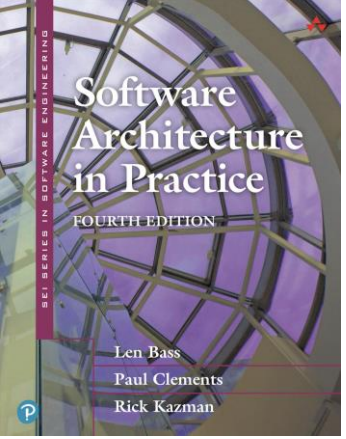
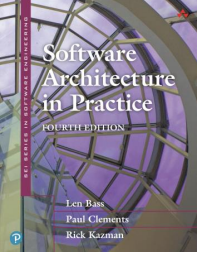


Chapter 24: The Role of Architects in Projects

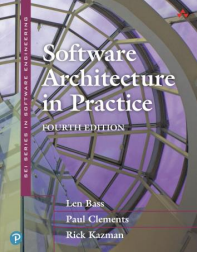
I don't know why people hire architects and then tell them what to do.

—Frank Gehry



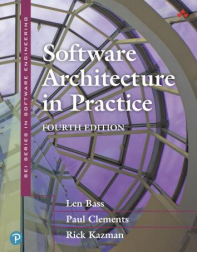
Chapter Outline

- The Architect and the Project Manager
- Incremental Architecture and Stakeholders
- Architecture and Agile Development
- Architecture and Distributed Development
- Summary



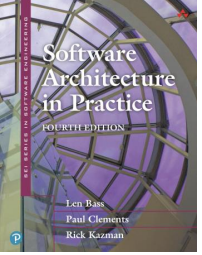
The Architect and the Project Manager

- The relationship between the software architect and the project manager (PM) is crucial.
- The PM is responsible for the overall performance of the project—keeping it on budget, on schedule, and properly staffed.
- To do this, the PM will often turn to the architect for support.
- The PM is responsible for the external-facing aspects of the project and the architect is responsible for the internal technical aspects.



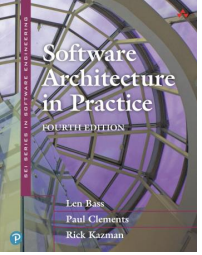
Architect's Role in Supporting PM Knowledge Areas

PMBOK Knowledge Area	Description	Software Architect Role
Project Integration Management	Ensuring that the various elements of the project are properly coordinated	Create design and organize team around design; manage dependencies. Implement the capture of metrics. Orchestrate requests for changes.
Project Scope Management	Ensuring that the project includes all of the work required and only the work required	Elicit, negotiate, and review runtime requirements and generate development requirements. Estimate cost, schedule, and risk associated with meeting requirements.
Project Time Management	Ensuring that the project completes in a timely fashion	Help define the work breakdown structure. Define tracking measures. Recommend assignment of resources to software development teams.
Project Cost Management	Ensuring that the project is completed within the required budget	Gather costs from individual teams; make recommendations regarding build/buy and resource allocations.



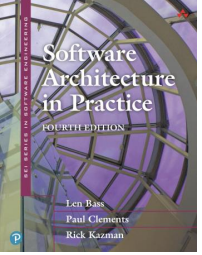
Architect's Role in Supporting PM Knowledge Areas

Project Quality Management	Ensuring that the project will satisfy the needs for which it was undertaken	Design for quality and track the system against the design. Define quality metrics.
Project Human Resource Management	Ensuring that the project makes the most effective use of the people involved with the project	Define the required technical skill sets. Mentor developers about career paths. Recommend training. Interview candidates.
Project Communications Management	Ensuring timely and appropriate generation, collection, dissemination, storage, and disposition of project information	Ensure communication and coordination among developers. Solicit feedback as to progress, problems, and risks. Oversee documentation.
Project Risk Management	Identifying, analyzing, and responding to project risk	Identify and quantify risks; adjust the architecture and processes to mitigate risk.
Project Procurement Management	Acquiring goods and services from outside the organization	Determine technology requirements; recommend technology, training, and tools.



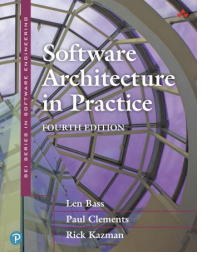
Recommendations to the Architect

- Maintain a good working relationship with the PM.
- Be aware of the PM's tasks and concerns, and how you as an architect may be asked to support those tasks and concerns.



