

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

Dr. Lawrence G. Jones  
Dr. Michael Konrad

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





SEPG<sup>SM</sup> 2011  
NORTH AMERICA

*Perform at a Higher Level*



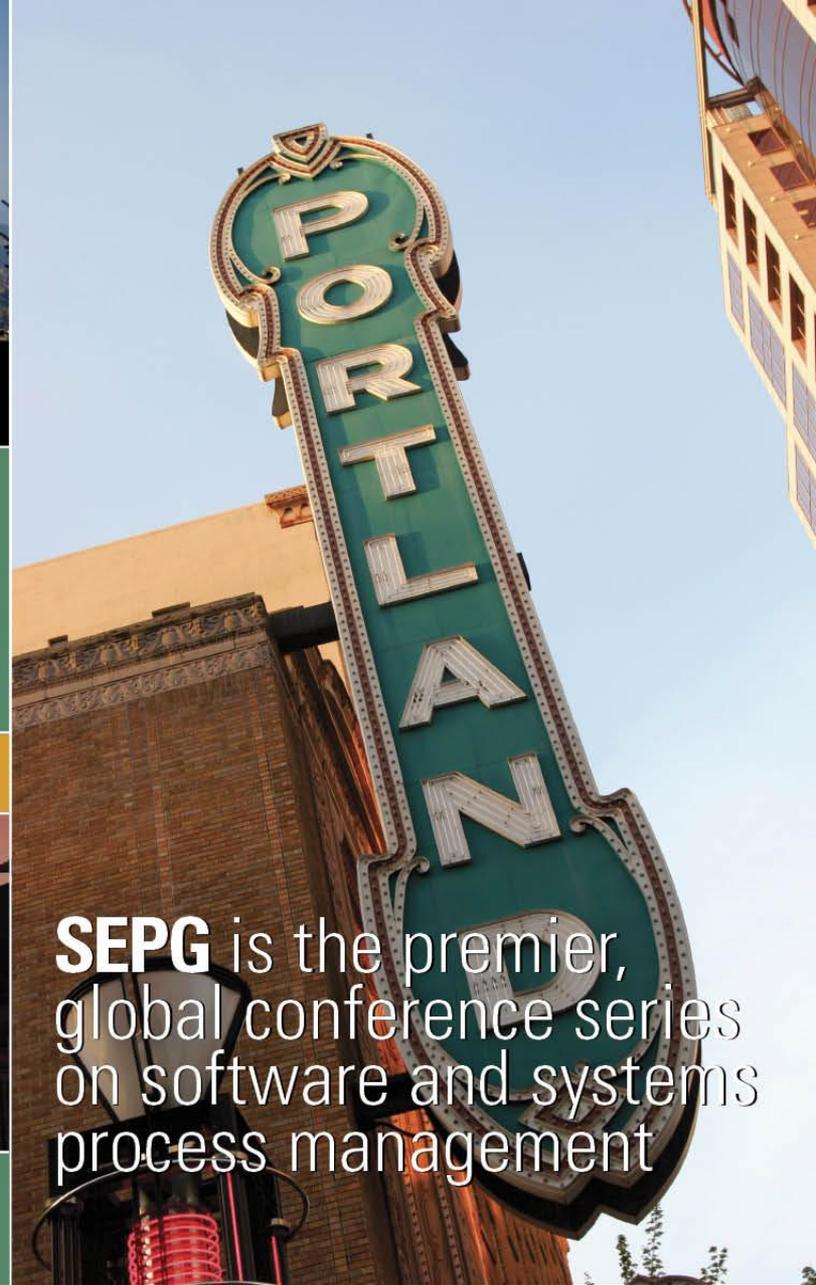
# The Power of Process

Coming to Portland in March 2011

SEPG North America 2011 | Oregon Convention Center | Portland, OR



[www.sei.cmu.edu/sepg/na/2011](http://www.sei.cmu.edu/sepg/na/2011)



**SEPG** is the premier, global conference series on software and systems process management



Software Engineering Institute

Carnegie Mellon

CMMI V1.3 and Architecture-Centric Engineering  
© 2011 Carnegie Mellon University

# SATURN 2011

Seventh Annual SEI Architecture  
Technology User Network Conference

# Architecting the Future



May 16-20, 2011 | San Mateo County, California



The SEI Architecture Technology User Network (SATURN) Conference brings together experts to exchange best architecture-centric practices in developing, acquiring, and maintaining software-reliant systems.

## [www.sei.cmu.edu/saturn/2011](http://www.sei.cmu.edu/saturn/2011)

### 7 Things You Need to Know About the Next 7 Years in Architecture.



Architecture is Not Just for Architects



Architecture, Agile Development, and Business Agility



Soft Skills for Architects



Service-Oriented Architecture (SOA) and Cloud Computing



Architectural Knowledge Management



Architecting to Meet Tomorrow's Global Challenges



Model-Driven Architecting



Software Engineering Institute | Carnegie Mellon

in collaboration with **Software**

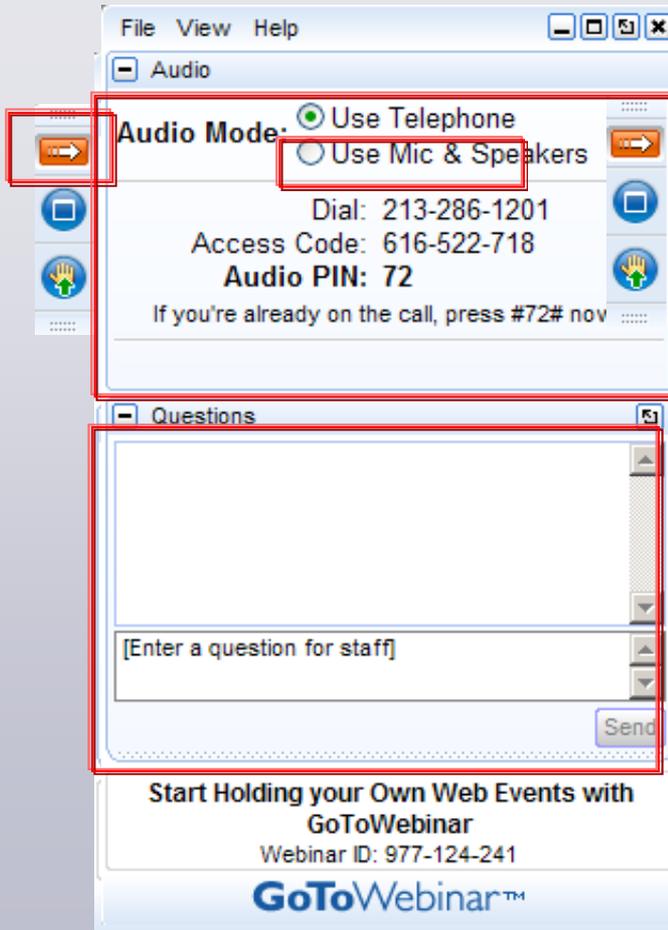


Software Engineering Institute

Carnegie Mellon

CMMI V1.3 and Architecture-Centric  
Engineering  
© 2011 Carnegie Mellon University

