An Architecture-Centric Approach for Acquiring Software-Reliant Systems

NPS Acquisition Research Symposium
11-12 May 2011

John Bergey
Larry Jones

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

Software Engineering Institute | Carnegie Mellon

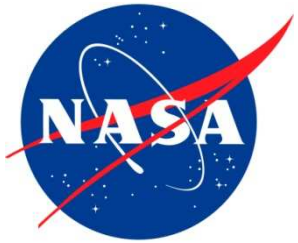
© 2011 Carnegie Mellon University

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.

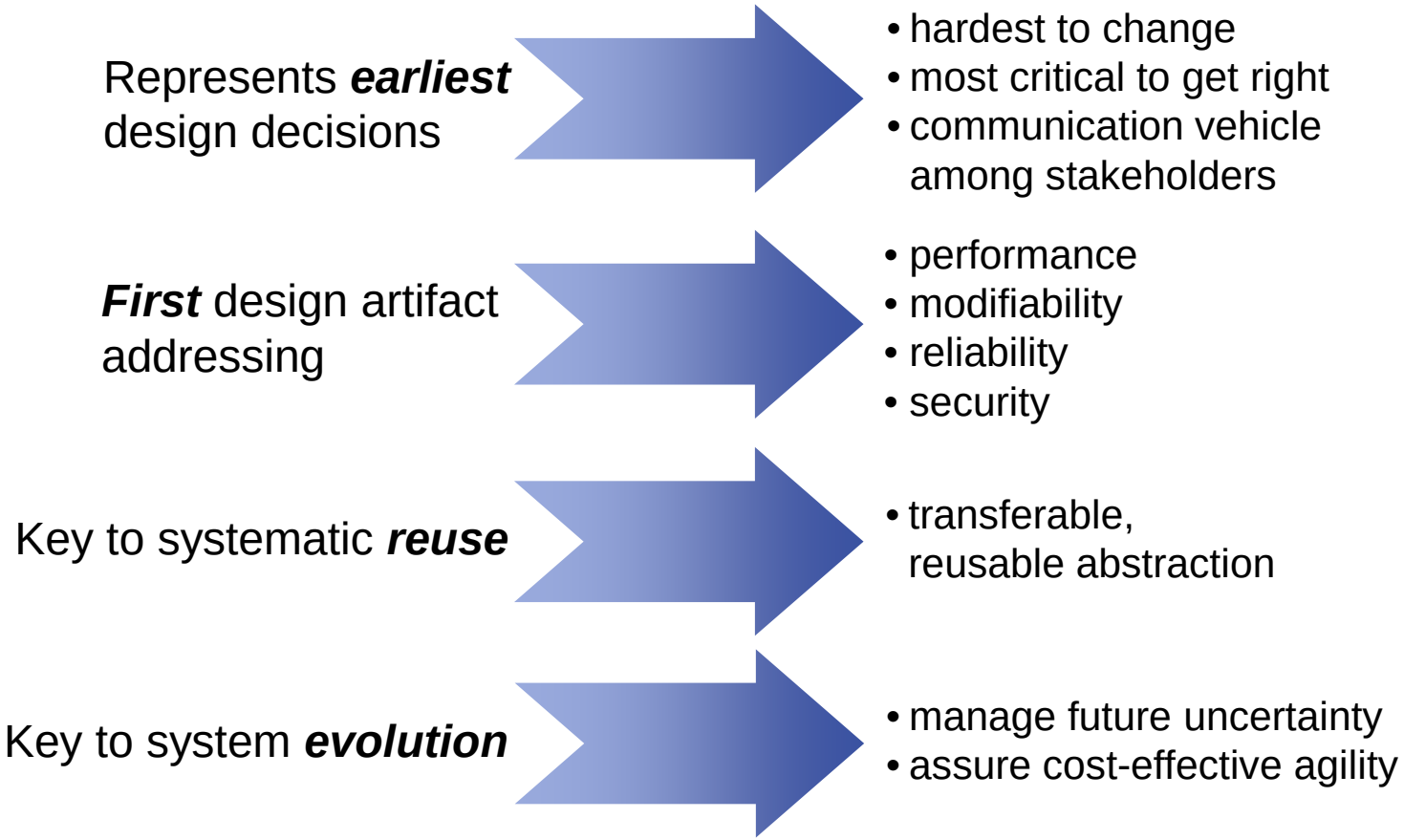
NDIA
National Defense Industrial Association



NRC The National Research Council

© 2011 Carnegie Mellon University

Why Is Architecture Important?



The **right architecture** paves the way for system **success**.
The **wrong architecture** usually spells some form of **disaster**.

© 2011 Carnegie Mellon University

Why is an Architecture-Centric Acquisition Approach Needed?

Studies have shown that acquisition practices have not kept up with architecture practices.

Architecture-centric acquisition can reduce acquisition risk.

KPPs, KSAs and TRLs can be evaluated earlier in the life cycle.

Architecture-centric acquisition can facilitate needed synergy between systems and software engineering.

The efficacy of the software architecture has a direct impact on the war fighter.

© 2011 Carnegie Mellon University

Presentation Outline

Software Architecture Basics

Architecture-Centric Engineering

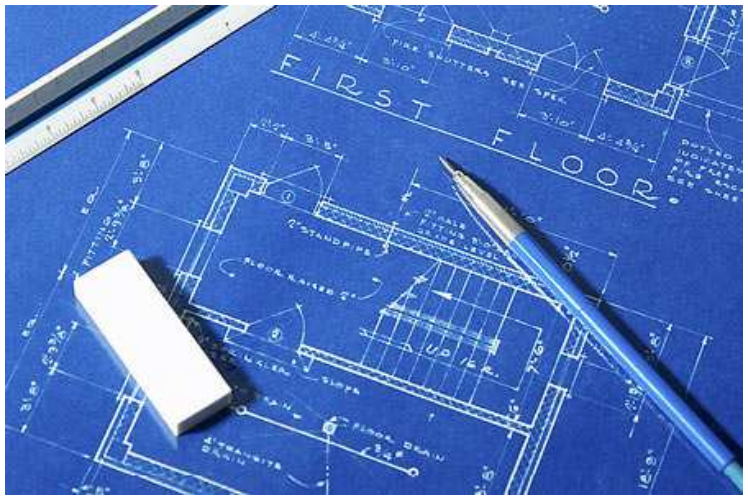
Architecture-Centric Acquisition

Conclusion

© 2011 Carnegie Mellon University

What Is an Architecture?

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



© 2011 Carnegie Mellon University

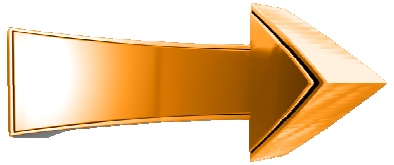
Formal Definition of Software Architecture

"*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

© 2011 Carnegie Mellon University

Implications



Architecture is an abstraction of a system.

Architecture defines the properties of elements.

Systems can and do have many structures.

Every software-reliant 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!

© 2011 Carnegie Mellon University

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.

