Software Architecture Fundamentals: Technical, Business, and Social Influences

Rob Wojcik
Senior Technical Staff

Rob is a senior member of the technical staff in the Research, Technology, and System Solutions Program at the Carnegie Mellon University’s Software Engineering Institute (SEI), a position he has held since 2004. In his current position, he performs training and consulting in software architecture technology and software architecture evaluations.

See his full bio at:
www.sei.cmu.edu/go/architecting-software-the-sei-way
Research, Technology, and System Solutions (RTSS) Program

Vision

- Enable assured and flexible system capabilities at all scales.

Mission

- Focus on the structure and behavior of software-reliant systems
Portfolio of RTSS Program Work

Initiatives

- Architecture-Centric Engineering
- System of Systems Practice
- System of Systems Software Assurance
- Product Line Practice

Cross-Cutting Efforts

- Concept Lab
- Integrating Solutions
- Ultra-Large-Scale Systems

Independent Research and Development
Today’s Topics

• What is Software Architecture?
• Why is Software Architecture Important?
• Which Requirements Are Most Important To Architectural Design?
• What Else Influences Software Architecture?

*I’ll take questions at the end of the presentation.*
Polling Question #1

How Much Do You Know?

a) I know a whole lot about software architecture
b) I know enough about software architecture to get by
c) I know very little about software architecture
d) What the heck is software architecture?
Today’s Topics

- What is Software Architecture?
- Why is Software Architecture Important?
- Which Requirements Are Most Important To Architectural Design?
- What Else Influences Software Architecture?
**Many Definitions for Software Architecture**

**Pandey:** the blueprint of the various framework components that coordinate together to satisfy the design guidelines of a specific domain

**Alfred:** Software architecture consists of the rules and principles for how a system is decomposed into its component parts, the responsibilities are allocated among those parts, the policies, and mechanisms that coordinate the interactions between those parts as they collaborate to fulfill the needs of the business, in terms of cost to develop/maintain the software components, time to market, and life expectancy of the components

**Adabala:** a style that is proven scientifically and adopted by the engineering discipline, with which software is developed as to sustain and adapt to the growing needs of the industry from time to time...

**ANSI/IEEE:** the fundamental organization of a system, embodied in its components, their relationships to each other, and to the environment, and the principles governing its design and evolution

**Mulvaney:** a set of implementation elements together with the mechanisms through which the elements collaborate to provide the system’s required functional and non-functional aspects, and the organizational structure that guides this organization---these elements and their interfaces, their collaborations, and their composition

**RUP:** the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with the operations specified in the collaborations among those structural elements and the composition of these structural elements into progressively larger subsystems, and the architecture style that guides this organization---these elements and their interfaces, their collaborations, and their composition

**Ahmed:** a coherent set of abstract patterns, or principles, guiding the design of each aspect of a large software system...

**Matthaeus:** a configurable skeleton of any kind of software beast on which you hang implementation specific muscle to make it live
Software architecture

- is an abstraction that describes software elements
- addresses the roles, responsibilities, behaviors and properties of software elements
- addresses the relationships between software elements
- shows what software elements provide to and require from each other
- shows the relationship to non-software elements
- is described from many different perspectives
Some Things Remain Certain - 2

Every software system has an architecture.

A software architecture is not inherently good or bad.
Today’s Topics

• What is Software Architecture?
• **Why is Software Architecture Important?**
• Which Requirements Are Most Important To Architectural Design?
• What Else Influences Software Architecture?
Why Is Software Architecture Important?

It’s a vehicle for communication.

It’s a manifestation of earliest design decisions that

• defines implementation constraints
• relates to organizational structure
• provides the basis for project artifacts and activities
• permits/precludes achieving requirements
• allows us to predict system qualities
• allows us to control complexity
• allows us to reason about and manage change
• allows us to develop a skeletal system
• provides a sound basis for cost and schedule estimates

It’s a transferable, reusable abstraction.
Today’s Topics

• What is Software Architecture?
• Why is Software Architecture Important?
• **Which Requirements Are Most Important To Architectural Design?**
• What Else Influences Software Architecture?
Which Requirements Are Most Important to Architectural Design?

What determines whether these requirements are met?
Which requirements are the most important when it comes to structuring an architecture?
Something to Consider

What’s wrong with designing a system that has one big source module, one big object module, and one big executable as long as it functions properly?

- buildability
- modifiability
- testability
- complexity
- maintainability
- portability
- reliability
- distributability
- availability
- reusability

Others?

Which requirements do you think would be negatively impacted by this “design”?
Here’s the Point!

If functionality is the only thing that matters, any software architecture will do!

It’s the requirements that are above and beyond functionality that require us to structure an architecture. They include:

- design constraints
- quality attributes
Requirements Elicitation

Eliciting
functional requirements and design constraints

Yet these are critical to architectural design!

Good Job

Eliciting
quality attribute requirements

Bad Job
Difficulties in Eliciting Quality Attribute Requirements

Non-Operational requirements

- “The system must be easy to use.”
- “The system must have high performance.”
- “The system must be portable.”

Debating the quality attribute to which a system behavior belongs

- “The system must process 10,000 messages per second.”

Vocabulary variations

- Everyone knows what “high performance” means, right?
Today’s Topics

• What is Software Architecture?
• Why is Software Architecture Important?
• Which Requirements Are Most Important To Architectural Design?
• What Else Influences Software Architecture?
Other Influences on the Architecture

Software architecture is influenced by the technical, business, and social environment.
Examples of Other Influences

Stakeholders

• customers, users, managers, marketing, developers, maintainers, etc.

Development organization

• immediate and long term business goals
• organizational structure

Technical environment

• object oriented, WWW, intelligent agents, EJB, service oriented, J2EE, thin client, .NET, etc.

Background and experience

• architect and organizational experience
• education and training
An architecture can influence the technical, business, and social environment.
Examples of What Architecture Can Influence

Development organization
  • structure, goals, artifacts, etc.

Stakeholder requirements
  • demand for similar features, existing components and system

Technical environment
  • relational databases, WWW, service oriented architectures, etc.

Background and experience
  • promote approaches that have been successful
  • reject approaches that have failed
Understanding These Influences

Understanding this cycle of influences helps us to plan for and manage change throughout the lifetime of a system.
Conclusion

Software architecture is important!

Every software system has an architecture!

Quality attribute requirements are critical!

Requirements aren’t the only things that influence software architectures!
For More Information


Other information is provided at http://www.sei.cmu.edu/architecture/.

Contact
Rob Wojcik
Software Engineering Institute
rwojcik@sei.cmu.edu
NO WARRANTY

THIS MATERIAL OF CARNEGIE MELLON UNIVERSITY AND ITS SOFTWARE ENGINEERING INSTITUTE 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.
As projects continue to grow in scale and complexity, effective collaboration across geographical, cultural, and technical boundaries is increasingly prevalent and essential to system success. SATURN 2012 will explore the theme of “Architecture: Catalyst for Collaboration.”