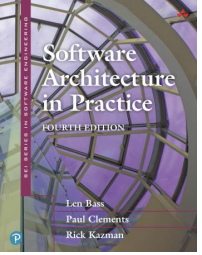
Incremental Architecture and Stakeholders

- Agile methodologies are built on the pillar of incremental development, with each increment delivering value to the customer or user.
- You should expect to develop and release your architecture in increments.
- This entails deciding which views to release (out of your planned set) and at which depth.



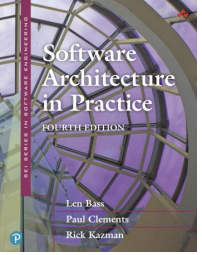
Incremental Architecture and Stakeholders

- Consider these as candidates for your first increment:
  - A module decomposition structure. This will inform the team structure for the project which must be defined, staffed, budgeted, and trained. The team structure will be the basis of project planning and budgeting.
  - A module “uses” structure. This will allow increments to be planned, which is critical in any project that hopes to release its software incrementally. Trying to create a system that purposefully supports incremental development is problematic if you don’t plan what exactly the increments will be.
  - Whichever component-and-connector (C&C) structure(s) best convey the overall solution approach.
  - A broad-brush deployment structure that at least addresses major questions such as whether the system will be deployed on mobile devices, on a cloud infrastructure, and so forth.



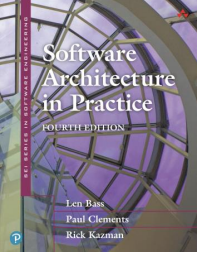
Recommendations to the Architect

- Know who your stakeholders are and what their needs are, so that you can design appropriate solutions and documentation.
- Work with stakeholders to determine the release tempo and the contents of each project increment.
- Your first increment should include module decomposition and uses views, as well as a preliminary C&C view.
- Use your influence to ensure that early releases deal with the system's most challenging QA requirements, thereby ensuring no unpleasant architectural surprises later.
- Stage your architecture releases to support those increments and the needs of developers as they work on each increment.



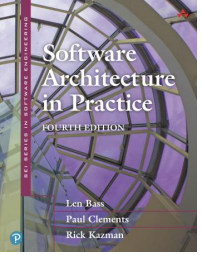
Architecture and Agile Development

- Agile development began as a rebellion against development approaches that were rigid and heavyweight with lots of documentation, focused on up-front planning, culminating in a single delivery.
- Agilistas advocate focusing on what the customer really wants and providing it in small, testable increments, starting early on.
- The key question is this: How much up-front work, in terms of requirements analysis, risk mitigation, and architecture design, should a project undertake?

