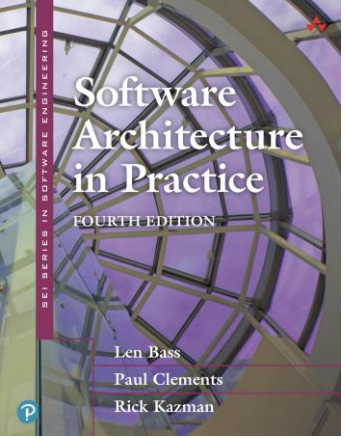
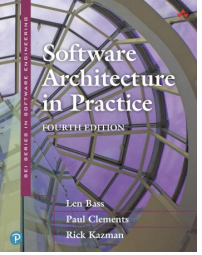


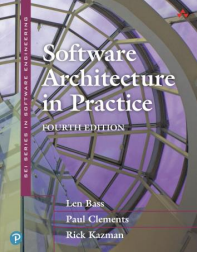
Chapter 19: Architecturally Significant Requirements

The most important single aspect of software development is to be clear about what you are trying to build.
—Bjarne Stroustrup



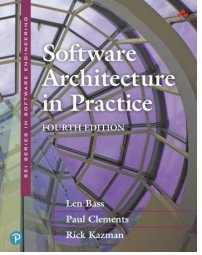
Chapter Outline

- Gathering ASRs from Requirements Documents
- Gathering ASRs by Interviewing Stakeholders
- Gathering ASRs by Understanding the Business Goals
- Capturing ASRs in a Utility Tree
- Change Happens
- Summary



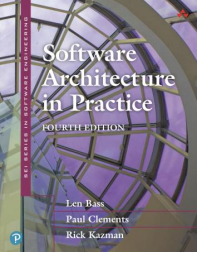
Gathering ASRs

- Architectures exist to build systems that satisfy requirements.
- Not all requirements are created equal.
- Some have a profound effect on the architecture than others: *architecturally significant requirements (ASRs)*.
- You cannot hope to design a successful architecture if you do not know the ASRs.
- How do we do this?



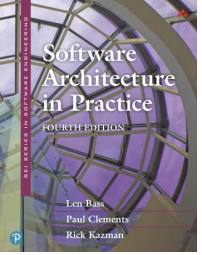
Gathering ASRs from Requirements Documents

- An obvious location to look for candidate ASRs is in the requirements document or in user stories.
- Don't get your hopes up.



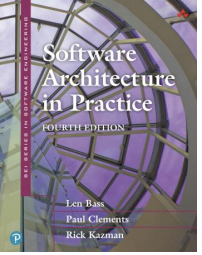
Gathering ASRs from Requirements Documents

- Many projects don't have good (or any) requirements document.
- And no architect can wait until the requirements are "done" before starting work.
- In practice, though, we rarely see adequate capture of QA requirements.
- Much of what is useful to an architect won't be found in even the best requirements document.



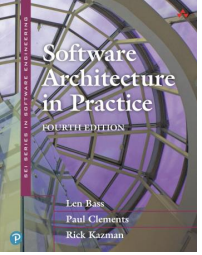
Gathering ASRs from Requirements Documents

- Some things to look for:
  - *Usage*. User roles versus system modes, internationalization, language distinctions.
  - *Time*. Timeliness and element coordination.
  - *External elements*. External systems, protocols, sensors or actuators (devices),
  - middleware.
  - *Networking*. Network properties and configurations (including their security properties).
  - *Orchestration*. Processing steps, information flows.
  - *Security properties*. User roles, permissions, authentication.
  - *Data*. Persistence and currency.
  - *Resources*. Time, concurrency, memory footprint, scheduling, multiple users, multiple activities, devices, energy usage, soft resources (e.g., buffers, queues), scalability requirements.
  - *Project management*. Plans for teaming, skill sets, training, team coordination.
  - *Hardware choices*. Processors, families of processors, evolution of processors.
  - *Flexibility of functionality, portability, calibrations, configurations.*
  - *Named technologies, commercial packages.*



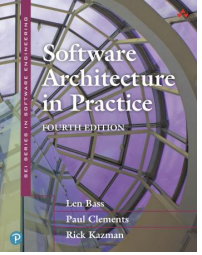
Gathering ASRs by Interviewing Stakeholders

- Stakeholders often don't know what their QA requirements actually are. In that case, architects may have to set the QA requirements.
- If you insist on quantitative QA requirements, you may get numbers that are arbitrary.
- Experienced architects often have deep insights based on similar systems.
- Architects can also give feedback as to which QA responses will be straightforward to achieve and which will likely be problematic.