# How to Participate Today



Open and close your Panel  
View, Select, and Test your audio

Submit text questions

Q&A addressed at the end of  
today's session



# The Presenters



**Larry**



**Mike**



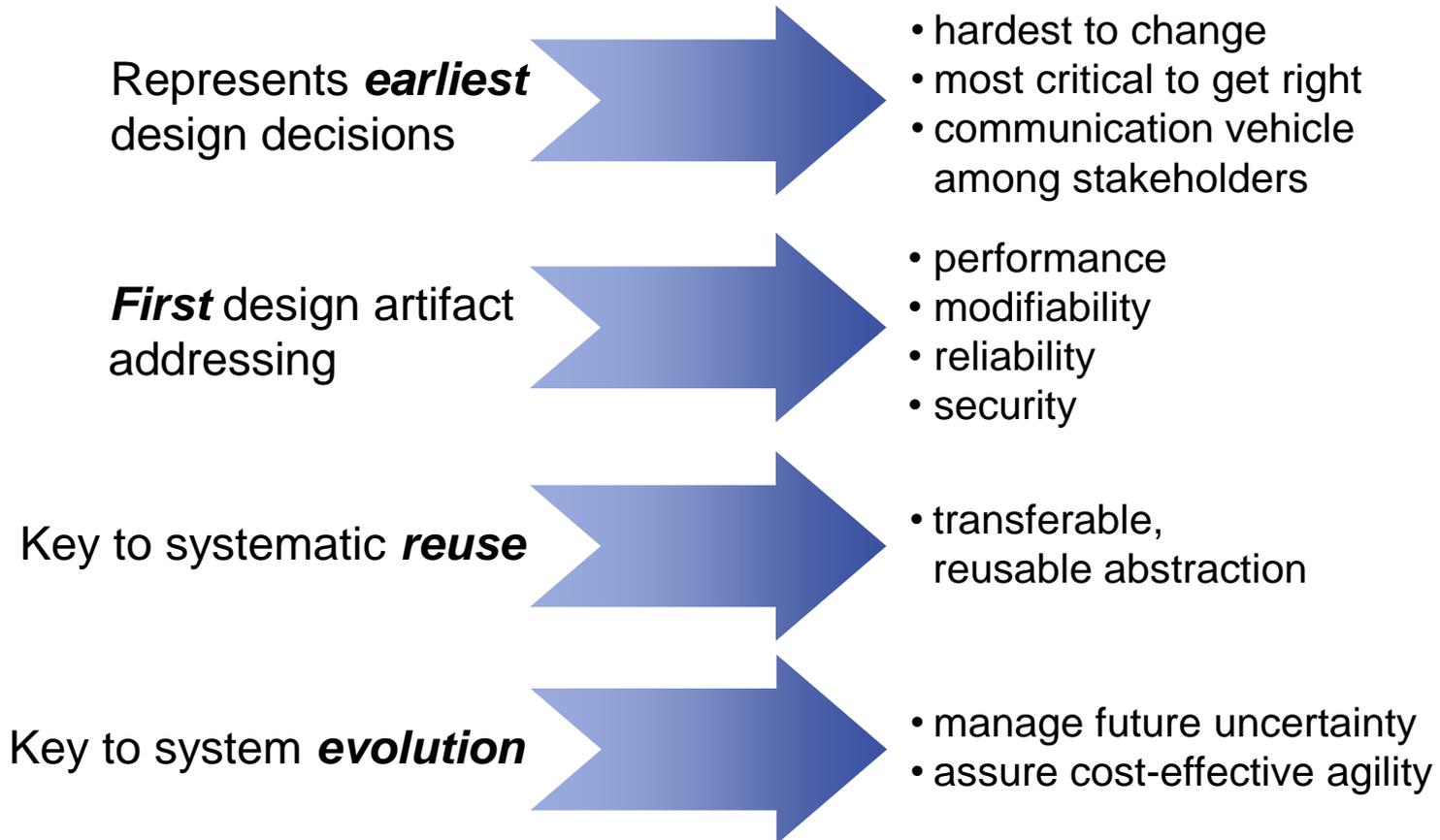
# Architecture is Important

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

In recent studies by OSD, the National Research Council, NASA, and the NDIA, architectural issues are identified as a systemic cause of software problems in DoD systems.



# Why Is Architecture Important?



The **right architecture** paves the way for system **success**.

The **wrong architecture** usually spells some form of **disaster**.



# People are Serious About Architecture

“Software Architect” was identified by CNN Money.com as the #1 “Best Job in America.” (Oct 2010)<sup>1</sup>



The US Army has mandated that all Program Executive Offices appoint a Chief Software Architect. (May 2009)<sup>2</sup>



1. <http://money.cnn.com/magazines/moneymag/bestjobs/2010/snapshots/1.html>
2. Memo by LTG N. Ross Thompson, Mil Dept of ASA (ALT) on May 26, 2009.



# Webinar Learning Outcomes

After completing this webinar, attendees should

- know the meaning of the terms “architecture” and “quality attribute”
- know that architecture is important to the achievement of business, product, or mission goals
- know that quality attributes have a dominant influence on a system’s architecture
- 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



# Conventions & Caveats for the Tutorial



The coverage of architecture-centric practices in CMMI V1.3 is broad, focused on “products” and “solutions” – not just on software.

- But much of the tutorial material came from SEI assets whose focus was software-intensive systems. Please bear this in mind. We believe the principles apply beyond simply software.

Our focus in the tutorial will be on CMMI for Development because that is where the architecture-centric practices are most deeply covered but similar changes were also made to the other two CMMI models.

CMMI uses the term “product” to refer to what is delivered to the customer or end-user. In this tutorial, we will often use the term “system” to refer to the product.

This tutorial cannot completely convey everything you might like to learn about architecture-centric engineering.

- References are provided at the end for you to learn more.



