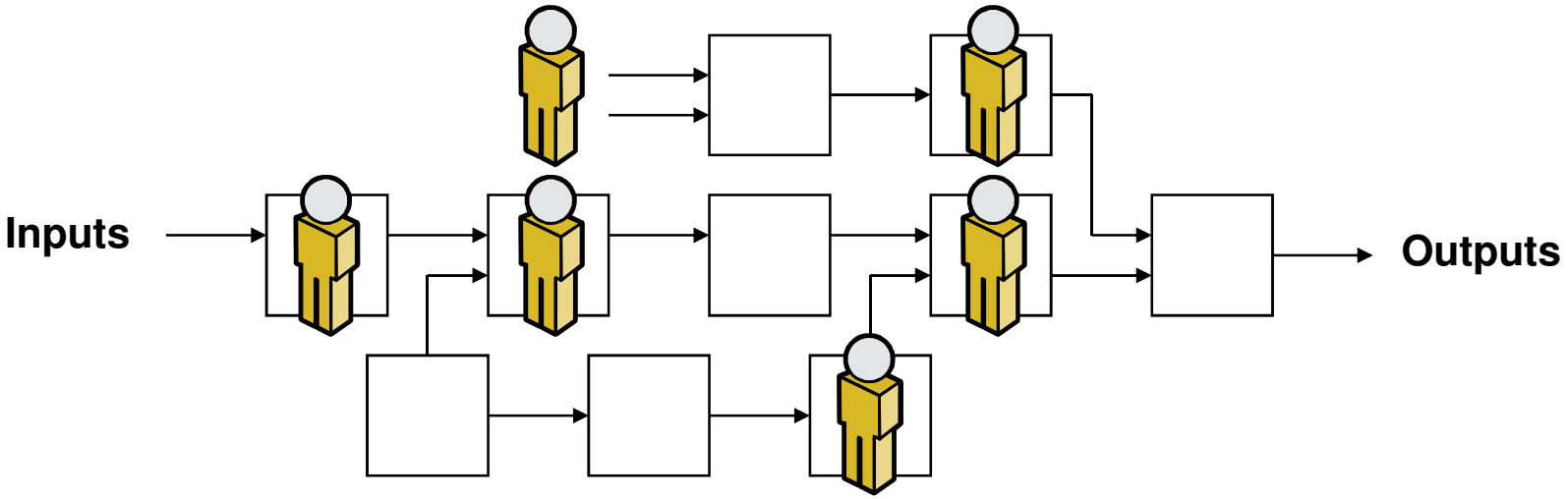
Essential Architecture Practices

Where Are the Architecture-Centric Practices in CMMI V1.3?

Conclusion

What Is a Process?

A **process** is a set of interrelated activities, which transform inputs into outputs, to achieve a given purpose.



Process Improvement flows from and extends the general management theories developed over the past ~50 years (Juran, Deming, Crosby, etc.)

Process! Are You Serious?

You're going to

- stifle my creativity!
- bog us down with bureaucracy!

It doesn't have to be that way.

Yes, I'm Serious.

Process discipline

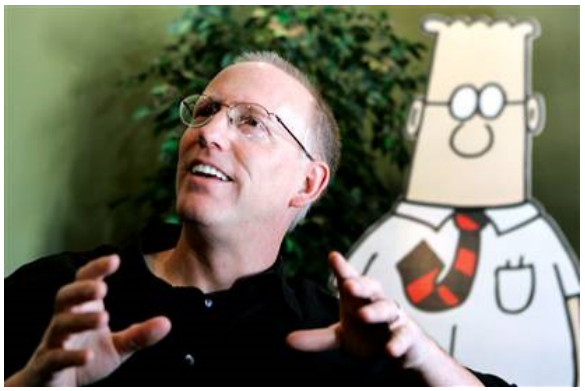
- helps coordinate team efforts
  - prevents tripping over each other
  - can pay for itself
- doesn't have to be heavyweight and bureaucratic
- is central to agility

If you've had a bad experience,

please remember …

"There is no idea so good that it can't be poorly implemented."

Scott Adams



CMMI in a Nutshell

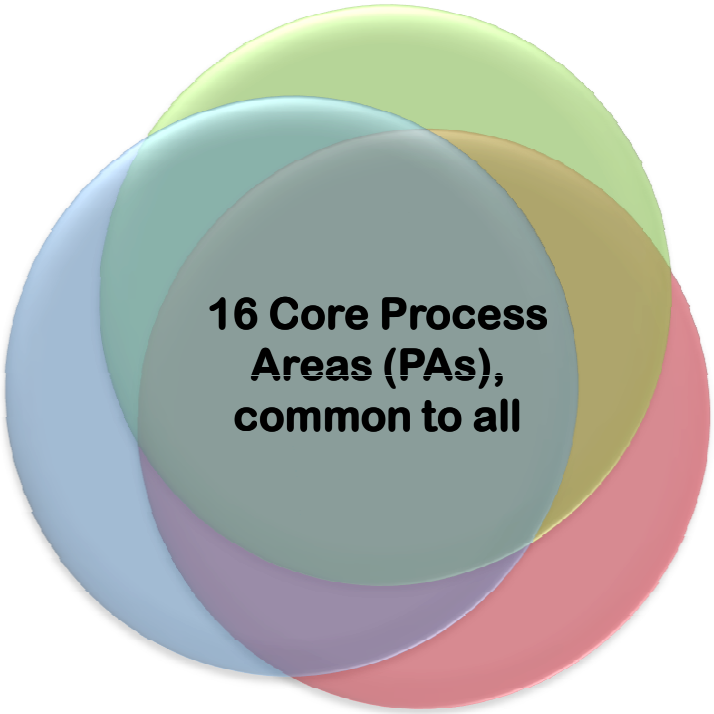
CMMI is a collection of *characteristics of effective processes* that provides guidance for improving an organization's processes and ability to manage the development, acquisition, and maintenance of *products* or *services*.

CMMI organizes these practices into structures that help an organization

- assess its processes
- establish priorities for improvement
- implement these improvements
- learn what works and make further changes to improve performance

"Improving processes for better products"

CMMI Models for Three Constellations



CMMI-SVC
CMMI-SVC provides guidance for those providing services within organizations and to external customers.