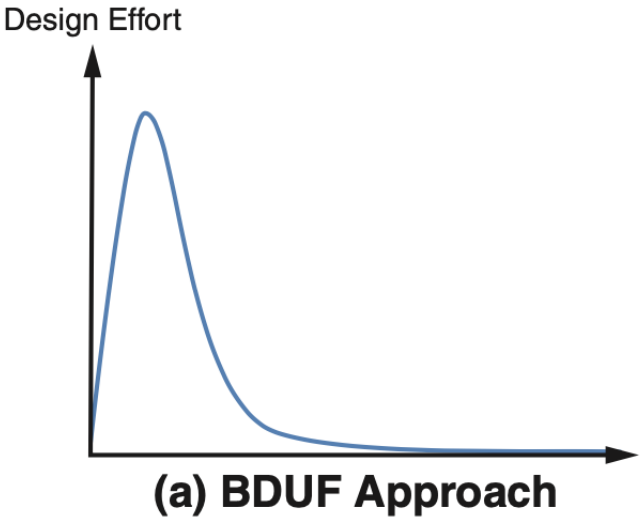


Architecture and Agile Development

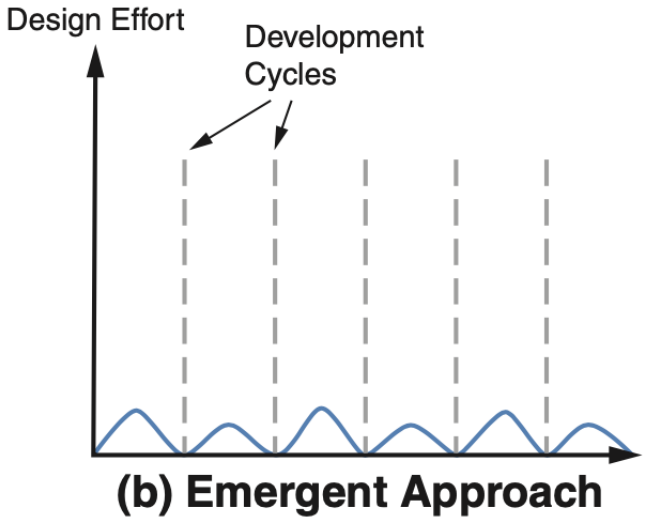
- There is no single right answer to this question, but you can find a “sweet spot” for any given project.
- The “right” amount of project work depends on several factors: project size, complex functional requirements, highly demanding QA requirements, volatile requirements, degree of distribution of development.



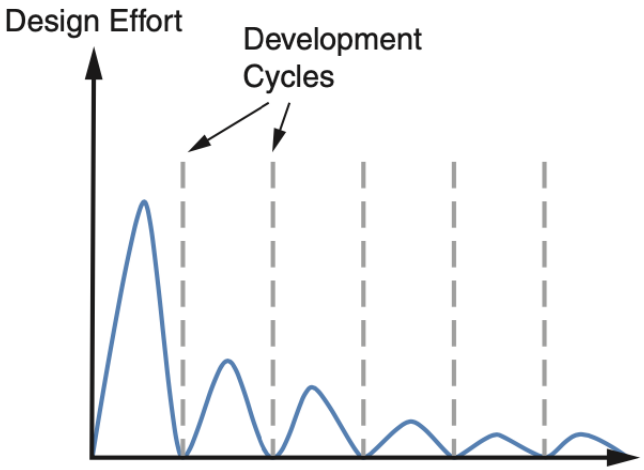
Three Approaches to Architectural Design



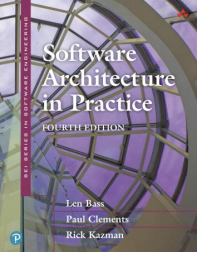
(a) BDUF Approach



(b) Emergent Approach

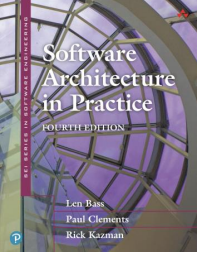


(c) Iteration 0 Approach



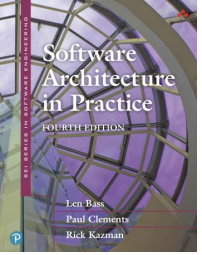
Three Approaches to Architectural Design

- We recommend the “Iteration 0” approach.
- If you have some understanding of the requirements, perform a few iterations of ADD (Chapter 20). Focus on choosing the major architectural patterns (including a reference architecture, if appropriate), frameworks, and components.
- This will help you structure the project, define work assignments and team formation, and address the most critical QAs.
- When requirements change—particularly driving QA requirements—adopt a practice of Agile experimentation with "spikes".



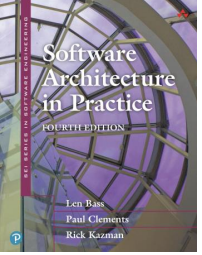
Three Approaches to Architectural Design

- Some Agile doctrine declares that if you aren't delivering *working software*, then you aren't doing anything of value.
- Therefore if you're working on an architecture, then you're not programming and, therefore, you're doing *nothing of value*! Write code, and the architecture will emerge organically.
- For medium to large systems, this view has inevitably collapsed under the harsh weight of experience.