# Presentation Outline

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

Architecture and its Importance

Essential Architecture Practices

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

Conclusion



# CMMI in a Nutshell

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

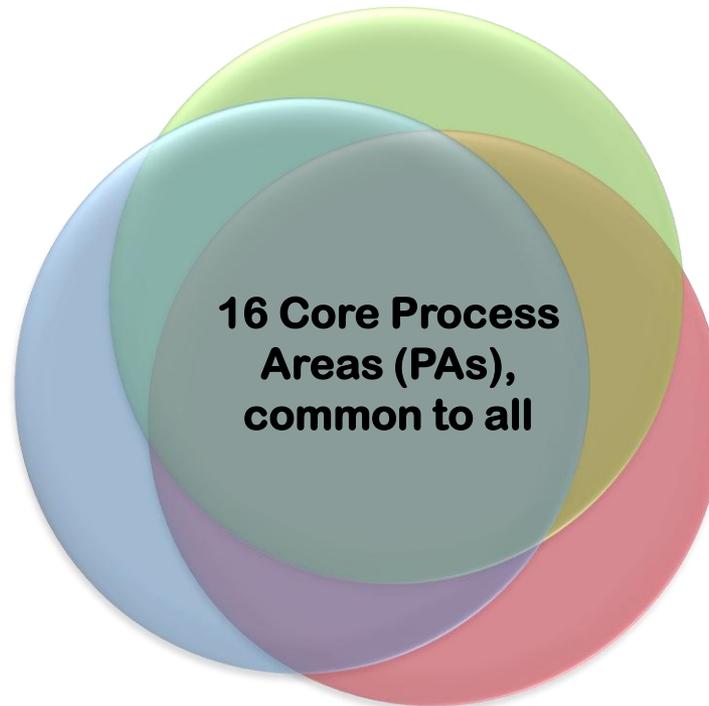
CMMI organizes these practices into structures that help an organization

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

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

**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)

## 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)

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



# 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 RD SG 3 and RD SP 3.2 emphasized functionality and not non-functional requirements.

Also, Engineering and other PAs 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

## Architecture and its Importance

Essential Architecture Practices

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

Conclusion



# Architecture is About Structure and Decisions

Structures result from decisions

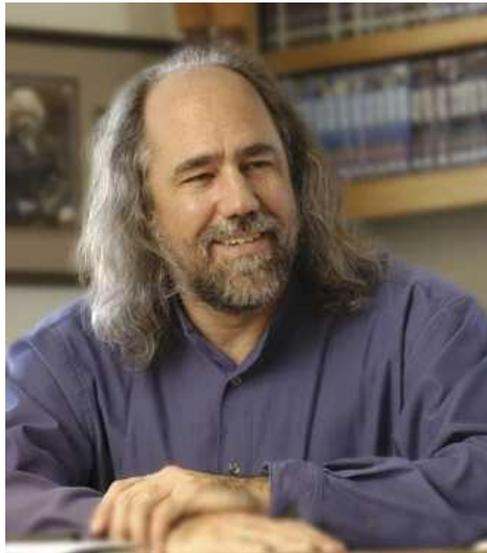
- Business / mission goals provide a reasoned basis for decisions.
- Each decision is a tradeoff that enables something and precludes other things.
- Tradeoffs are driven by quality attribute requirements.

This is true regardless of the domain  
– commercial or defense.



# “Every system has an architecture...

...encompassing the key abstractions and mechanisms that define that system's structure and behavior... In every case - from idioms to mechanisms to architectures - these patterns are either



intentional

or

accidental”

- Grady Booch in the Preface to *Handbook of Software Architecture*



# Architecture and Strategy

An *Intentional Architecture* is the embodiment of your business strategy

- Intentional Architecture links technology decisions to business goals



An *Accidental Architecture* limits strategy options

- Accidental Architecture becomes your de facto strategy



# Isn't Architecting Expensive?!

Compared to what?

- Over-committing because you don't have a blueprint for the whole system?
- Inefficiency from inability to coordinate work?
- Late rework when defects found in test and integration?
- Delivering late and over budget?
- Developing a failed product that doesn't meet stakeholder's needs?



# What Is an Architecture?

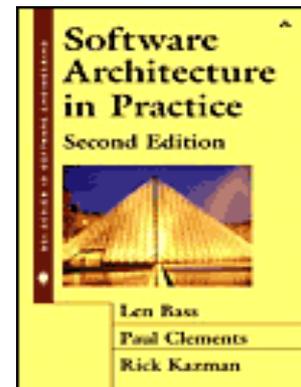
Informally, an architecture is the blueprint describing the software structure of a system.



# Formal Definition of Software Architecture

*“The **software architecture** of a program or computing system is the **structure or structures** of the system, which comprise the **software elements**, the **externally visible properties** of those elements, and the **relationships among them**.”<sup>1</sup>*

<sup>1</sup> Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition*. Boston, MA: Addison-Wesley, 2003.



# Formal Definition of System Architecture

A **system architecture** describes the elements and interactions of a complete system including its hardware elements and its software elements.

**System Architecture:** “*The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors.*”<sup>1</sup>

*Systems Engineering* is a design and management discipline useful in designing and building large, complex, and interdisciplinary systems.<sup>2</sup>

<sup>1</sup> Rechtin, E. *Systems Architecting: Creating and Building Complex Systems*. Englewood Cliffs, NJ : Prentice-Hall, 1991.

<sup>2</sup> International Council On Systems Engineering (INCOSE), Systems Architecture Working Group, 1996.



# Implications

Architecture is an abstraction of a system.

Architecture defines the properties of elements.

Systems can and do have many structures.

Every software-intensive system *has* an architecture.

Just having an architecture is different from having an architecture that is known to everyone.

If you don't develop an architecture, you will get one anyway –  
**and you might not like what you get!**



# Structures and Views - 1

One house, many views



{  
Carpentry view  
Plumbing view  
Electrical view  
Ductwork view

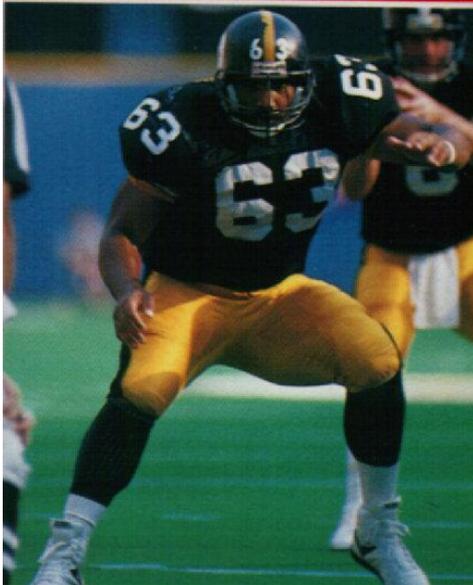
No single view accurately represents the house.

No single view can be used to build the house.

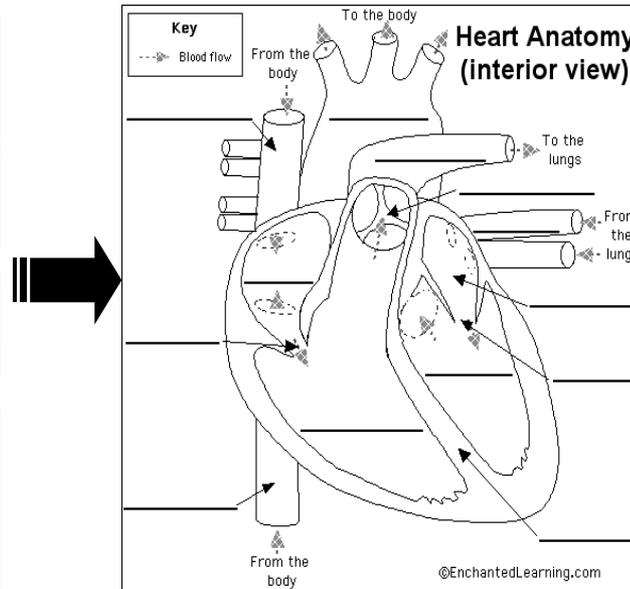
Although these views are pictured differently, and each has different properties, all are related. Together, they describe the architecture of the house.