CMMI-DEV
CMMI-DEV provides guidance for measuring, monitoring and managing development processes.

CMMI-ACQ
CMMI-ACQ provides guidance to enable informed and decisive acquisition leadership.

CMMI-DEV PAs by Category

Process Management

Organizational Innovation and Deployment (OID)
Organizational Process Definition (OPD)
Organizational Process Focus (OPF)
Organizational Process Performance (OPP)
Organizational Training (OT)

Support

Causal Analysis and Resolution (CAR)
Configuration Management (CM)
Decision Analysis and Resolution (DAR)
Measurement and Analysis (MA)
Process and Product Quality Assurance (PPQA)

For the V1.3 release, REQM was moved from "Engineering" to "Project Management."

Project Management

Integrated Project Management (IPM)
Project Monitoring and Control (PMC)
Project Planning (PP)
Quantitative Project Management (QPM)
Requirements Management (REQM)
Risk Management (RSKM)
(+) Supplier Agreement Management (SAM)

Engineering

Product Integration (PI)
Requirements Development (RD)
Technical Solution (TS)
Validation (VAL)
Verification (VER)

Some CMMI ABCs

The models are built to apply to both systems engineering and software engineering.

The process areas are crafted to be independent of a life-cycle model.

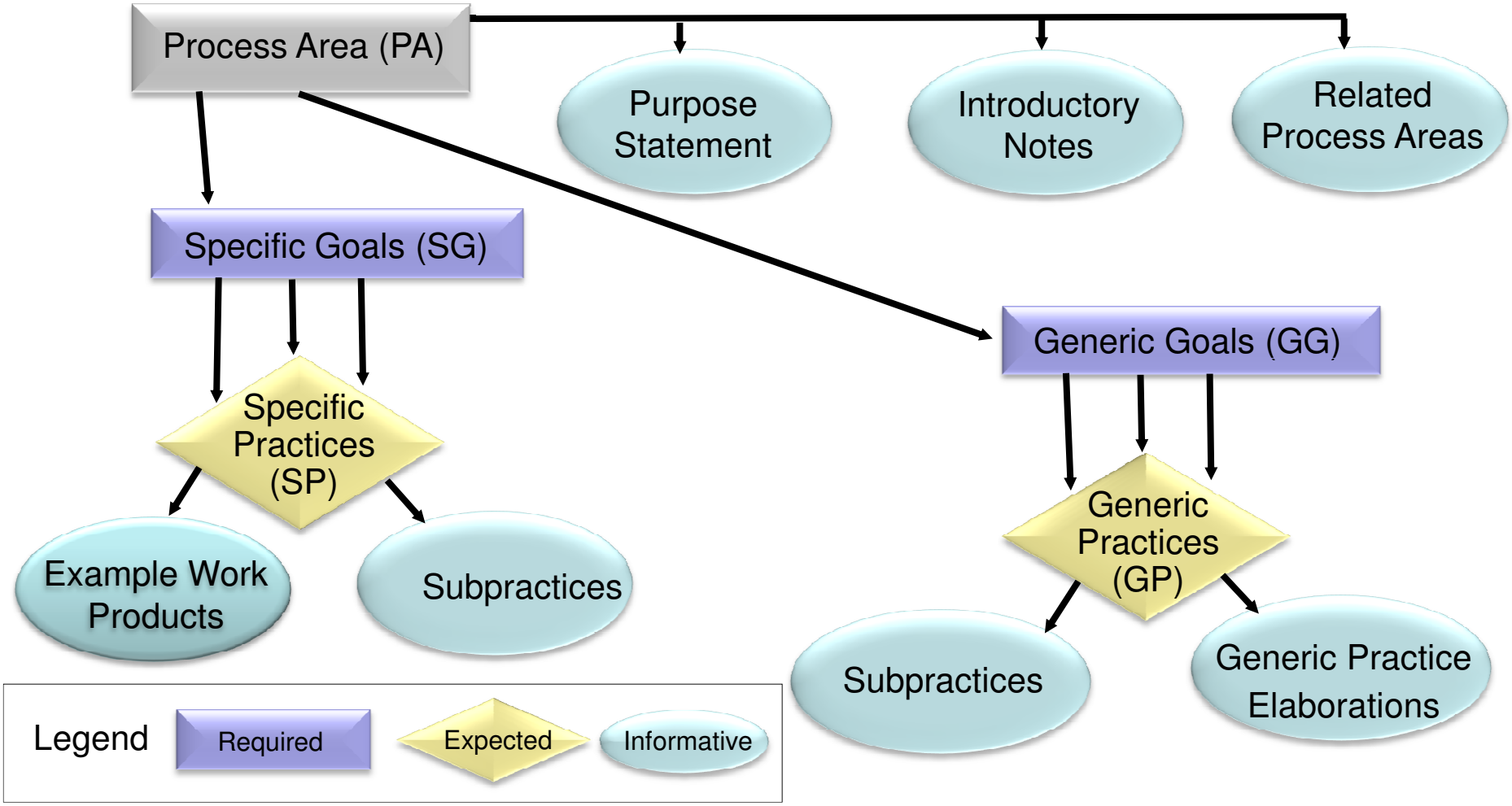
- Engineering process areas should be interpreted as applying to engineering at any level of design.
  - Think of the process areas as being “callable” at any point from high-level design to detailed design.

The models support both *staged* and *continuous* representations.