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

analysis, design, development, evolution

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

has these qualities

Non-functional Requirements

© 2011 Carnegie Mellon University

Specifying Quality Attributes

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

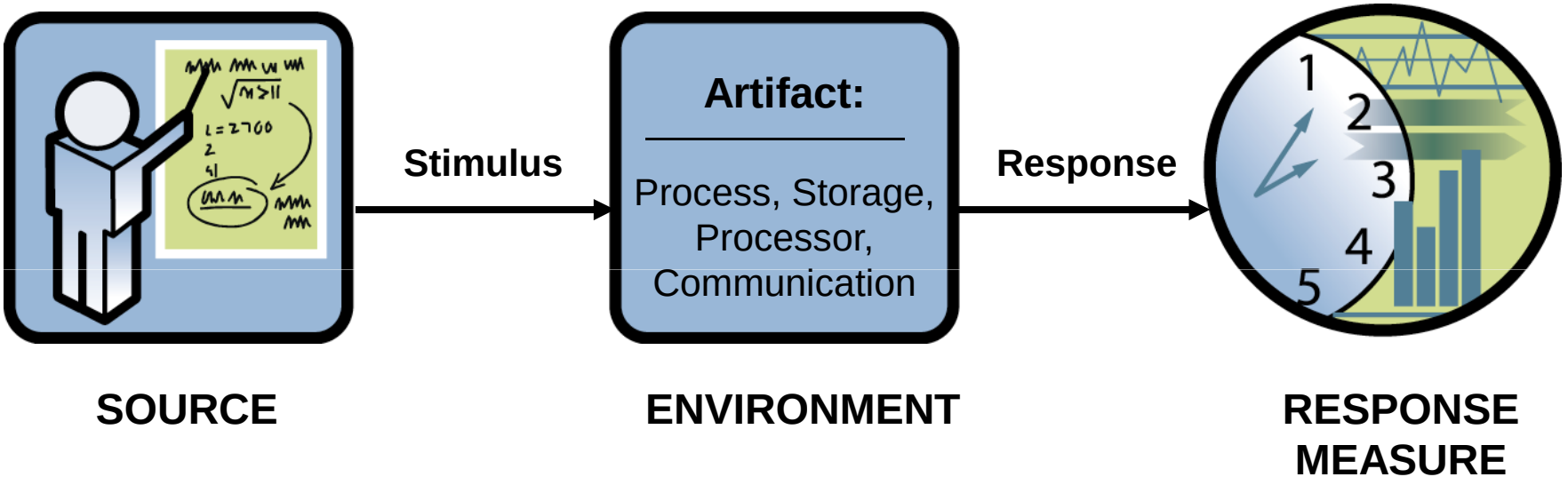
Just citing the desired qualities is not enough; it is meaningless to say that the system shall be “modifiable” or “interoperable” or “secure” without details about the context.

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.

© 2011 Carnegie Mellon University

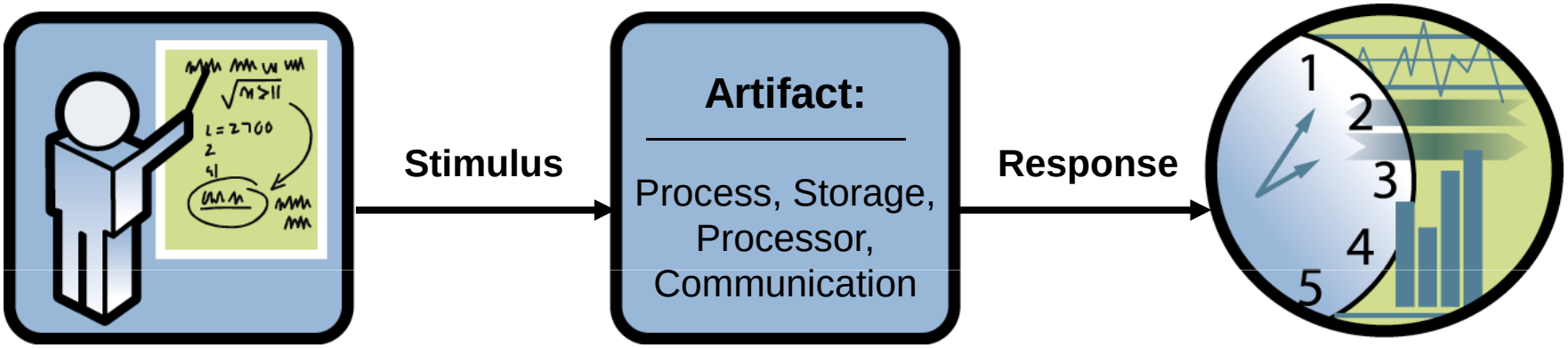
Parts of a Quality Attribute Scenario



© 2011 Carnegie Mellon University

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

© 2011 Carnegie Mellon University

Presentation Outline

Software Architecture Basics

Architecture-Centric Engineering

Architecture-Centric Acquisition

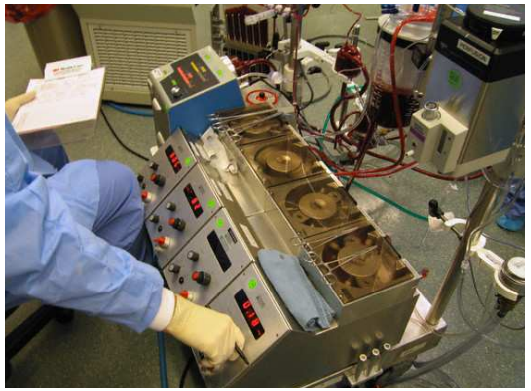
Conclusion

© 2011 Carnegie Mellon University

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.*



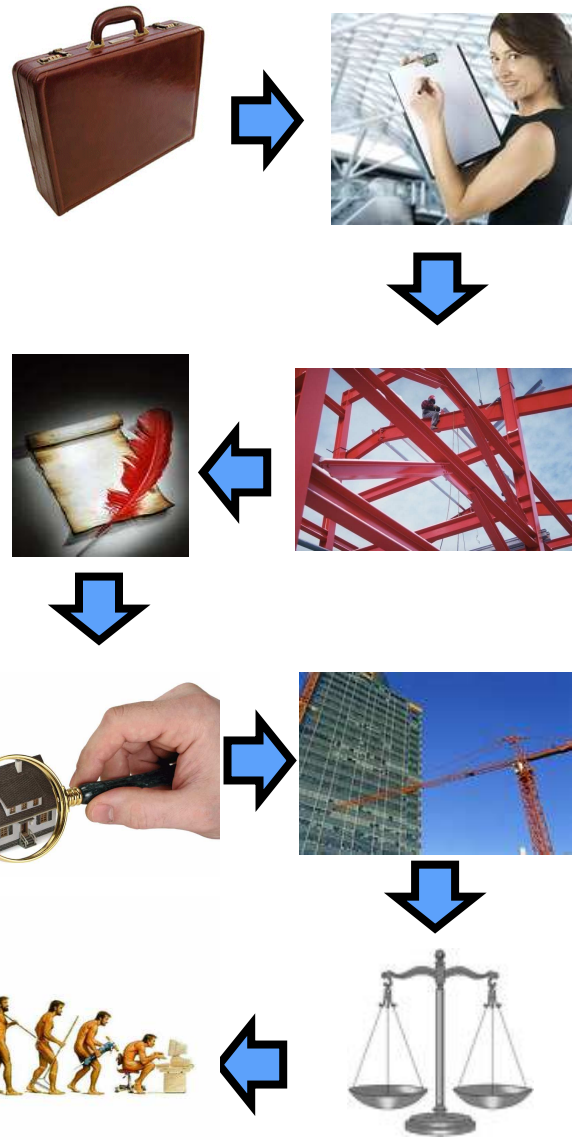
The **SEI ACE Initiative** develops principles, methods, foundations, techniques, tools, and materials in support of creating, fostering, and stimulating widespread transition of the ACE discipline.

