L01 Architecture requirements and design

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Harvinder S Jabbal
CSIS, Work Integrated Learning Programs

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

SEZG651/ SSZG653 Software Architectures Module 3-CS 04

Requirements, Designing, Documentation

innovate achieve lead

- Architecture and Requirements
- Designing an Architecture
- Documenting Software Architecture

Architecture and Requirements

innovate achieve lead

- Gathering ASRs from Requirements Documents
- Gathering ASRs by Interviewing Stakeholders
- Gathering ASRs by Understanding the Business Goals
- Capturing ASRs in a Utility Tree
- Tying the Methods Together

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Designing an Architecture

Designing an Architecture

- Design Strategy
- The Attribute-Driven Design Method
- The Steps of ADD

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Documenting Software Architecture

Documenting Software Architecture

innovate achieve lead

- Uses and Audiences for Architecture Documentation
- Notations for Architecture Documentation
- Views
- Choosing the Views
- Combining Views
- Building the Documentation Package
- Documenting Behavior
- Architecture Documentation and Quality Attributes
- Documenting Architectures That Change Faster Than You Can Document Them
- Documenting Architecture in an Agile Development Project

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Architectures in Agile Projects

Chapter Outline

How Much Architecture?

Agility and Architecture Methods

A Brief Example of Agile Architecting

Guidelines for the Agile Architect

Summary

innovate achieve lead

-- Charles Darwin

It is not the strongest of the species that survives,

not the most intelligent that survives. It is the

one that is the most adaptable to change.

Architecture in Agile Environment

- How Much Architecture?
- Agility and Architecture Methods
- A Brief Example of Agile Architecting
- Guidelines for the Agile Architect

Agility

Agile processes were a response to a need for projects to

- be more responsive to their stakeholders
- be quicker to develop functionality that users care about
- show more and earlier progress in a project's life cycle
- be less burdened by documenting aspects of a project that would inevitably change.

These needs are not in conflict with architecture.

Agility

The question for a software project is not "Should I do Agile or architecture?" Instead ask:

- "How much architecture should I do up front versus how much should I defer until the project's requirements have solidified somewhat?"
- "How much of the architecture should I formally document, and when?"

Agile and architecture are happy companions for many software projects.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions	• over processes and tools
Working software	• over comprehensive documentation
Customer collaboration	• over contract negotiation
Responding to change	• over following a plan

That is, while there is value in the items on the right,

• we value the items on the left more.

Twelve Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile

- These processes were initially employed on small- to medium-sized projects with short time frames and enjoyed considerable success.
- They were not often used for larger projects, particularly those with distributed development.

How Much Architecture?

There are two activities that can add time to the project schedule:

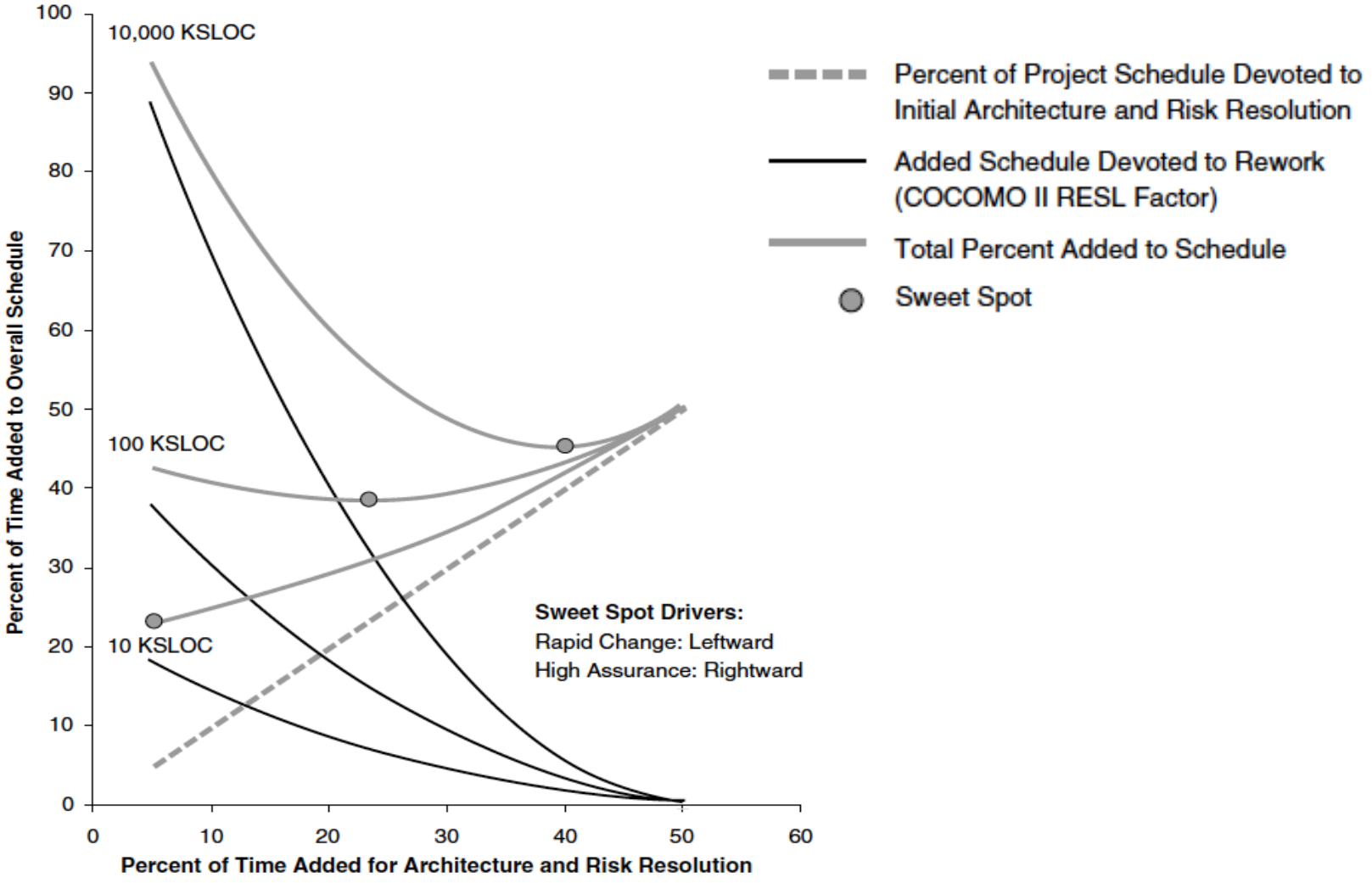
- Up-front design work on the architecture and up-front risk identification, planning, and resolution work
- Rework due to fixing defects and addressing modification requests.