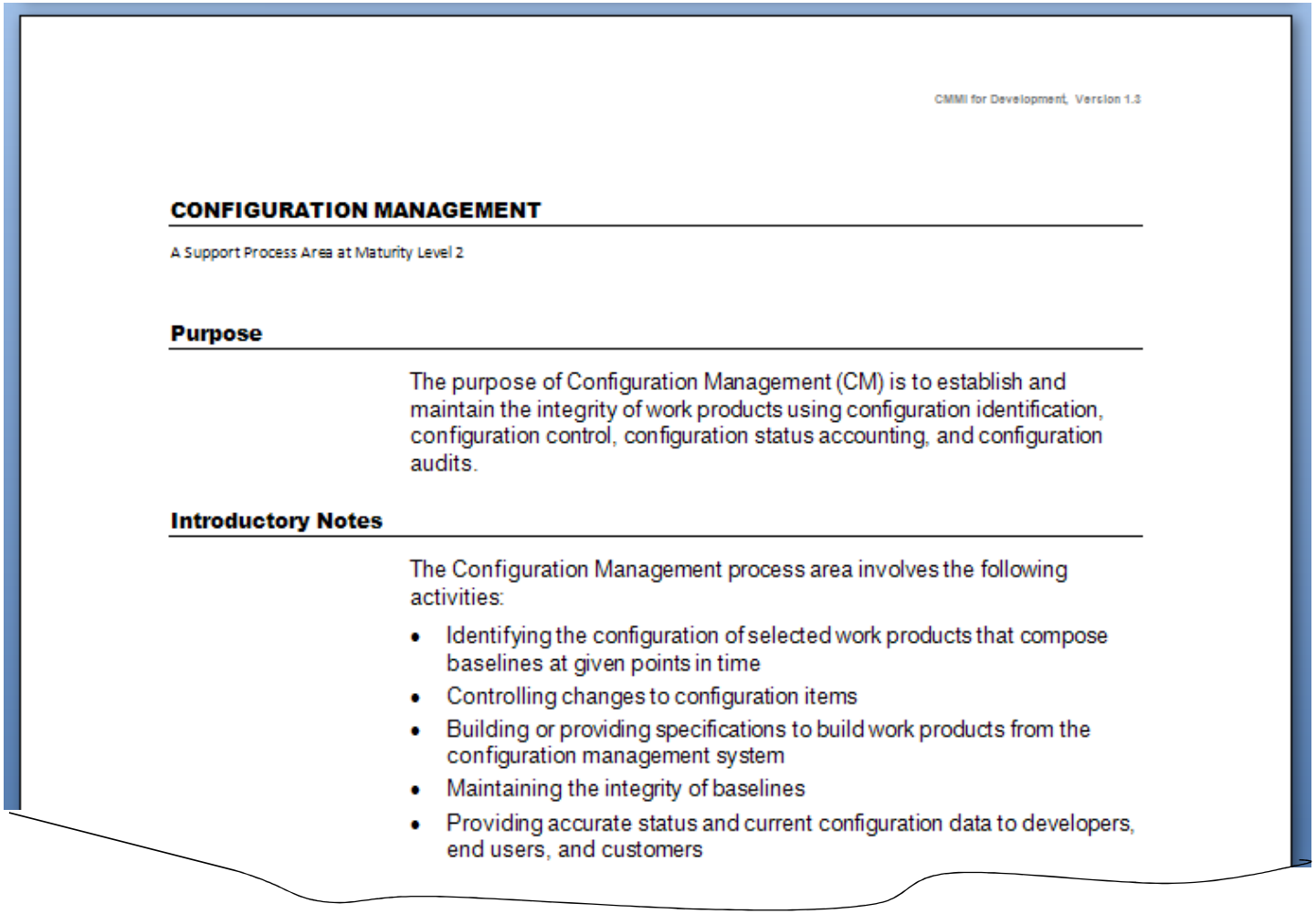
- generally, these representations have different philosophies about the implementation sequence of the process areas.

The “I” in CMMI refers to integration of disparate source models and disciplines.

Process Area Components



Example Page from a Model



CONFIGURATION MANAGEMENT

A Support Process Area at Maturity Level 2

Purpose

The purpose of Configuration Management (CM) is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.

Introductory Notes

The Configuration Management process area involves the following activities:

- Identifying the configuration of selected work products that compose baselines at given points in time
- Controlling changes to configuration items
- Building or providing specifications to build work products from the configuration management system
- Maintaining the integrity of baselines
- Providing accurate status and current configuration data to developers, end users, and customers

Summary of Generic Goals and Practices

Generic Goals	*Generic Practices*
GG1: Achieve Specific Goals	GP 1.1: Perform Specific Practices
GG2: Institutionalize a Managed Process	GP 2.1: Establish an Organizational Policy GP 2.2: Plan the Process GP 2.3: Provide Resources GP 2.4: Assign Responsibility GP 2.5: Train People GP 2.6: Control Work Products GP 2.7: Identify and Involve Relevant Stakeholders GP 2.8: Monitor and Control the Process GP 2.9: Objectively Evaluate Adherence GP 2.10: Review Status with Higher Level Management
GG3: Institutionalize a Defined Process	GP 3.1: Establish a Defined Process GP 3.2: Collect Process Related Experiences

Adapted from
Cepeda Systems &
Software Analysis, Inc.

CMMI Coverage of Modern Engineering Approaches

Much of the engineering content of CMMI-DEV V1.2 is ten years old.

As DEV was a starting point for the other two constellations, no V1.2 model adequately addressed modern engineering approaches.

- For example, both Requirements Development Specific Goal 3 and Specific Practice 3.2 emphasized functionality and not non-functional requirements.

Also, Engineering and other Process Areas rarely mentioned these concepts:

- Quality attributes
- Allocation of product capabilities to release increments
- Product lines
- Technology maturation (and obsolescence)
- Agile methods

Presentation Outline

CMMI V1.3 – Overview and Context for Modern Engineering Practices Changes

Essential Architecture Practices

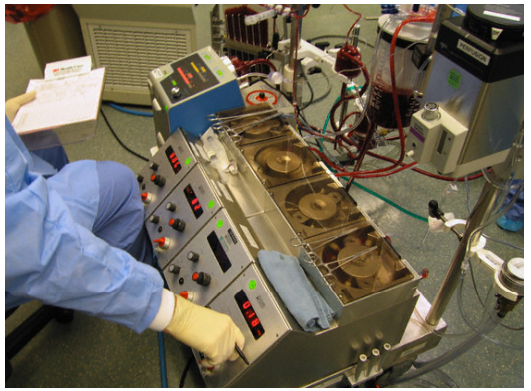
Where Are the Architecture-Centric Practices in CMMI V1.3?

Conclusion

What is Architecture-Centric Engineering?