© 2011 Carnegie Mellon University

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**



© 2011 Carnegie Mellon University

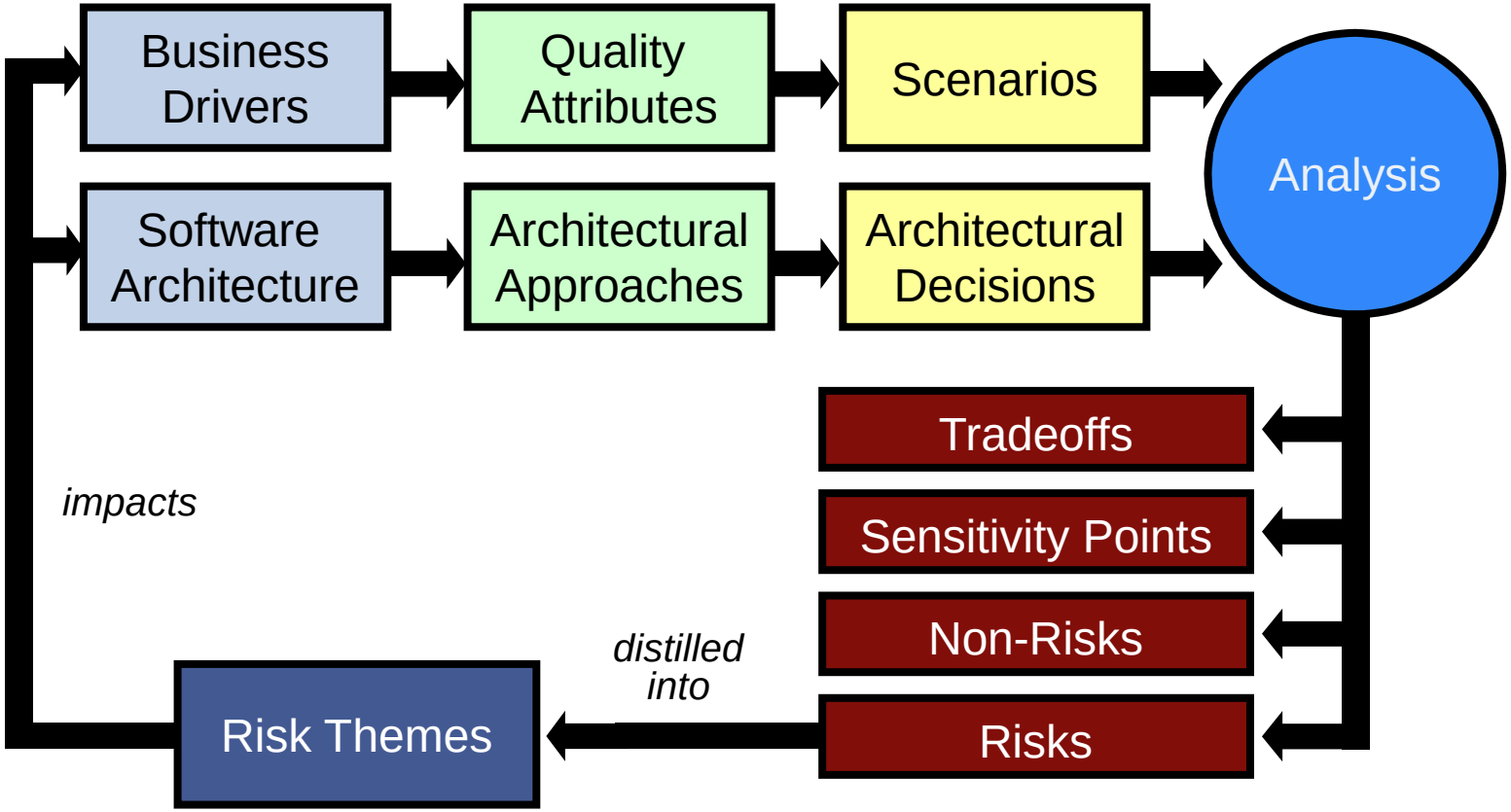
Some SEI Techniques and Methods of Particular Interest to Acquisition Organizations

understanding the **requirements**	*Quality Attribute Workshop (QAW)* *Mission Thread Workshop (MTW)*
analyzing or evaluating the architecture	*Architecture Tradeoff Analysis Method (ATAM);*
documenting and communicating the architecture	*Views and Beyond Approach*

© 2011 Carnegie Mellon University

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

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



© 2011 Carnegie Mellon University

Presentation Outline

Software Architecture Basics

Architecture-Centric Engineering

Architecture-Centric Acquisition

- **definitions and key elements**
- **effect on system evaluation**
- **an acquisition example**
- **application of the new practices**

Conclusion

© 2011 Carnegie Mellon University

What is Architecture-Centric Acquisition?

Architecture-Centric Acquisition is the act of *using architecture and architecture-centric practices as a contractual means* to reduce risk and gain early confidence that the system being acquired will meet its mission goals.

[Bergey 2010]

© 2011 Carnegie Mellon University

Key Elements of an Architecture-Centric Acquisition Approach

Architecture-centric acquisition involves:

- Determining the system's architecturally significant requirements and specifying them in a meaningful way
- Commissioning the development of the architecture and ensuring it is appropriately documented
- Evaluating the architecture to determine its suitability to support the architecturally significant requirements and …
  - Mission (and Business) Goals
  - Key Performance Parameters (KPPs)
  - Technology Readiness Levels (TRLs)
- Leveraging other promising *architecture-related practices* so a program office can perform its acquisition responsibilities more effectively

© 2011 Carnegie Mellon University

Earlier System Evaluation

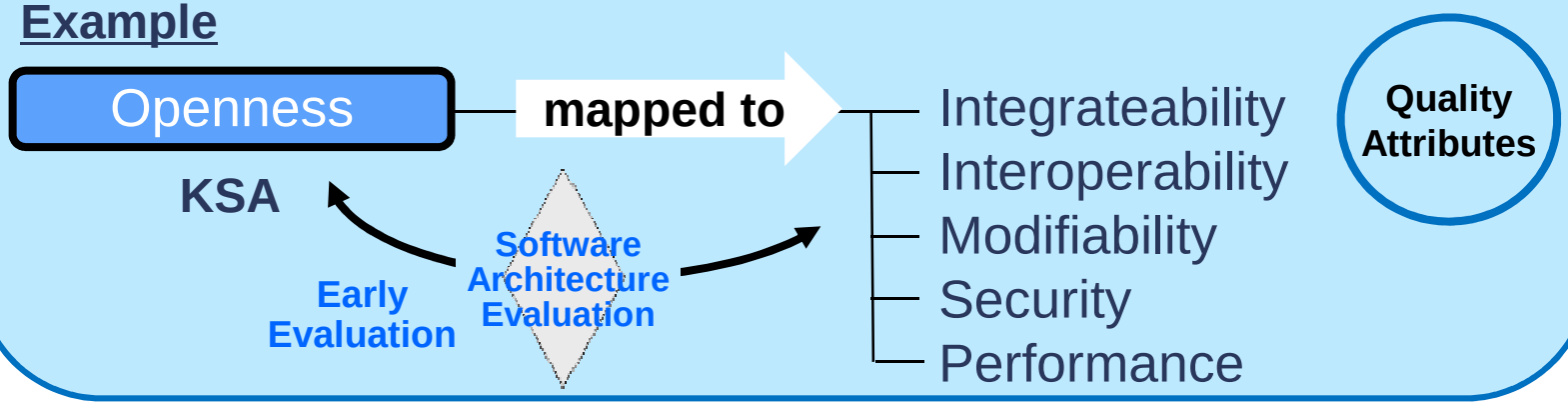
An architecture-centric acquisition can help a Program Office evaluate:

- Key Performance Parameters (KPPs)
- **Key System Attributes (KSAs)**
- Technology Readiness Levels (TRLs)

A **KSA** is an attribute or characteristic considered crucial in achieving a KPP or some other key performance attribute deemed necessary by the sponsor. **KSAs** provide decision makers with an additional level of capability performance characteristics below the KPP level.