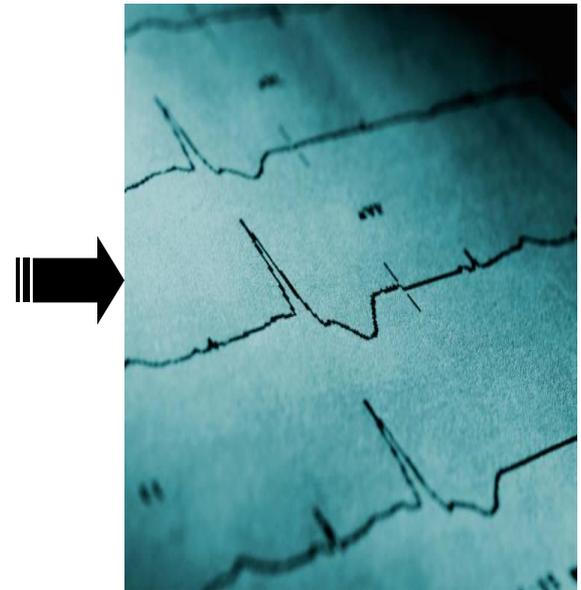
# Structures and Views - 2



A human body comprises multiple *structures*.



a *static view* of one human *structure*



a *dynamic view* of that *structure*

One body has many structures, and those structures have many views. So it is with software.



# Presentation Outline

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

Architecture and its Importance

**Essential Architecture Practices**

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

Conclusion

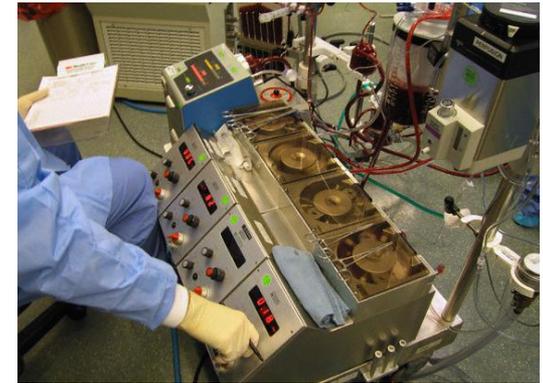


# What is Architecture-Centric Engineering?

**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.



# Principles of ACE

1. Regardless of scale, architecture is the appropriate abstraction for reasoning about business/mission goal satisfaction.
  - provides sufficient detail to reason about mission and business goal satisfaction and constrain implementation
  - provides sufficient abstraction for a relatively small number of architects to conceptually understand the system
2. Quality attributes have a dominant influence on a system's architecture.
  - Quality attribute requirements stem from business and mission goals.
  - Key quality attributes need to be characterized in a system-specific way.
  - Scenarios are a powerful way to characterize quality attributes and represent stakeholder views.
3. Architectural prescriptions must be demonstrably satisfied by the implementation.
  - Software architecture must be central to software development activities.
  - These activities must have an explicit focus on quality attributes.
  - These activities must directly involve stakeholders – not just the architecture team.
  - The architecture must be descriptive and prescriptive.



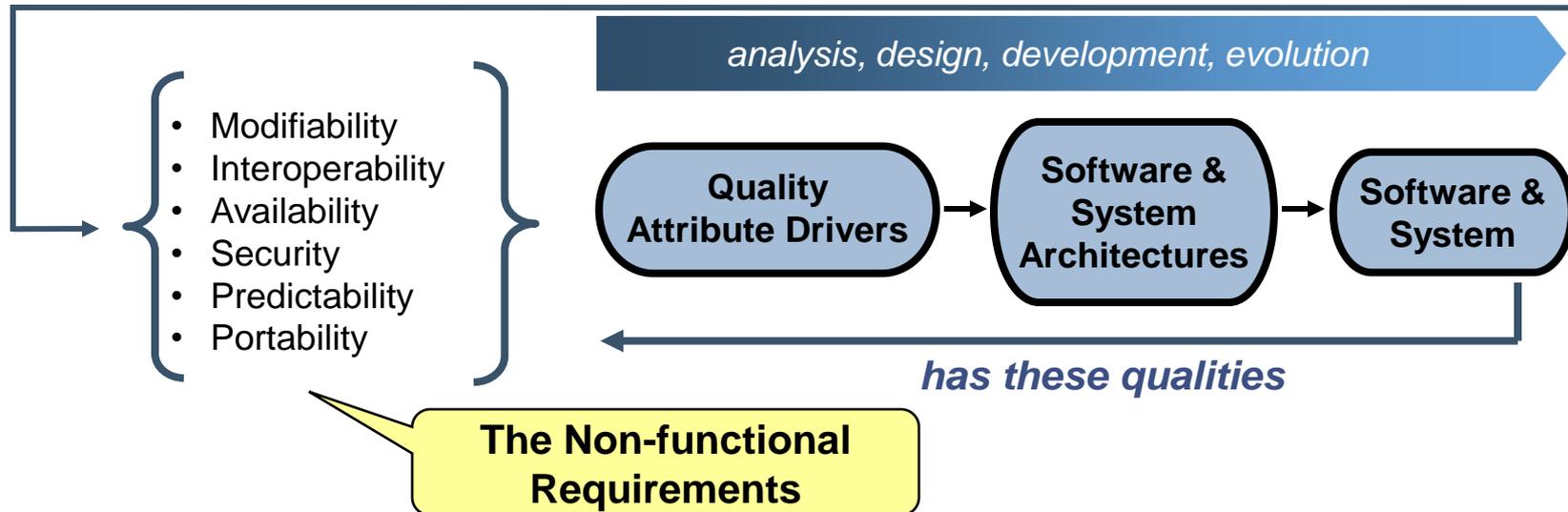
# System Development



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.*



# Specifying Quality Attributes

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

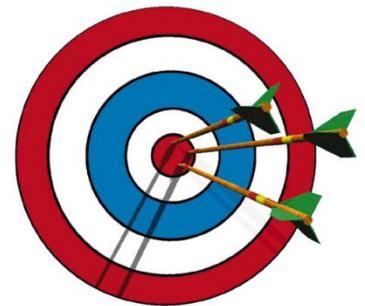
Just citing the desired qualities is not enough; it is meaningless to say that the system shall be “modifiable” or “interoperable” or “secure” without details about the context.