Architecture-Centric Engineering (ACE) is the discipline of using architecture as the focal point for performing ongoing analyses to gain increasing levels of confidence that systems will support their missions.

Architecture is of enduring importance *because it is the right abstraction for performing ongoing analyses throughout a system's lifetime.*



The **SEI ACE Initiative** develops principles, methods, foundations, techniques, tools, and materials in support of creating, fostering, and stimulating widespread transition of the ACE discipline.

Formal Definition of Software Architecture

"*The **software architecture** of a computing system is the set of **structures** needed to reason about the system, which comprise **software components, relations** among them and **properties** of both.*"

Clements et al, *Documenting Software Architectures, Second Edition.* Addison-Wesley, 2011

System Development

Functional Requirements

If function were all that mattered, any monolithic implementation would do, ..***but other things matter***…

The important quality attributes and their characterizations are key.

analysis, design, development, evolution

- Modifiability
- Interoperability
- Availability
- Security
- Predictability
- Portability

Quality Attribute Drivers → **Software & System Architectures** → **Software & System**

has these qualities

The Non-functional Requirements

Specifying Quality Attributes

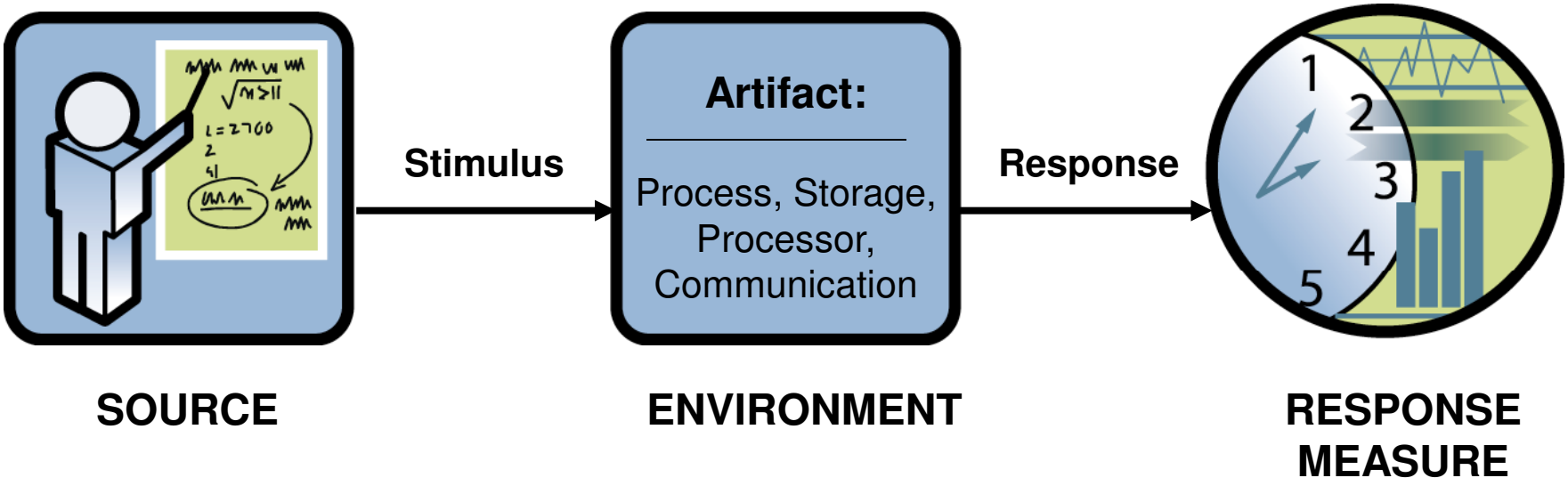
Quality attributes are rarely captured *effectively* in requirements specifications; they are often vaguely understood and weakly articulated.

Just citing the desired qualities is not enough; it is meaningless to say that the system shall be “modifiable” or “interoperable” or “secure” without details about the context.

The practice of specifying quality attribute scenarios can remove this imprecision and allows desired qualities to be evaluated meaningfully.

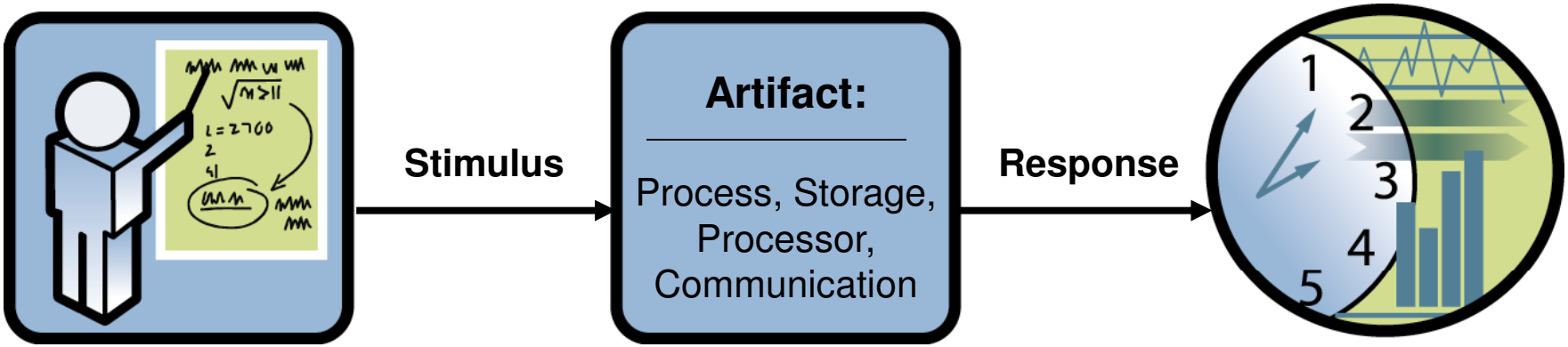
A quality attribute scenario is a short description of an interaction between a stakeholder and a system and the response from the system.

Parts of a Quality Attribute Scenario



Example Quality Attribute Scenario

A "performance" scenario: A remote user requests a data base report under peak load and receives it in under 5 seconds.