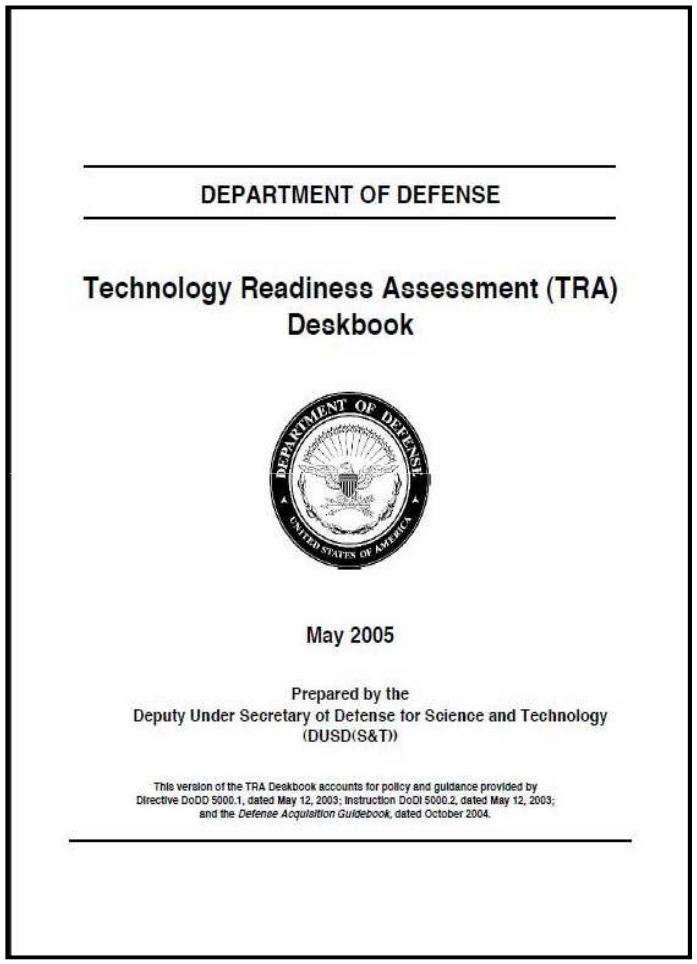
A **KSA** can often be mapped to one or multiple Quality Attributes.

Example



© 2011 Carnegie Mellon University

Technology Readiness Levels (TRLs)



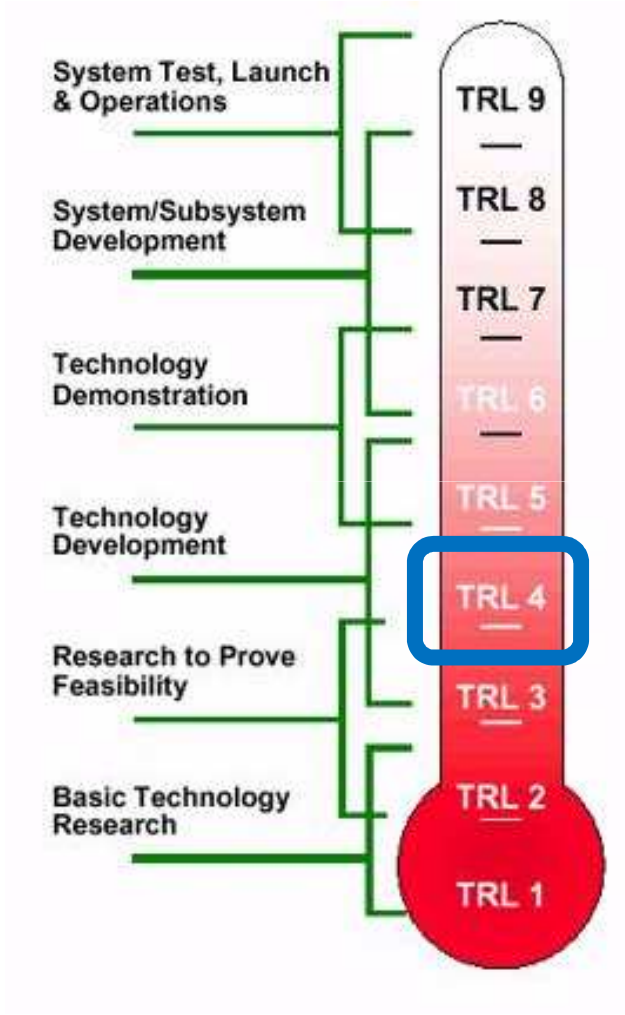
DEPARTMENT OF DEFENSE

Technology Readiness Assessment (TRA) Deskbook

May 2005

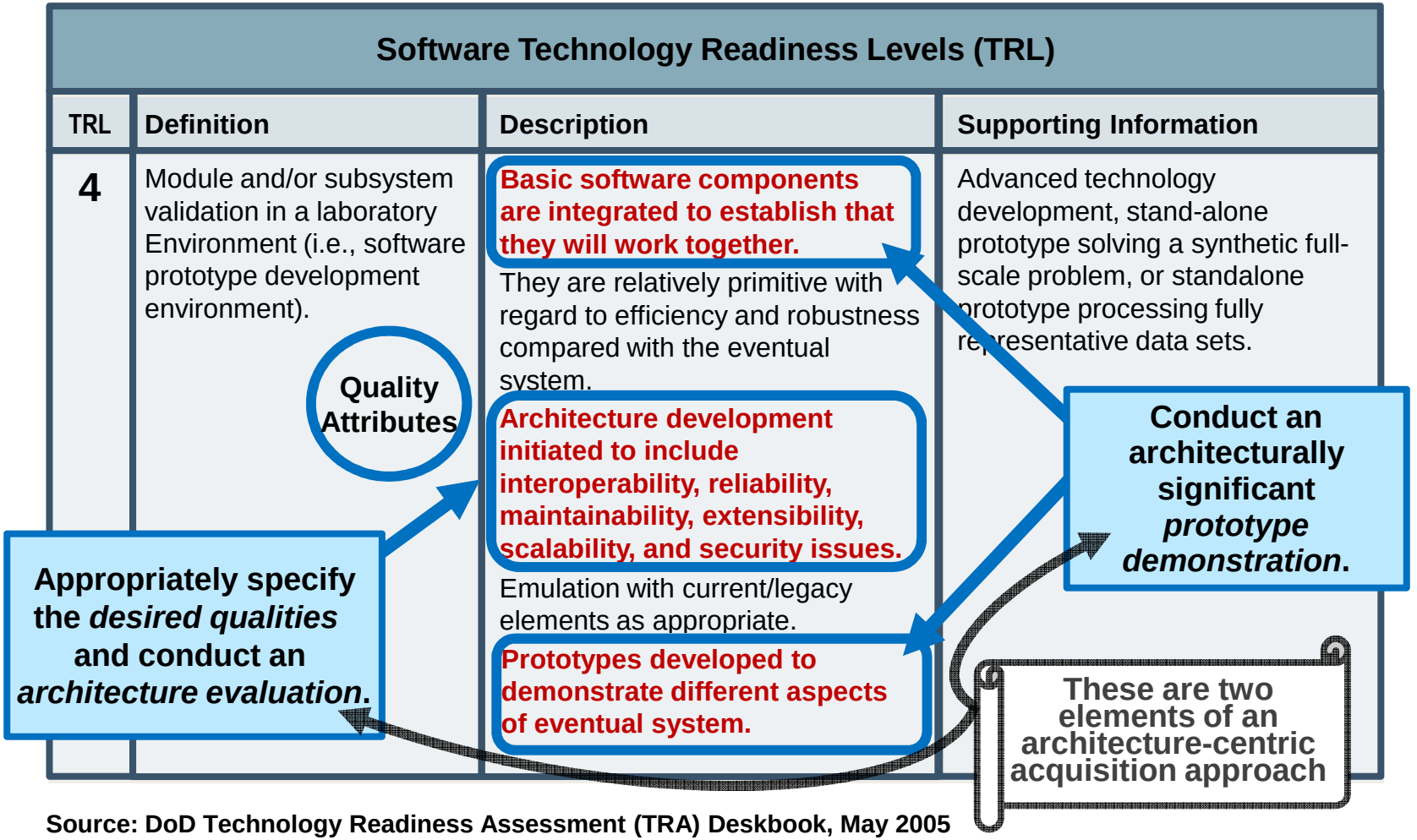
Prepared by the
Deputy Under Secretary of Defense for Science and Technology
(DUSD(S&T))

This version of the TRA Deskbook accounts for policy and guidance provided by Directive DoDD 5000.1, dated May 12, 2003; Instruction DoDI 5000.2, dated May 12, 2003; and the *Defense Acquisition Guidebook*, dated October 2004.