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

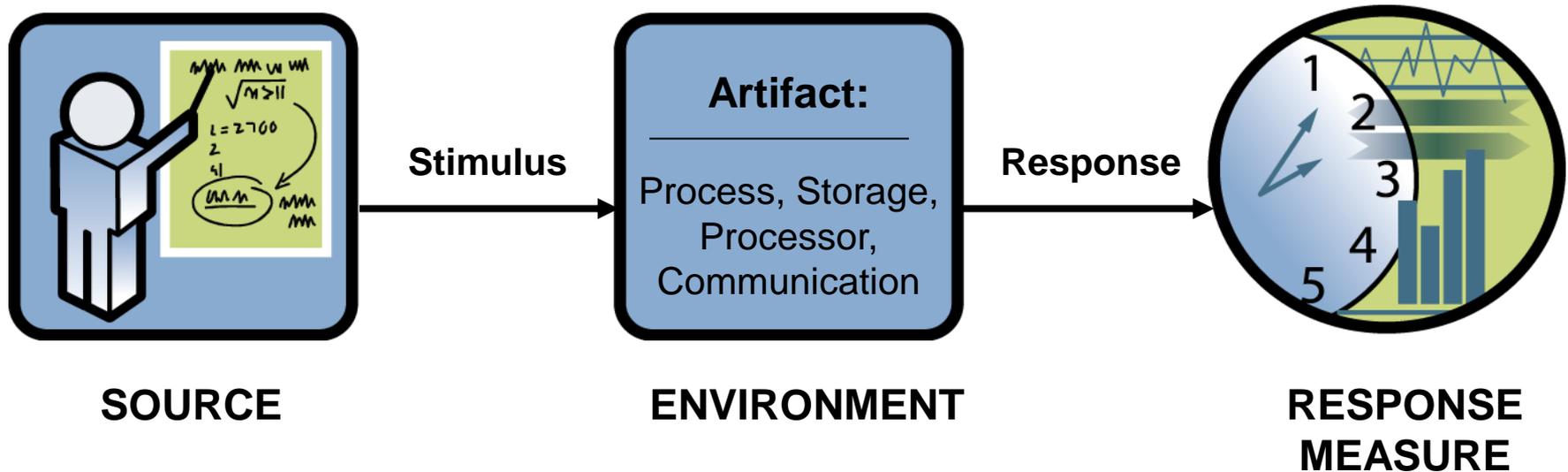
A quality attribute scenario is a short description of an interaction between a stakeholder and a system and the response from the system.