SOURCE

Remote user

ENVIRONMENT

Database under peak load

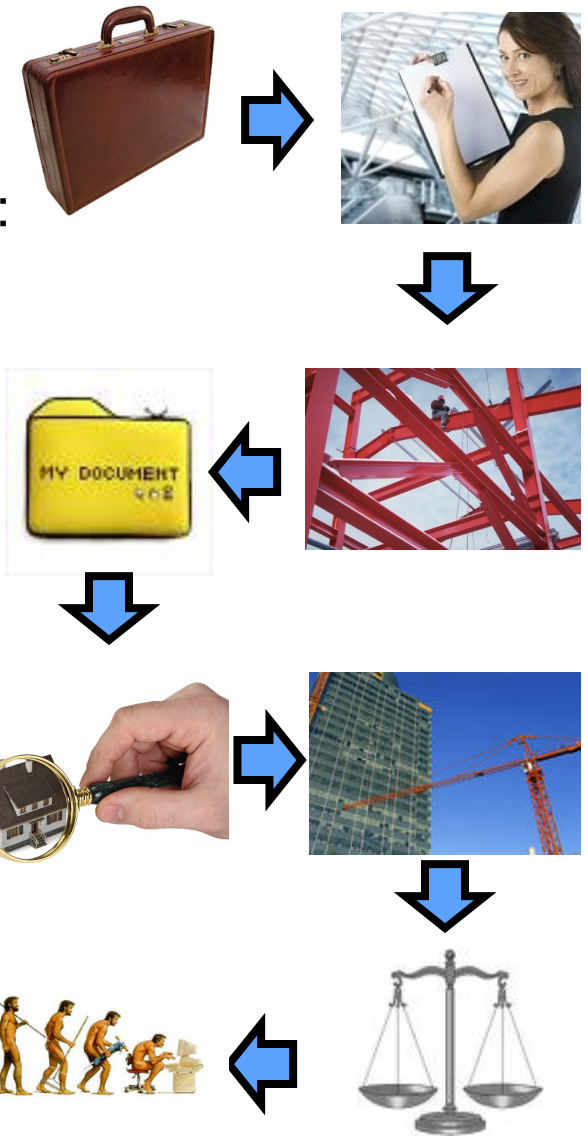
RESPONSE MEASURE

under 5 seconds

Architecture-Centric Activities

Architecture-centric activities include the following:

- creating the **business case** for the system
- understanding the **requirements**
- **creating and/or selecting** the architecture
- **documenting and communicating** the architecture
- **analyzing or evaluating** the architecture
- **implementing** the system based on the architecture
- ensuring that the implementation **conforms** to the architecture
- **evolving** the architecture so that it **continues to meet business and mission goals**



Some SEI Techniques, Methods, and Tools

creating the **business case** for the system	
understanding the **requirements**	*Quality Attribute Workshop (QAW) ** *Mission Thread Workshop (MTW) **
creating and/or selecting the architecture	*Attribute-Driven Design (ADD) and ArchE*
documenting and communicating the architecture	*Views and Beyond Approach; AADL*
analyzing or evaluating the architecture	*Architecture Tradeoff Analysis Method (ATAM) *; SoS Arch Eval *; Cost Benefit Analysis Method (CBAM); AADL*
implementing the system based on the architecture	
ensuring that the implementation **conforms** to the architecture	*ARMIN*
evolving the architecture so that it **continues to meet business and mission goals**	*Architecture Improvement Workshop (AIW)* and ArchE*
ensuring **use of effective architecture practices**	*Architecture Competence Assessment*

*** = indicates a software engineering method that has been extended to systems engineering**

Presentation Outline

CMMI V1.3 – Overview and Context for Modern Engineering Practices Changes

Essential Architecture Practices

Where Are the Architecture-Centric Practices in CMMI V1.3?

Conclusion

Modern Engineering Practices in CMMI

For Version 1.3, CMMI provides better guidance in support of architecture-centric practices (where the practice is addressed in CMMI V1.3 is shown in parentheses).

- creating the **business case** for the system (partially in RD)
- understanding the **requirements** (RD)
- **creating and/or selecting** the architecture (TS)
- **documenting and communicating** the architecture (RD, TS)
- **analyzing or evaluating** the architecture (RD, TS, VAL, VER)
- **implementing** the system based on the architecture (TS; A/PL notes)
- ensuring that the implementation **conforms** to the architecture (VER)
- **evolving** the architecture so that it **continues to meet business and mission goals** (implicit in the changes made for V1.3 to the term “establish and maintain”)

RD = Requirements Development

TS = Technical Solution

VER = Verification

VAL = Validation

Requirements Development

SG 1: Develop Customer Requirements
- SP 1.1 Elicit Needs
- SP 1.2 Transform Stakeholder Needs into [Prioritized] Customer Requirements

SG 2: Develop Product Requirements
- SP 2.1 Establish Product and Product Component Requirements
- SP 2.2 Allocate Product Component Requirements
- SP 2.3 Identify Interface Requirements

SG 3: Analyze and Validate Requirements
- SP 3.1 Establish Operational Concepts and Scenarios
- SP 3.2 Establish a Definition of Required Functionality and Quality Attributes
- SP 3.3 Analyze Requirements
- SP 3.4 Analyze Requirements to Achieve Balance
- SP 3.5 Validate Requirements

In SP1.2, added that customer requirements should be **prioritized** based on their criticality to the **customer** and other stakeholders "representing all phases of the product's lifecycle … including *business* as well as technical functions."

In SP 2.1, added a focus on **architectural requirements** and quality attribute **measures**.

In SP 2.2, added a subpractice allocating requirements to **delivery increments**.

Addressed "**Quality attributes**" (QAs) as well as functionality in SG3 and SP 3.2 statements.

In SP 3.1, broadened emphasis to **"operational, sustainment,** and **development"** scenarios.

In SP 3.2, determined **architecturally-significant QAs** from mission and business drivers.