© 2011 Carnegie Mellon University

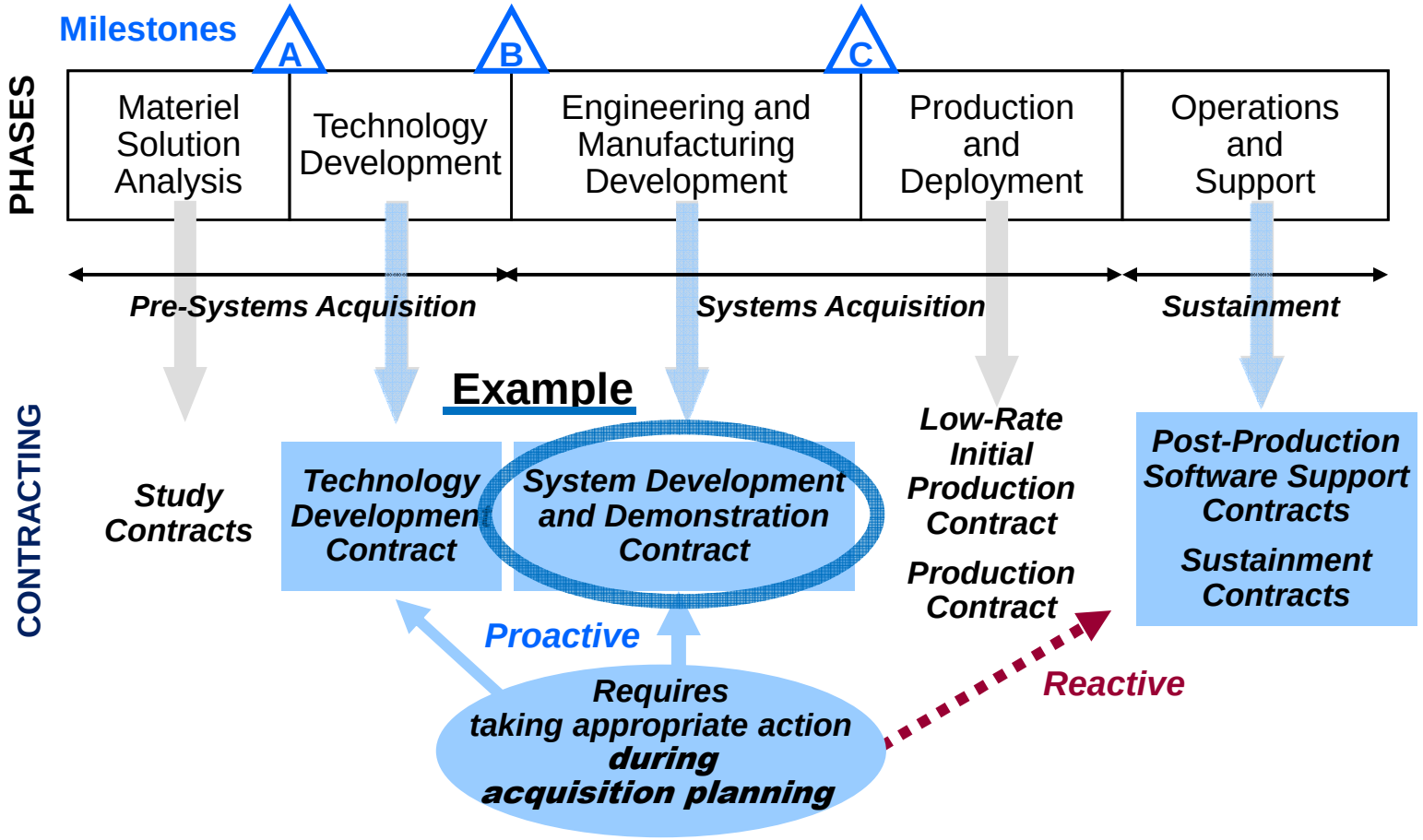
Technology Readiness Level 4



Source: DoD Technology Readiness Assessment (TRA) Deskbook, May 2005

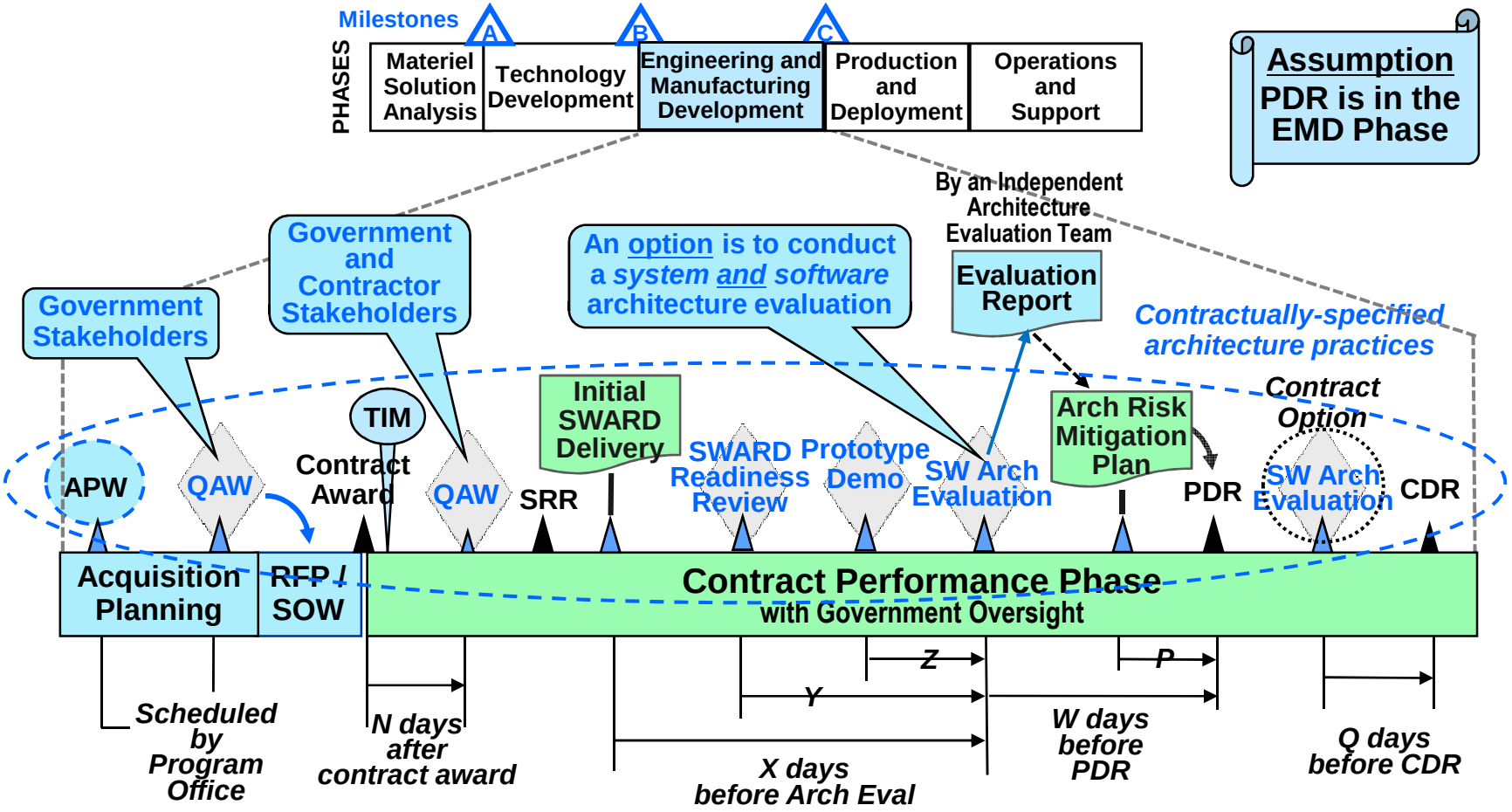
© 2011 Carnegie Mellon University

Incorporating Architecture-Centric Practices



© 2011 Carnegie Mellon University

An Architecture-Centric Acquisition Example



Legend					
	APW	– Acquisition Planning Workshop		SRR	– System Requirements Review
	CDR	– Critical Design Review		SWARD	– SoftWare ARchitecture Description Document
	PDR	– Preliminary Design Review		TIM	– Technical Interchange Meeting