bcp033-04 fotosearch.com

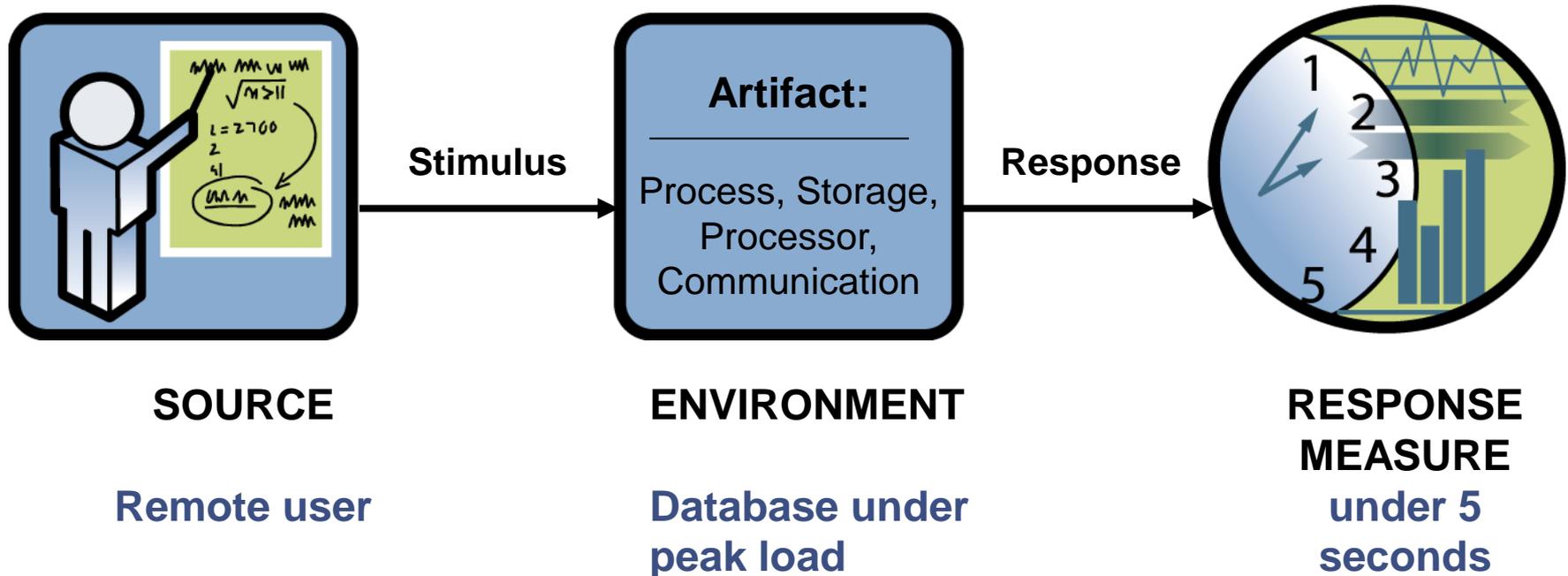


# Parts of a Quality Attribute Scenario



# 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.



# 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

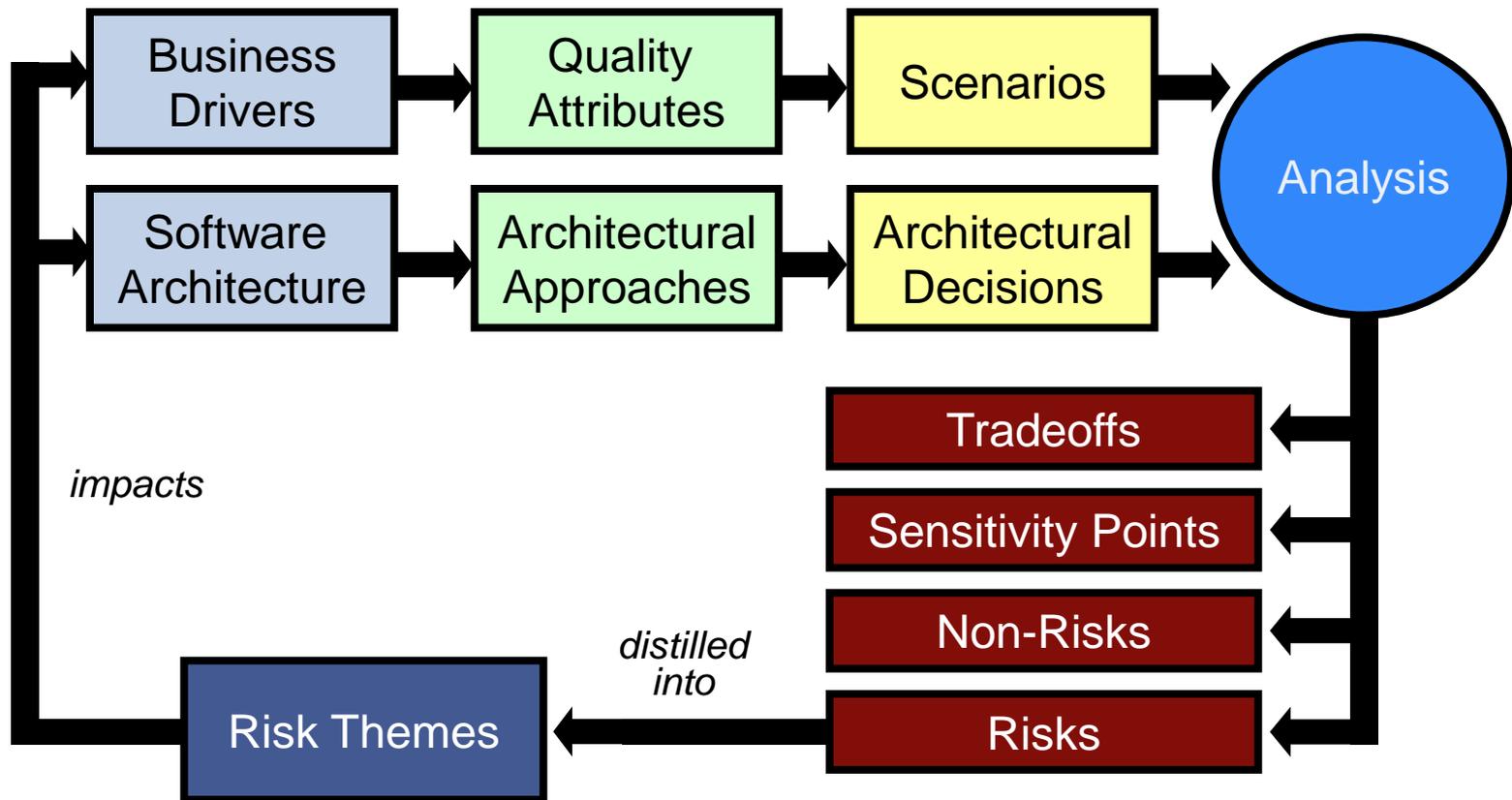
creating the <b>business case</b> for the system	
understanding the <b>requirements</b>	<i>Quality Attribute Workshop (QAW) *</i> <i>Mission Thread Workshop (MTW) *</i>
<b>creating and/or selecting</b> the architecture	<i>Attribute-Driven Design (ADD)</i> <i>and ArchE</i>
<b>documenting and communicating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

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



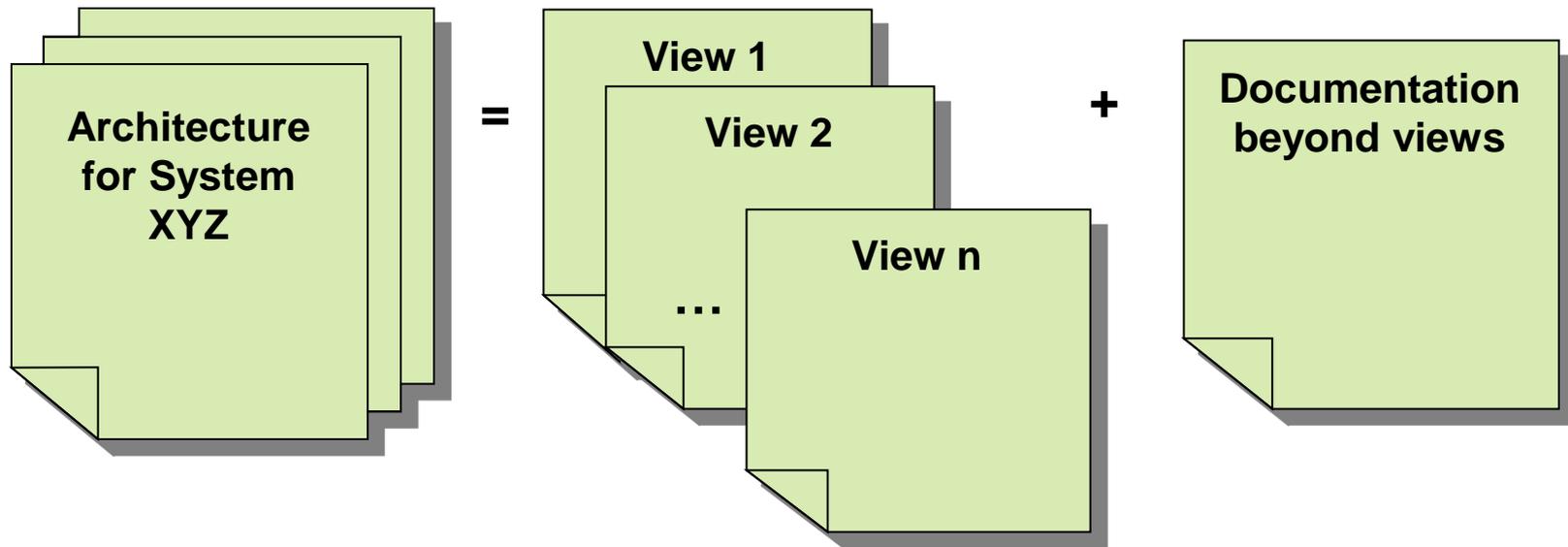
# Analyzing the Architecture – SEI’s Architecture Tradeoff Analysis Method® (ATAM®)

The ATAM is an architecture evaluation method that focuses on multiple quality attributes.



# View-Based Documentation

Views give us our basic principle of architecture documentation



Documenting an architecture is a matter of documenting the relevant views, and then adding documentation that applies to more than one view.

The choice of views used depends on the nature of the system and the stakeholder needs.



# Software Architecture Documentation Needs

Runtime views to show how software will handle:

- hazards, faults, and errors
- fault tolerance/reconfigurations
- performance
- data (e.g., quality, timeliness, ownership, access privileges)
- interface boundaries

Non-runtime views of software (vital to project planning, allocating work assignments, designing for modifiability, reusability, portability, extensibility, etc., facilitating incremental development, and a host of other critical purposes)

Architectural decisions and the rationale/implications/impact of those decisions on key system qualities



# Presentation Outline

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

Architecture and its Importance

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”)



# Requirements Development

## SG 1: Develop Customer Requirements

SP 1.1 Elicit Needs

SP 1.2 **Transform Stakeholder Needs into [Prioritized] Customer 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.”

## 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

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**.