Technical Solution

SG 1: Select Product Component Solutions

- SP 1.1 Develop Alternative Solutions and Selection Criteria
- SP 1.2 Select Product Component Solutions

SG 2: Develop the Design

- SP 2.1 Design the Product or Product Component
- SP 2.2 Establish a Technical Data Package
- SP 2.3 Design Interfaces Using Criteria
- SP 2.4 Perform Make, Buy, or Reuse Analyses

SG 3: Implement the Product Design

- SP 3.1 Implement the Design
- SP 3.2 Develop Product Support Documentation

Intro Notes: "QA **models, simulations, prototypes or pilots** can be used to provide additional information about the properties of the potential design solutions to aid in the selection of solutions. Simulations can be particularly useful for projects developing systems-of-systems."

In SP 1.1, Added an example selection criterion, "Achievement of key quality attribute requirements" and a new subpractice: "Identify **re-usable solution** components or applicable **architecture patterns.".**

In SP 2.1, described architecture definition tasks such as selecting **architectural patterns** and formally defining component behavior and interactions using an **architecture description language.**

In SP 2.2, added subpractice to determine **views** to document structures and address stakeholder concerns.

In SP 2.3, mentioned exception and error handling,

Product Integration

SG 1: Prepare for Product Integration
- SP 1.1 Establish an Integration Strategy
- SP 1.2 Establish the Product Integration Environment
- SP 1.3 Establish Product Integration Procedures and Criteria

SG 2: Ensure Interface Compatibility
- SP 2.1 Review Interface Descriptions for Completeness
- SP 2.2 Manage Interfaces

SG 3: Assemble Product Components and Deliver the Product
- SP 3.1 Confirm Readiness of Product Components for Integration
- SP 3.2 Assemble Product Components
- SP 3.3 Evaluate Assembled Product Components
- SP 3.4 Package and Deliver the Product or Product Component

Revised the purpose to ensure proper **behavior** instead of proper function, thereby more implicitly including **quality attributes** as well as functionality.

Changed emphasis from integration sequence to an emphasis on **integration strategy**, i.e., the approach to receiving, assembling, and evaluating product components. The **architecture** will significantly influence the selection of a product integration strategy.

In the PA notes, addressed: **interfaces** to data sources and middleware; APIs, **automated builds, continuous integration**

Validation

SG 1: Prepare for Validation

- SP 1.1 Select Products for Validation
- SP 1.2 Establish the Validation Environment
- SP 1.3 Establish Validation Procedures and Criteria

SG 2: Validate Product or Product Components

- SP 2.1 Perform Validation
- SP 2.2 Analyze Validation Results

Reinforced when validation occurs in the product lifecycle: "validation is performed early (concept/exploration phases) and incrementally throughout the product lifecycle (including transition to operations and sustainment)."

In VAL SP 1.1, included **access protocols** and data interchange reporting formats as examples of what to validate.

Also, included **incremental delivery of working and potentially acceptable product** as an example validation method.

Verification

SG 1: Prepare for Verification
- SP 1.1 Select Work Products for Verification
- SP 1.2 Establish the Verification Environment
- SP 1.3 Establish Verification Procedures and Criteria

SG 2: Perform Peer Reviews
- SP 2.1 Prepare for Peer Reviews
- SP 2.2 Conduct Peer Reviews
- SP 2.3 Analyze Peer Review Data

SG 3: Verify Selected Work Products
- SP 3.1 Perform Verification
- SP 3.2 Analyze Verification Results

In SP 1.1, added example verification methods**: software architecture evaluation and implementation conformance evaluation** and continuous integration.

In SP 1.3, added example sources of verification criteria: customers reviewing work products collaboratively with developers.

In SP 2.1, added example type of peer review: **architecture implementation conformance evaluation**

In SP 2.3, added examples of peer review data that can be analyzed: **user stories** or case studies associated with a defect and the end-users and customers who are associated with defects

Changes in CMMI Terminology - 1

Allocated requirement

DEFINITION

Requirement that ~~levies~~results from levying all or part of ~~the performance and functionality of~~ a higher level requirement on a lower level architectural element or design component.

More generally, requirements can be allocated to other logical or physical components including people, consumables, delivery increments, or the architecture as a whole, depending on what best enables the product or service to achieve the requirements.

The improvements to the definition make the substance of the solution space and allocation of requirements to it more explicit, allowing for superior architectures and more insightful analyses (including verification) of requirements and technical solutions.

Changes in CMMI Terminology - 2

Architecture

DEFINITION

The set of structures needed to reason about a product. These structures are comprised of elements, relations among them, and properties of both.

In a service context, the architecture is often applied to the service system.

Note that functionality is only one aspect of the product. Quality attributes, such as responsiveness, reliability, and security, are also important to reason about. Structures provide the means for highlighting different portions of the architecture. (See also “functional architecture.”)

This term and its use throughout the rest of the model is intended to encourage use of proven, architecture-centric practices and the recognition of “architecture” as a principal engineering artifact.

Changes in CMMI Terminology - 3

Definition of required functionality and quality attributes

DEFINITION

A characterization of required functionality and quality attributes obtained through "chunking," organizing, annotating, structuring, or formalizing the requirements (functional and non-functional) to facilitate further refinement and reasoning about the requirements as well as (possibly, initial) solution exploration, definition, and evaluation.

As technical solution processes progress, this characterization can be further evolved into a description of the architecture versus simply helping scope and guide its development, depending on the engineering processes used; requirements specification and architectural languages used; and the tools and the environment used *[snip].*

The term "definition of required functionality" that appeared in V1.2 has been removed from CMMI because of the implicit suggestion that functionality be addressed first or has higher priority. The term has been replaced with the one above, which is intended to help ensure a sufficiently balanced focus (functional *and* non-functional) in requirements analysis.

Changes in CMMI Terminology - 4

"Functional analysis" and "functional architecture"

These terms, which appeared in V1.2, are now "cul de sacs" in the model.

The only place these terms now appear in CMMI-DEV V1.3 outside the Glossary is in the first note of RD SP 3.2 and as an example work product.

The note contrasts the approaches implied by these terms with "modern engineering approaches" that encourage a more balanced treatment of requirements, both functional and non-functional.

Changes in CMMI Terminology - 5

Product line

DEFINITION

A group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission ~~.~~ and that are developed from a common set of core assets in a prescribed way.