© 2011 Carnegie Mellon University

New Practices Being Piloted or Explored

New architecture-centric practices that are being explored or piloted include:

- Concurrently *evaluating proposed architectures* of two development contractors *during a competitive down select*
- Incorporating an *architecture competency skills survey* as part of a competitive acquisition
- An architecture-centric approach as part of a *product line acquisition*
- An *architecture-driven test approach* to better focus testing efforts so they are more effective from an importance/time/cost standpoint
- Taking *remedial action in the O&S* phase* to motivate a recalcitrant legacy system contractor to adopt good architecture practices
- Incorporating a set of *architecturally significant metrics* and an *architecture improvement roadmap* in a system acquisition
- Incorporating *model-based development* as part of an architecture-centric approach

* O&S: Operations and Support phase of the DoD 5000 acquisition life cycle

© 2011 Carnegie Mellon University

Presentation Outline

Software Architecture Basics

Architecture-Centric Engineering

Architecture-Centric Acquisition

Conclusion

© 2011 Carnegie Mellon University

Summary

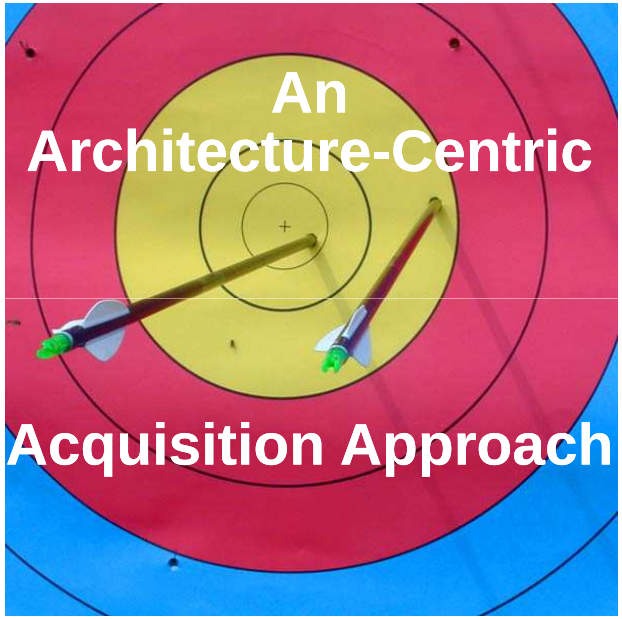
An architecture-centric acquisition approach

- provides early insight into critical requirements and design decisions that drive the entire development effort
- provides a proven and effective means for discovering software design risks and risk themes
- enables risks to be mitigated earlier and more cost effectively
- results in fewer test and integration problems and costly rework downstream
- provides the knowledge base needed for cost-effective system evolution and sustainment
- Provides a focal point that aligns with a program office's responsibilities and limited resources, time available, and key contractual events

Enables an acquisition organization to perform its contract management and technical monitoring responsibilities with greater effectiveness.

© 2011 Carnegie Mellon University

Your Choice



or

© 2011 Carnegie Mellon University

Contact Information

John Bergey

Research, Technology, and Systems Solutions Program

Telephone: 215-348-0530

Email: jkb@sei.cmu.edu

Larry Jones

Research, Technology, and Systems Solutions Program

Telephone: 719-481-8672

Email: lgj@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

© 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.

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.

© 2011 Carnegie Mellon University

Back up slides

© 2011 Carnegie Mellon University

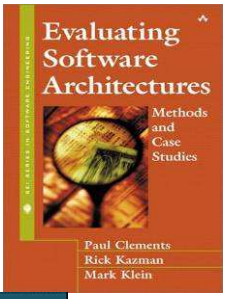
References - 1

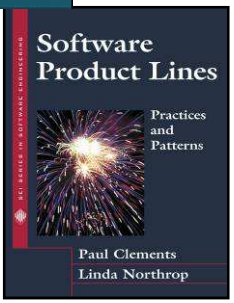
Software Architecture in Practice, Second Edition
Bass, L.; Clements, P.; & Kazman, R. Reading, MA: Addison-Wesley, 2003.

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.





© 2011 Carnegie Mellon University

References - 2

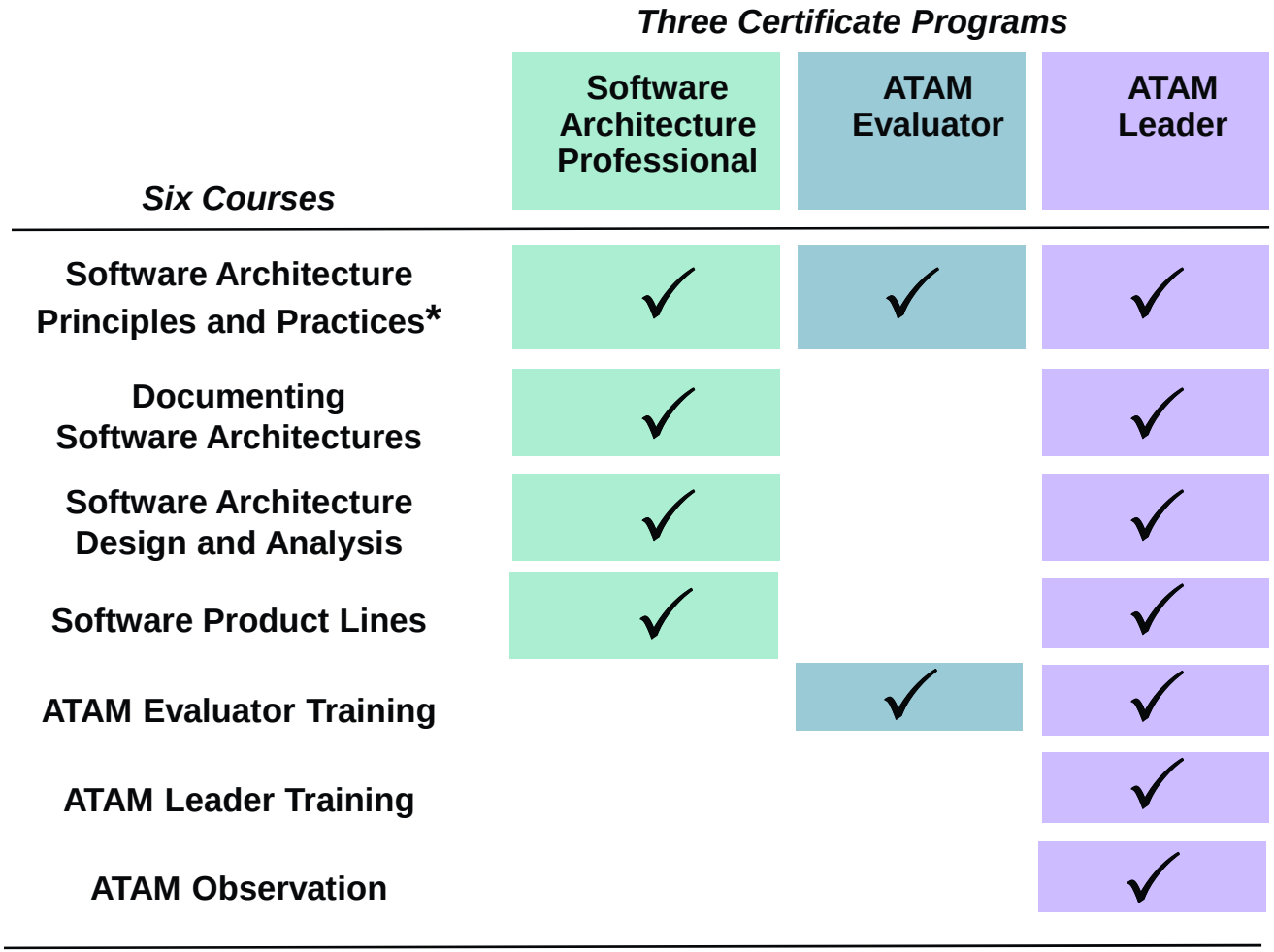
Kazman, R. & Bass, L. *Categorizing Business Goals for Software Architectures* (CMU/SEI-2005-TR-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/reports/05tr021.pdf