## 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

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

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

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

## 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

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 defect

## SG 3: Verify Selected Work Products

SP 3.1 Perform Verification

SP 3.2 Analyze Verification Results



# 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

Architecture and its Importance

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*

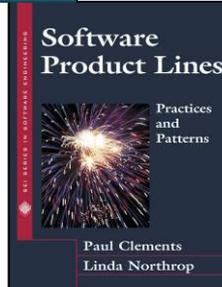
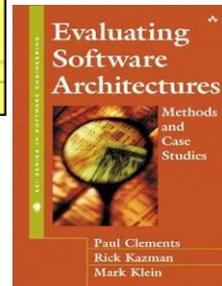
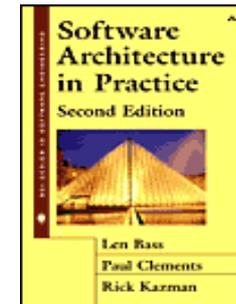
Clements, P.; Kazman, R.; & Klein, M. Reading, MA: Addison-Wesley, 2002.

*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, 2010.

*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

<i>Six Courses</i>	<i>Three Certificate Programs</i>			
	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		✓	✓	✓ : required to receive certificate
ATAM Leader Training			✓	
ATAM Observation			✓	*: available through e-learning



# Contact Information

## Larry Jones

Research, Technology, and Systems  
Solutions Program

Telephone: 719-481-8672

Email: [lgj@sei.cmu.edu](mailto:lgj@sei.cmu.edu)

## Mike Konrad

SEPM

Telephone: 412-268-5813

Email: [mdk@sei.cmu.edu](mailto: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





SEPG<sup>SM</sup> 2011  
NORTH AMERICA

*Perform at a Higher Level*



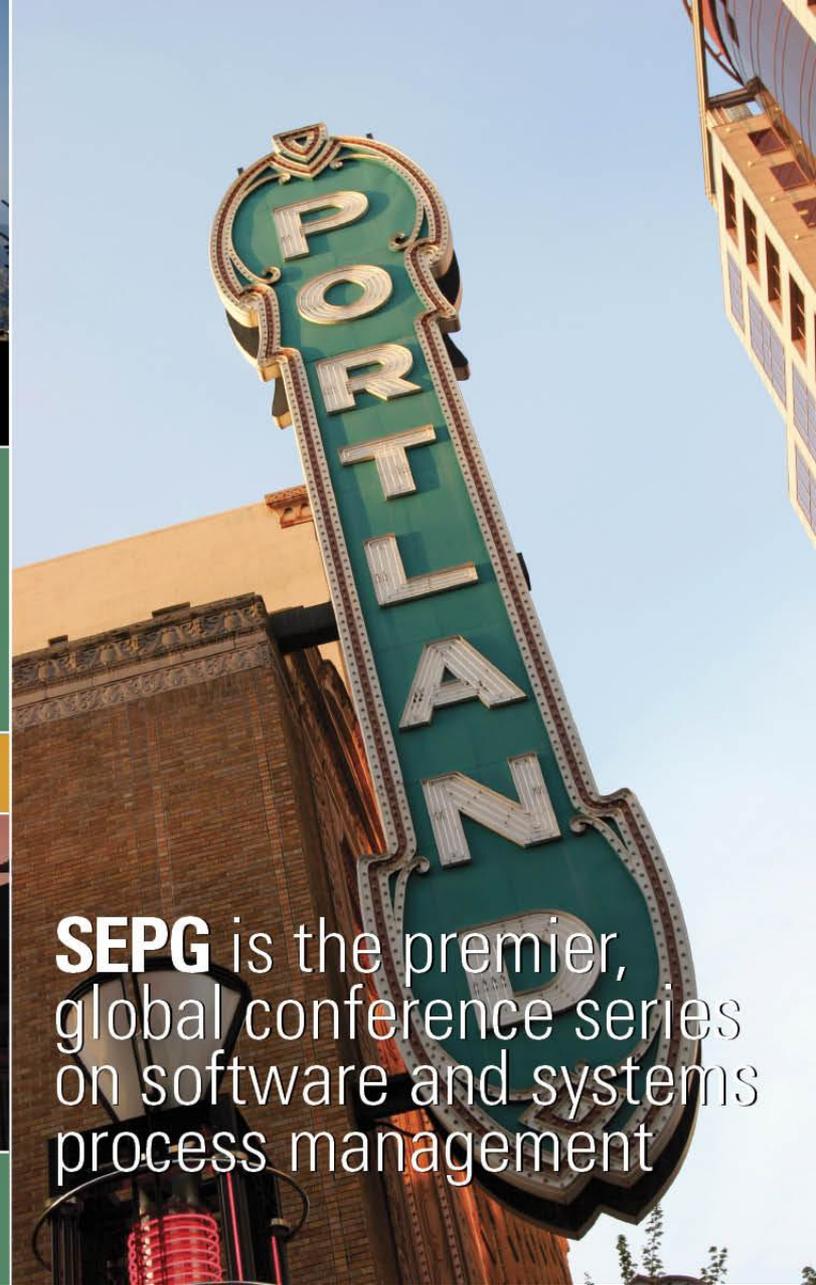
# The Power of Process

Coming to Portland in March 2011

SEPG North America 2011 | Oregon Convention Center | Portland, OR



[www.sei.cmu.edu/sepg/na/2011](http://www.sei.cmu.edu/sepg/na/2011)



**SEPG** is the premier, global conference series on software and systems process management



Software Engineering Institute

Carnegie Mellon

CMMI V1.3 and Architecture-Centric Engineering  
© 2011 Carnegie Mellon University



May 16-20, 2011 | San Mateo County, California



The SEI Architecture Technology User Network (SATURN) Conference brings together experts to exchange best architecture-centric practices in developing, acquiring, and maintaining software-reliant systems.

## 7 Things You Need to Know About the Next 7 Years in Architecture.



Architecture is Not Just for Architects



Architecture, Agile Development, and Business Agility



Soft Skills for Architects



Service-Oriented Architecture (SOA) and Cloud Computing



Architectural Knowledge Management



Architecting to Meet Tomorrow's Global Challenges



Model-Driven Architecting

# [www.sei.cmu.edu/saturn/2011](http://www.sei.cmu.edu/saturn/2011)



Software Engineering Institute | Carnegie Mellon

in collaboration with **Software**





## New SEI eLearning Portal Brings SEI Courses to You

The SEI is pleased to announce the new **SEI eLearning Portal**, a new platform for the development and delivery of SEI eLearning courses to conveniently meet your professional development needs.

The SEI eLearning Portal provides expert instruction as well as exercises, assessments, and other resources, creating a rich educational experience that is accessible by learners worldwide.

- learn at your own pace
- communicate easily with instructors
- access courses 24/7
- study at home, work, or on the road
- read materials online or download for later
- track your course progress

<http://www.sei.cmu.edu/training/elearning/>





- LIBRARY
- Music
  - Movies
  - TV Shows
  - Podcasts
  - Radio
- STORE
- iTunes Store
  - Purchased
- PLAYLISTS
- Party Shuffle
  - OEM Radio ---
  - Ambient
  - Books & Spoken
  - Classical
  - Downtempo
  - Electro
  - Electronic
  - Eno

Podcasts > CERT's Podcast Series: Security for Business Leaders

## CERT's Podcast Series: Security for Business Leaders



**CERT**  
 Category: Tech News  
 Language: English

Free

PODCAST DESCRIPTION

▲	Name	Time	Art
1	Convergence: Integrating Physical...	28:43	B, C
2	IT Infrastructure: Tips for Navigat...		
3	The Value of De-Identified Perso...		
4	Adapting to Changing Risk Enviro...		