The development or acquisition of products for the product line is based on exploiting commonality and bounding variation (i.e., restricting unnecessary product variation) across the group of products. The managed set of core assets (e.g., requirements, architectures, components, tools, testing artifacts, operating procedures, software) includes prescriptive guidance for their use in product development. Product line operations involve interlocking execution of the broad activities of core asset development, product development, and management.

Many people use “product line” just to mean the set of products produced by a particular business unit, whether they are built with shared assets or not. We call that collection a "portfolio," and reserve "product line" to have the technical meaning given here.

Changes in CMMI Terminology - 6

Quality attribute

DEFINITION

A property of a product or service by which its quality will be judged by relevant stakeholders. Quality attributes are characterizable by some appropriate measure.

Quality attributes are non-functional, such as timeliness, throughput, responsiveness, security, modifiability, reliability, and usability. They have a significant influence on the architecture.

This term is now included in the Glossary for the first time. This term is intended to supplant others – especially those focusing on only a few dimensions (e.g., "performance") – to encourage a broader view of non-functional requirements. The term was refined through much effort, as neither ISO 25030 (SQuaRE) nor the original SEI definitions were quite satisfactory. In addition, uses of the term "performance" throughout the model were reviewed for clarity, and where appropriate, revised or qualified.

Changes in CMMI Terminology - 7

Establish and maintain

DEFINITION

Create, document, use, and revise . . . as necessary to ensure ~~it remains~~ they remain useful.

The phrase "establish and maintain" ~~means more than a combination of its component terms;~~ . . . plays a special role in communicating a deeper principle in CMMI: work products that have a central or key role in work group, project, and organizational performance should be given attention to ensure they are used and useful in that role.

This phrase has particular significance in CMMI because it often appears in goal and practice statements . . . and should be taken as shorthand for applying the principle to whatever work product is the object of the phrase.

The above term appears in many CMMI practices. This term was changed in V1.3 to support the evolution of key artifacts so that they remain useful. Example from RD SP 2.1 note: "The modification of requirements due to approved requirement changes is covered by the "maintain" aspect of this specific practice…" Likewise for architecture (TS SP 2.2).

V1.3 Includes Notes on How to Address Agile Methods and Product Lines

Other Informative Material Changes

Special notes for Agile and for Product Lines have been inserted in the Intro Notes of various PAs in V1.3.

Changes Supporting Use of Agile Methods

Because CMMI practices are written for use in a broad variety of contexts, business situations, and application domains, it is not possible (even if it were appropriate) to advocate any specific implementation approach.

However, Agile methods and approaches are now in wider use, and so for V1.3, it seemed appropriate to identify how Agile approaches can address CMMI practices and conversely, identify the value that CMMI can bring to Agile implementations. And likewise for Product Lines.

Addressing Agile – Example PA Notes

A note added in the RD Intro Notes:

> In Agile environments, requirements are communicated and tracked through mechanisms such as product backlogs, story cards, and screen mock-ups. *[snip]* Traceability and consistency across requirements and work products is addressed through the mechanisms already mentioned as well as during start-of-iteration or end-of-iteration activities such as "retrospectives" and "demo days."

A note added in the TS Intro Notes:

> In Agile environments, the focus is on early solution exploration. By making the selection and tradeoff decisions more explicit, the Technical Solution process area helps improve the quality of those decisions, both individually and over time. *[snip]* When someone other than the team will be working on the product in the future, release information, maintenance logs, and other data are typically included with the installed product. To support future product updates, rationale (for trade-offs, interfaces, and purchased parts) is captured so that why the product exists can be better understood. *[snip]*

Addressing Product Lines – Example Notes

An example of a note added in the RD Intro Notes:

> For product lines, engineering processes (including requirements development) may be applied to at least two levels in the organization. At an organizational or product line level, a "commonality and variation analysis" is performed to help elicit, analyze, and establish core assets for use by projects within the product line. At the project level, these core assets are then used as per the product line production plan as part of the project's engineering activities.

An example of a note added in the TS Intro Notes:

> For product lines, these practices apply to both core asset development (i.e., building for reuse) and product development (i.e., building with reuse). Core asset development additionally requires product line variation management (the selection and implementation of product line variation mechanisms) and product line production planning (the development of processes and other work products that define how products will be built to make best use of these core assets).

Presentation Outline

CMMI V1.3 – Overview and Context for Modern Engineering Practices Changes

Essential Architecture Practices

Where Are the Architecture-Centric Practices in CMMI V1.3?

Conclusion

Summary & Conclusions

The quality and longevity of a software-intensive system is largely determined by its architecture.

Early identification of architectural risks saves money and time.

There are proven practices to help ensure that suppliers and acquirers can develop and acquire systems that have appropriate architectures.

CMMI V1.3 has a new emphasis on architecture.

The efficacy of the architecture has a direct impact on program or mission success, and customer satisfaction.

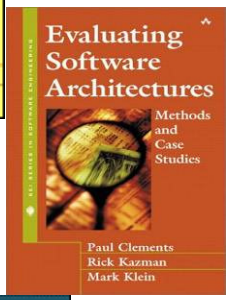
References - 1

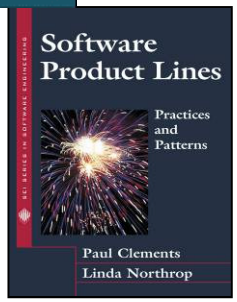
Software Architecture in Practice, Second Edition
Bass, L.; Clements, P.; & Kazman, R. Reading, MA: Addison-Wesley, 2003.

Evaluating Software Architectures: Methods and Case Studies
Clements, P.; Kazman, R.; & Klein, M. Reading, MA: Addison- Wesley, 2002.

Documenting Software Architectures: Views and Beyond, Second Edition
Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. Reading, MA: Addison-Wesley, 2011.

Software Product Lines: Practices and Patterns
Clements, P.; Northrop, L. Reading, MA: Addison-Wesley, 2001.