Nord, R.; Bergey, J.; Blanchette, S. & Klein, M. *Impact of Army Architecture Evaluations* (CMU/SEI 2009-SR-007). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, April 2009. http://www.sei.cmu.edu/reports/09sr007.pdf

Bergey, J. *A Proactive Means for Incorporating a Software Architecture Evaluation in a DoD System Acquisition* (CMU/SEI-2009-TN-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July 2009. http://www.sei.cmu.edu/reports/09tn004.pdf

© 2011 Carnegie Mellon University

The SEI Software Architecture Curriculum



© 2011 Carnegie Mellon University

Some SEI Techniques, Methods, and Tools

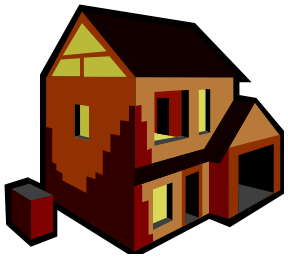
creating the **business case** for the system	Pedigreed Attribute eLicitation Method (PALM)
understanding the **requirements**	*Quality Attribute Workshop (QAW) * Mission Thread Workshop (MTW) **
creating and/or selecting the architecture	*Attribute-Driven Design (ADD) and ArchE*
documenting and communicating 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**

© 2011 Carnegie Mellon University

Structures and Views

One house, many views



- Carpentry view
- Plumbing view
- Electrical view
- Ductwork view

No single view accurately represents the house.

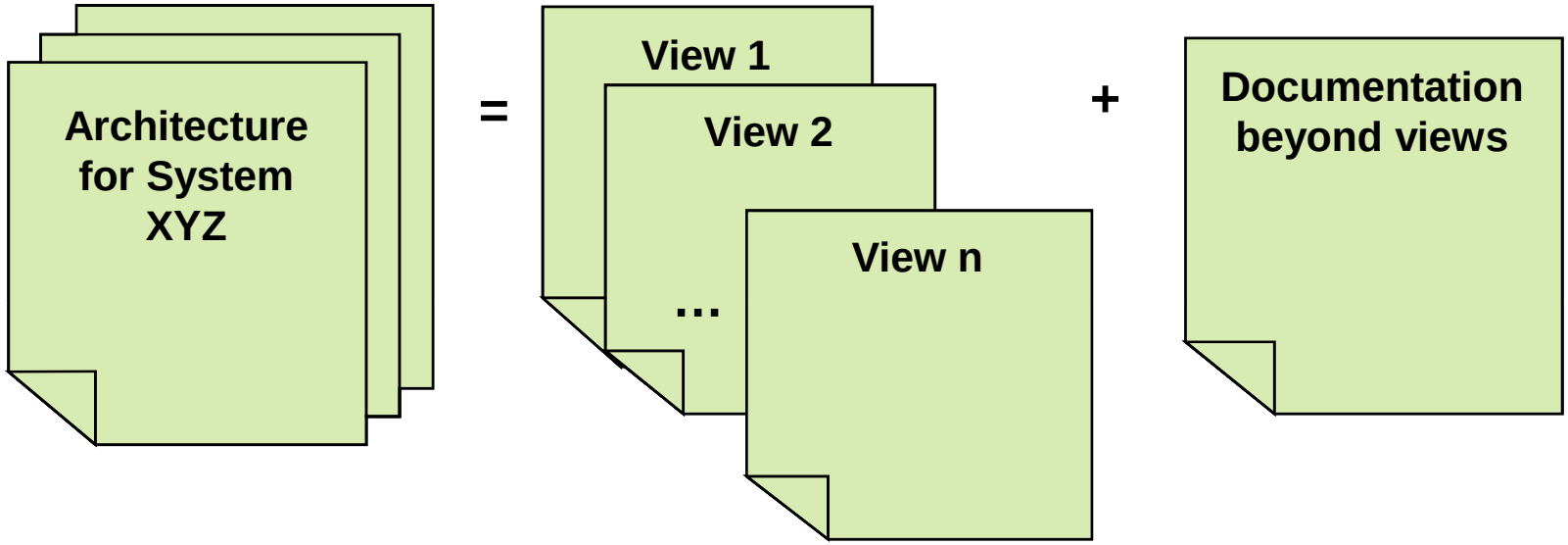
No single view can be used to build the house.

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

© 2011 Carnegie Mellon University

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.

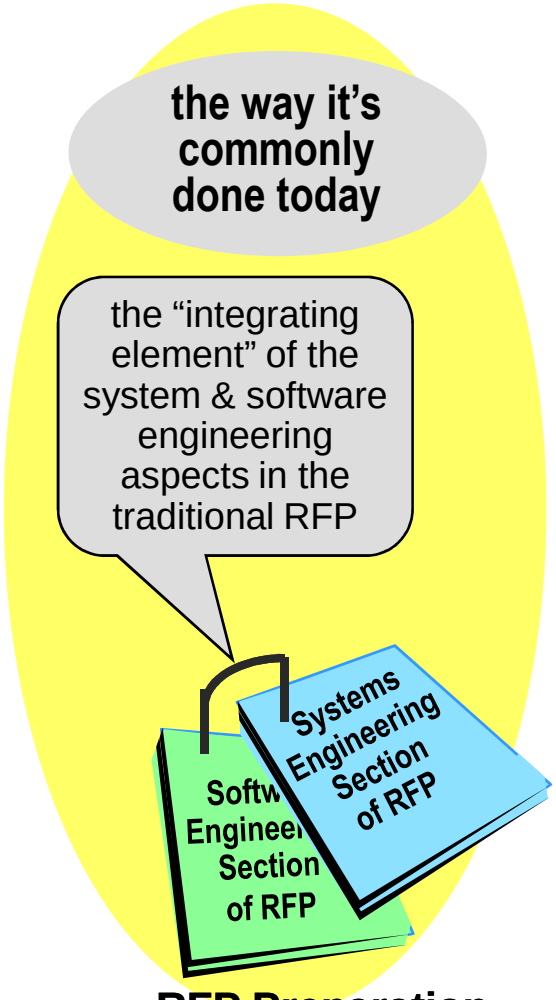
© 2011 Carnegie Mellon University

Typical Acquisition Impact

BEFORE: There is no software architecture documentation. **AFTER: A software architecture description document is a contract deliverable.**
BEFORE: The development contractor presents a couple of PowerPoint box-and-line drawings to describe the architecture and high-level software design. **AFTER: The software architecture description includes a comprehensive set of views (e.g., module decomposition, allocation, run-time) that can be suitably analyzed.**
BEFORE: The system's non-functional (i.e., quality) requirements that greatly impact the architecture design and software implementation are poorly defined. **AFTER: The system's quality requirements have been specified by key stakeholders in terms of a clear and concise set of quality attribute scenarios that are testable.**
BEFORE: The proposed software design is not appropriately analyzed or evaluated. **AFTER: The software architecture is evaluated with stakeholder participation and risks (and risk themes) are subsequently identified and appropriately documented so they can be mitigated early and cost effectively.**
BEFORE: Software reviews are largely perfunctory and based on checklists and PowerPoint presentations. **AFTER: During the Preliminary Design Review (PDR), the development contractor describes architecture evaluation results and presents its risk mitigation plan.**
BEFORE: Plans for system evolution are ad hoc and estimates are often unreliable. **AFTER: The development contractor manages quality requirements, architectural changes, and risks and follows an architectural improvement roadmap.**