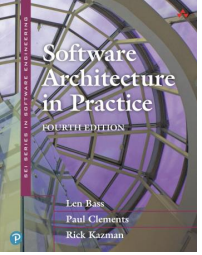
Agile Principles and Architecture-centric Perspective

Agile Principle	Architecture-centric View
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Absolutely.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Absolutely. This principle is served by architectures that provide high degrees of modifiability (Chapter 8) and deployability (Chapter 5).
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter time scale.	Absolutely, as long as this principle is not seen as precluding a thoughtful architecture. DevOps has a large role to play here, and we have seen, in Chapter 5, how architectures can support DevOps.
Business people and developers must work together daily throughout the project.	Business goals lead to quality attribute requirements, which the architecture's primary duty is to fulfill, as we discussed in Chapter 19.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	While we agree in principle, many developers are inexperienced. So make sure to include a skilled, experienced, and motivated architect to help guide these individuals.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	This is nonsense for nontrivial systems. Humans invented writing because our brains can't remember everything we need to remember. Interfaces, protocols, architectural structures, and more need to be written down, and the inefficiencies and ineffectiveness of repeated instruction and resulting errors from misunderstanding belie this principle. According to this argument, nobody should produce user manuals, but should just publish the developers' phone numbers with an open invitation to call them anytime. This is also nonsense for any system that has a maintenance phase (that's pretty much every system) in which the original team is nowhere to be found. With whom are you going to have that face-to-face conversation to learn important details? See Chapter 22 for our guidance in this matter.



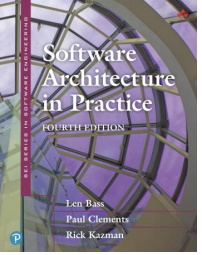
Agile Principles and Architecture-centric Perspective

Working software is the primary measure of progress.	Yes, as long as "primary" is not taken to mean "only," and as long as this principle is not used as an excuse to eliminate all work except coding.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Absolutely.
Continuous attention to technical excellence and good design enhances agility.	Absolutely.
Simplicity—the art of maximizing the amount of work not done—is essential.	Yes, of course, as long as it is understood that the work we are not doing can actually be jettisoned safely without detriment to the system being delivered.
The best architectures, requirements, and designs emerge from self-organizing teams.	No, they don't. The best architectures are consciously designed by skilled, talented, trained, and experienced architects, as we describe in Chapter 20
At regular intervals, the team reflects on how to become more effective, and then tunes and adjusts its behavior accordingly.	Absolutely.



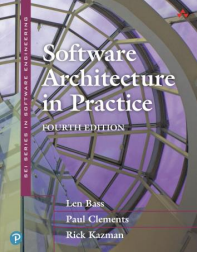
Agile Architecture and SAFe

- One approach to apply Agile at enterprise scale is the Scaled Agile Framework (SAFe).
- SAFe provides a set of workflows, roles, and processes to coordinate the activities of many teams, each operating in classic Agile fashion.
- SAFe acknowledges the role of architecture. It admits “intentional architecture”: a set of purposeful, planned architectural strategies and initiatives.
- And it includes a counterbalancing force called “emergent design”.



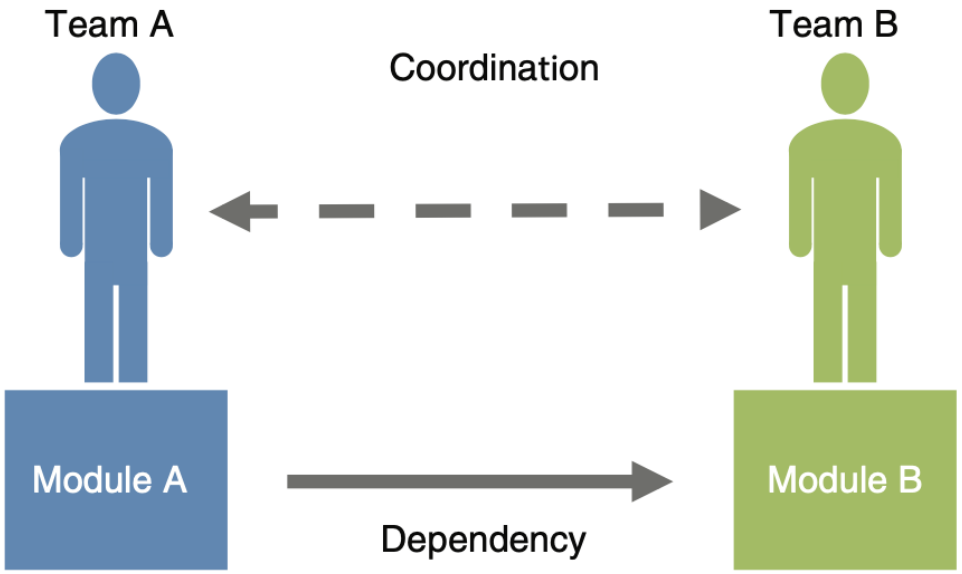
Architecture and Distributed Development