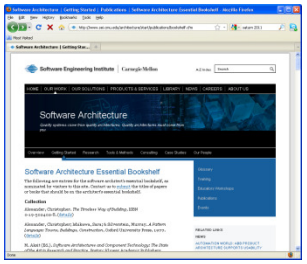




References - 2

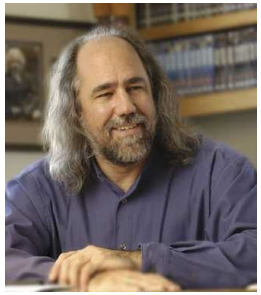
You can find a moderated list of references on the “Software Architecture Essential Bookshelf”

http://www.sei.cmu.edu/architecture/start/publications/bookshelf.cfm



Grady Booch: Handbook of Software Architecture (currently only an on-line reference):

http://www.handbookofsoftwarearchitecture.com/index.jsp?page=Main

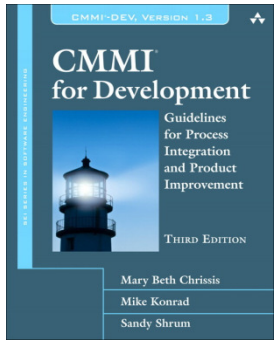


CMMI for Development, Version 1.3

http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm

(also available as a book from the SEI Series on Software Engineering)

Chrissis, Mary Beth; Konrad, Mike; & Shrum, Sandy. CMMI: Guidelines for Process Integration and Product Improvement, 3rd Edition. Boston: Addison-Wesley, 2011.



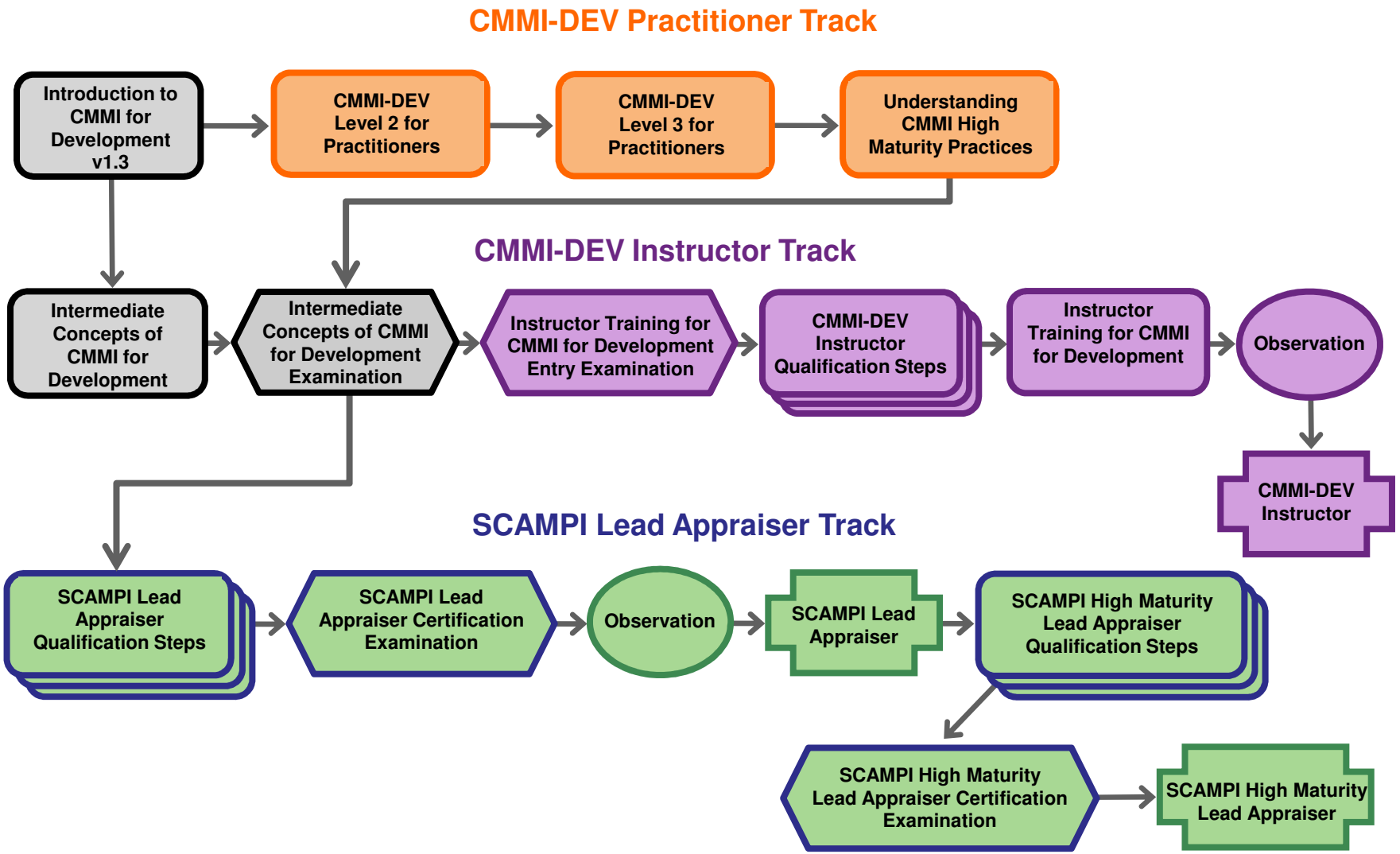
The SEI Software Architecture Curriculum

Six Courses	*Three Certificate Programs*		
	Software Architecture Professional	**ATAM Evaluator**	**ATAM Leader**
Software Architecture Principles and Practices*	✓	✓	✓
Documenting Software Architectures	✓		✓
Software Architecture Design and Analysis	✓		✓
Software Product Lines	✓		✓
ATAM Evaluator Training		✓	✓
ATAM Leader Training			✓
ATAM Observation			✓

✓ : required to receive certificate

*: available through e-learning

CMMI Roadmap for Professionals



Contact Information

Larry Jones

Research, Technology, and Systems Solutions Program

Telephone: 719-481-8672

Email: lgj@sei.cmu.edu

Mike Konrad

SEPM

Telephone: 412-268-5813

Email: mdk@sei.cmu.edu

U.S. Mail:

Software Engineering Institute

Carnegie Mellon University

4500 Fifth Avenue

Pittsburgh, PA 15213-3890

World Wide Web:

http://www.sei.cmu.edu/productlines

SEI Fax: 412-268-5758

Questions

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.