© 2011 Carnegie Mellon University

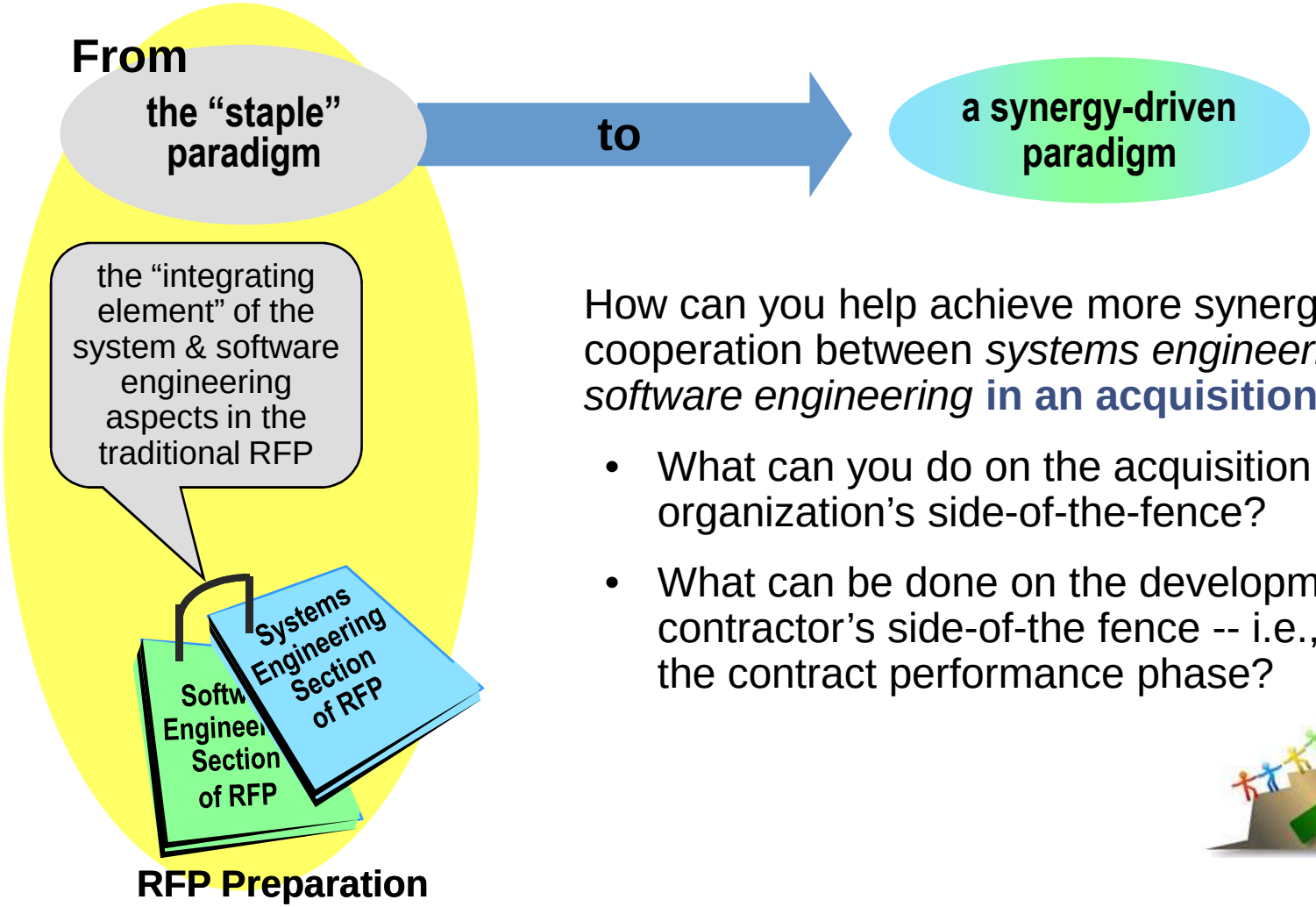
Integration of Systems Engineering and Software Engineering in an RFP



RFP Preparation

© 2011 Carnegie Mellon University

Promoting System Engineering and Software Engineering Congruency

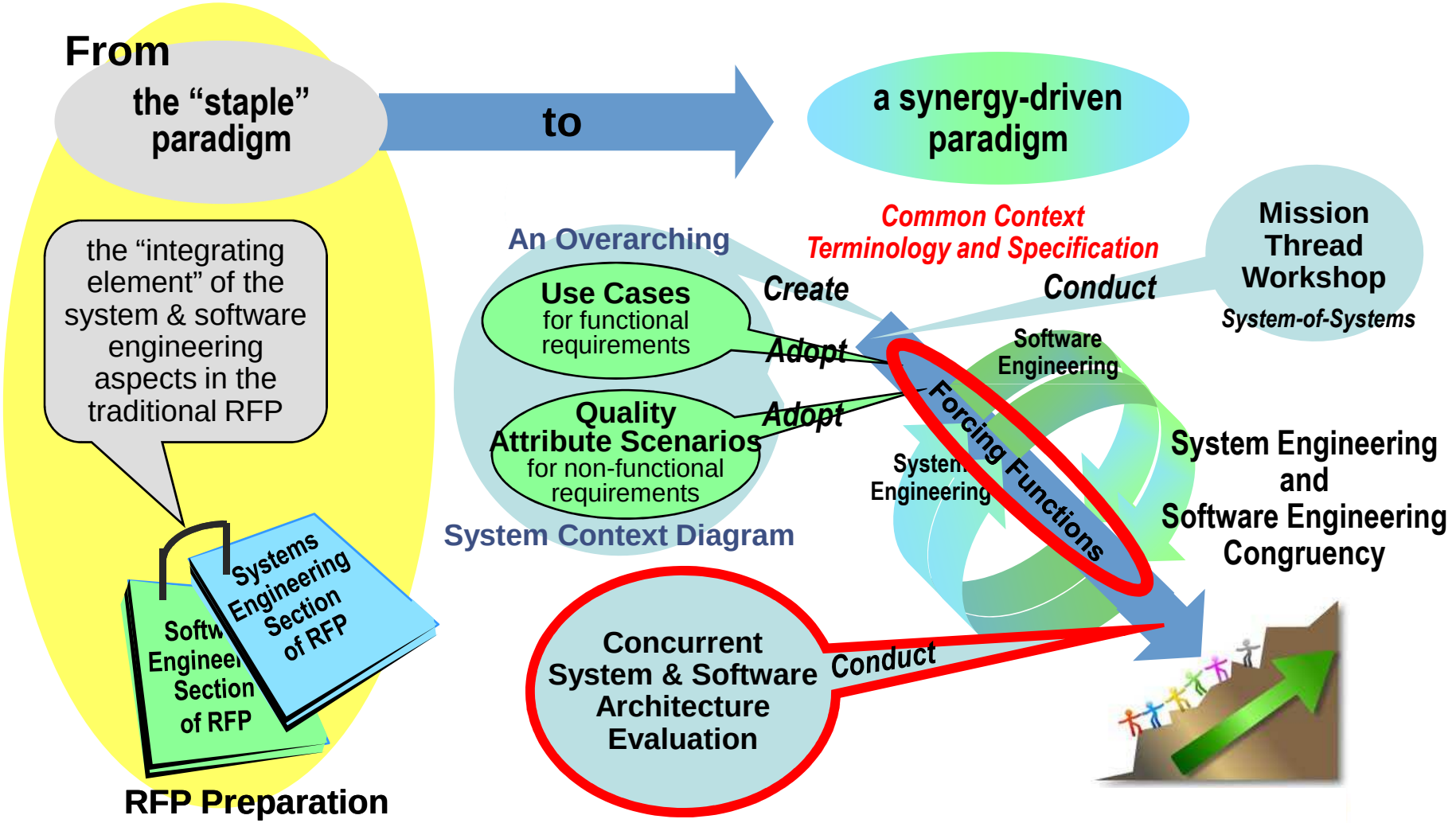


How can you help achieve more synergy and cooperation between *systems engineering* and *software engineering* **in an acquisition**?

- What can you do on the acquisition organization's side-of-the-fence?
- What can be done on the development contractor's side-of-the fence -- i.e., during the contract performance phase?

© 2011 Carnegie Mellon University

Promoting System Engineering and Software Engineering Congruency



© 2011 Carnegie Mellon University