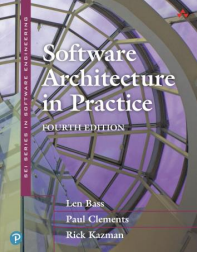
- Many projects today are developed by distributed teams, possibly outsourced, possibly scattered around the globe.
- Distributed development comes with both benefits and challenges, such as lowering costs, leveraging specialized skills and local knowledge of markets.



Architecture and Distributed Development

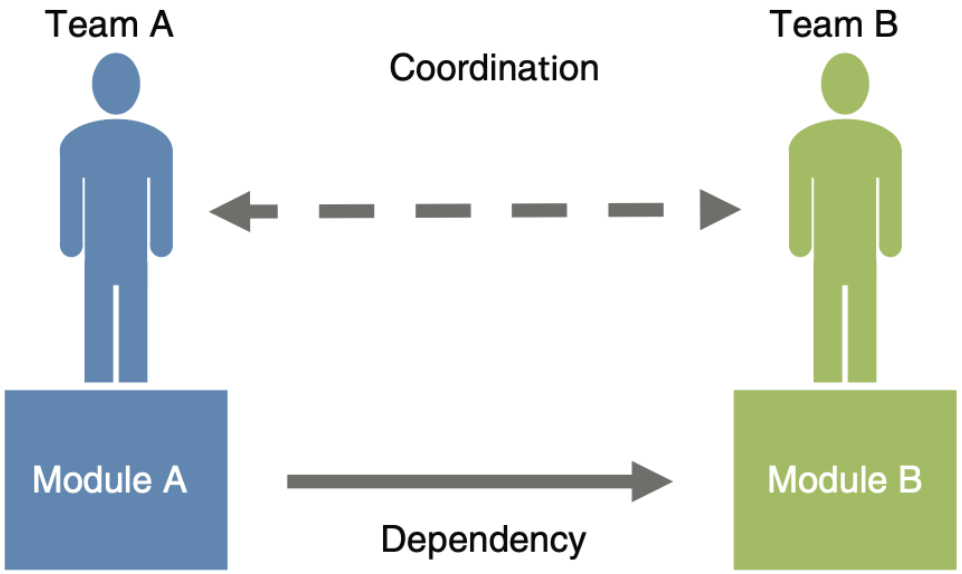
- How does distributed development play out on a project?
- Assume Module A uses an interface from Module B. In time this interface may need to be modified.
- The team responsible for Module B must coordinate with the team responsible for Module A.

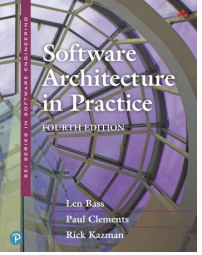




Architecture and Distributed Development

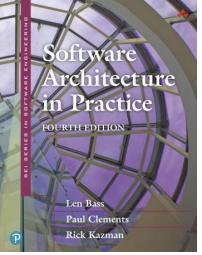
- Methods for coordination include :
  - *Informal contacts*.
  - *Documentation*.
  - *Meetings*.
  - *Asynchronous electronic communication*.





Architecture and Distributed Development

- What does this mean for architecture and the architect?
  - Allocation of responsibilities to teams is more important in distributed development than in co-located development.
  - Attention to module dependencies takes on added importance: Dependencies among modules owned by distributed teams are more likely to be problematic and should be minimized.
  - Documentation is especially important in distributed development.



Summary

- Architects do their work in the context of a development project; they need to understand their roles and responsibilities from that perspective.
- The project manager and the architect have complementary roles: The manager runs the project from an administrative perspective, the architect runs it from a technical perspective. These roles intersect.
- Architectures are released in increments that are useful to stakeholders. Thus the architect needs to have a good understanding of the architecture's stakeholders and their information needs.
- Agile methodologies focus on incremental development. Over time, architecture and Agile have become indispensable partners.
- Global development creates a need for an explicit coordination strategy that is based on more formal strategies than are needed for co-located development.