Intuitively, these two trade off against each other.

Boehm and Turner plotted these two values against each other for three hypothetical projects:

- One project of 10 KSLOC
- One project of 100 KSLOC
- One project of 1,000 KSLOC

How Much Architecture?



How Much Architecture?

- These lines show that there is a sweet spot for each project.
  - For the 10KSLOC project, the sweet spot is at the far left. Devoting much time to up-front work is a waste for a small project.
  - For the 100 KSLOC project, the sweet spot is around 20 percent of the project schedule.
  - For the 1,000 KSLOC project, the sweet spot is around 40 percent of the project schedule.
- A project with a million lines of code is enormously complex.
- It is difficult to imagine how Agile principles alone can cope with this complexity if there is no architecture to guide and organize the effort.

Agility and Documentation

Write for the reader!

If the reader doesn't need it, don't write it.

- But remember that the reader may be a maintainer or other newcomer not yet on the project!

Agility and Architecture Evaluation

- Architecture evaluation work as part of an Agile process? Absolutely.
- Meeting stakeholders' important concerns is a cornerstone of Agile philosophy.
- Our approach to architecture evaluation is exemplified by the Architecture Tradeoff Analysis Method (ATAM).
  - ATAM does not endeavor to analyze all, or even most, of an architecture.
  - The focus is determined by a set of quality attribute scenarios that represent the most important of the concerns of the stakeholders.
- It is easy to tailor a lightweight architecture evaluation.

An Example of Agile Architecting

WebArrow Web-conferencing system

To meet stakeholder needs, architect and developers found that they needed to think and work in two different modes at the same time:

- *Top-down*—designing and analyzing architectural structures to meet the demanding quality attribute requirements and tradeoffs
- *Bottom-up*—analyzing a wide array of implementation-specific and environment-specific constraints and fashioning solutions to them

Experiments to Make Tradeoffs

- To analyze architectural tradeoffs, the team adopted an agile architecture discipline combined with a rigorous program of experiments.
  - These experiments are called “spikes” in Agile terminology.

Experiments to Make Tradeoffs

The experiments (spikes) answered questions such as:

- Would moving to a distributed database from local flat files negatively impact feedback time (latency) for users?
- What (if any) scalability improvement would result from using mod_perl versus standard Perl?
- How difficult would the development and quality assurance effort be to convert to mod_perl?
- How many participants could be hosted by a single meeting server?
- What was the correct ratio between database servers and meeting servers?

Lesson

- Making architecture processes agile does not require radical re-invention of either Agile practices or architecture methods.
- The WebArrow team's emphasis on experimentation proved the key factor.

Guidelines for the Agile Architect

Barry Boehm and colleagues have developed the Incremental Commitment Model to blend agility and architecture.

This model is based upon the following principles:

- Commitment and accountability of success-critical stakeholders
- Stakeholder "satisficing" (meeting an acceptability threshold) based on success-based negotiations and tradeoffs
- Incremental and evolutionary growth of system definition and stakeholder commitment
- Iterative system development and definition
- Interleaved system definition and development allowing early fielding of core capabilities, continual adaptation to change, and timely growth of complex systems without waiting for every requirement and subsystem to be defined
- Risk management—risk-driven anchor point milestones, which are key to synchronizing and stabilizing all of this concurrent activity

Authors' Advice

- If you are building a large, complex system with relatively stable and well-understood requirements and/or distributed development, doing a large amount of architecture work up front will pay off.
- On larger projects with *unstable* requirements, start by quickly designing a candidate architecture even if it leaves out many details.
  - Be prepared to change and elaborate this architecture as circumstances dictate, as you perform your spikes and experiments, and as functional and quality attribute requirements emerge and solidify.
- On smaller projects with uncertain requirements, at least try to get agreement on the major patterns to be employed. Don't spend too much time on architecture design, documentation, or analysis up front.

Summary

- The Agile Manifesto and principles value close-knit teams, with continuous, frequent delivery of working software.
- Agile processes were initially employed on small- to medium-sized projects with short time frames. They were seldom used for larger projects, particularly with distributed development.
- Large-scale successful projects need a blend of agile and architecture.
- Agile architects take a middle ground, proposing an initial architecture and running with that, until its technical debt becomes too great, at which point they need to refactor.
- Boehm and Turner found that projects have a “sweet spot” where up-front architecture planning pays off.