CERT's Podcast Series

http://www.cert.org/podcast/undockplayer.html

- Mitigating Insider Threat: New and Improved Practices: 08.18.2009 - Featuring Dawn Cappelli
- Analyzing Internet Traffic for Better Cyber Situational Awareness: 07.28.2009 - Featuring Derek Gabbard
- Rethinking Risk Management: 07.07.2009 - Featuring Chris Alberts
- The Upside and Downside of Security in the Cloud: 06.16.2009 - Featuring Tim Mather
- More Targeted, Sophisticated Attacks: Where to Pay Attention: 05.26.2009 - Featuring Marty Lindner
- Is There Value in Identifying Software Security "Never Events?": 05.05.2009 - Featuring Robert Charette
- Cyber Security, Safety, and Ethics for the Net Generation: 04.14.2009 - Featuring Rodney Petersen
- An Experienced-Based Maturity Model for Software Security: 03.31.2009 - Featuring Gary McGraw
- Mainstreaming Secure Coding Practices: 03.17.2009 - Featuring Robert Seacord
- Security: A Key Enabler of Business Innovation: 03.03.2009 - Featuring Roland Cloutier
- Better Incident Response Through Scenario Based Training: 02.17.2009 - Featuring Chris May
- An Alternative to Risk Management for Information and Software Security: 02.03.2009 - Featuring Brian Chess

Waiting for www.cert.org...



***CERT's Podcast Series:  
 Security for Business Leaders***

**▶ [www.cert.org/podcast/](http://www.cert.org/podcast/)**





Want a Closer Connection to the SEI?

Become an SEI Member!

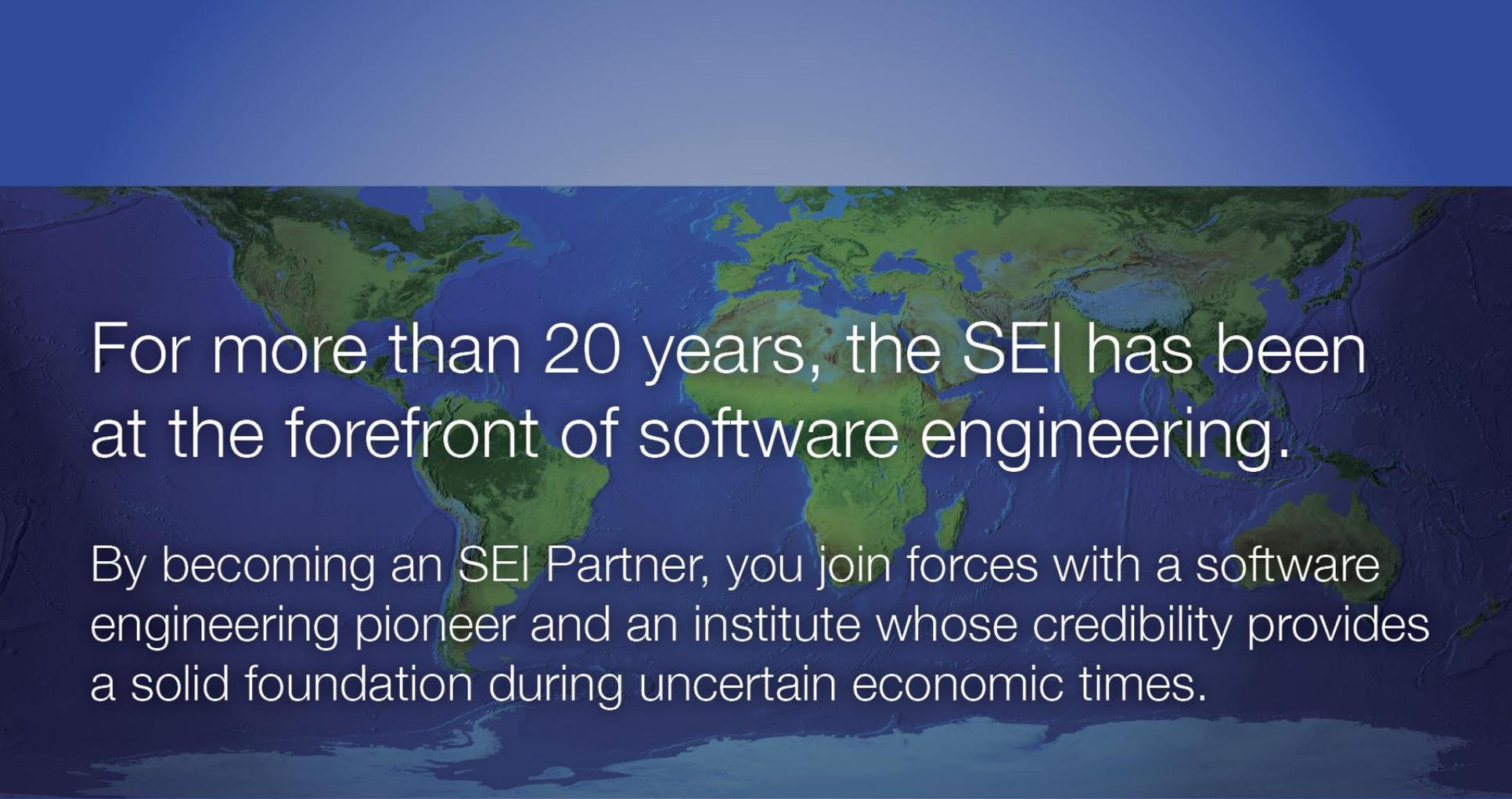
▶ [www.sei.cmu.edu/membership](http://www.sei.cmu.edu/membership)



**Software Engineering Institute**

**Carnegie Mellon**

CMMI V1.3 and Architecture-Centric  
Engineering  
© 2011 Carnegie Mellon University

A world map with a blue-to-green color gradient, showing the continents. The map is centered on the Atlantic Ocean, with North and South America on the left and Europe, Africa, and Asia on the right.

For more than 20 years, the SEI has been at the forefront of software engineering.

By becoming an SEI Partner, you join forces with a software engineering pioneer and an institute whose credibility provides a solid foundation during uncertain economic times.

SEI Partner Network

▶ [www.sei.cmu.edu/partners](http://www.sei.cmu.edu/partners)





Do you have the knowledge you need?

SEI Training

▶ [www.sei.cmu.edu/training](http://www.sei.cmu.edu/training)



**Software Engineering Institute**

**Carnegie Mellon**

CMMI V1.3 and Architecture-Centric  
Engineering  
© 2011 Carnegie Mellon University

## 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 Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

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.