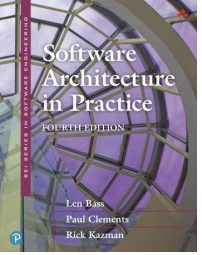
Gathering ASRs by Interviewing Stakeholders

- Interviewing the relevant stakeholders is the surest way to learn what they know and need.
- The results of stakeholder interviews should include a prioritized list of architectural drivers including QA scenarios.
- One such method to do this is the Quality Attribute Workshop (QAW).
- The QAW is a facilitated, stakeholder-focused method to generate, prioritize, and refine quality attribute scenarios before the software architecture is completed.



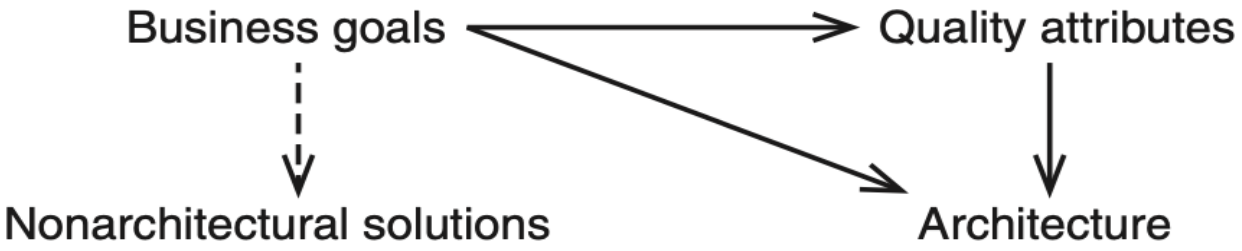
Steps of the QAW

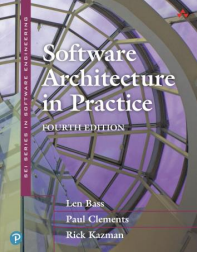
- *Business/mission presentation.*
- *Architectural plan presentation.*
- *Identification of architectural drivers.*
- *Scenario brainstorming.*
- *Scenario consolidation.*
- *Scenario prioritization.*
- *Scenario refinement.*



Gathering ASRs by Understanding Business Goals

- Business goals are of interest to architects because they frequently lead directly to ASRs.
- There are three possible relationships between business goals and an architecture:
  - *Business goals often lead to quality attribute requirements.*
  - *Business goals may affect the architecture without inducing a quality attribute requirement at all.*
  - *No influence of a business goal on the architecture.*

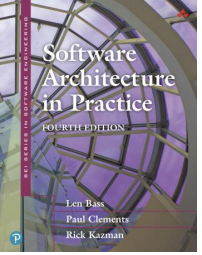




Gathering ASRs by Understanding Business Goals

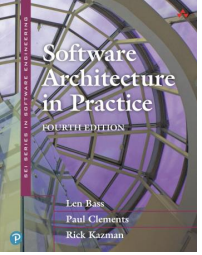
- Categories of business goals:
  - Growth and continuity of the organization
  - Meeting financial objectives
  - Meeting personal objectives
  - Meeting responsibility to the employees
  - Meeting responsibility to society
  - Meeting responsibility to the state
  - Meeting responsibility to the shareholders
  - Managing market position
  - Improving business processes
  - Managing the quality and reputation of products
  - Managing change in the environment over time

- These categories can be used as an aid to brainstorming and elicitation.



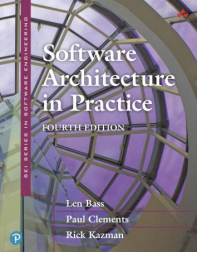
Capturing ASRs in a Utility Tree

- Architects can use a construct called a *utility tree* when the "primary sources" of requirements are not available.
- A utility tree is a top-down representation of what you, as architect, believe to be the most important QA-related ASRs.
- The leaves of the tree are scenarios that you prioritize along two dimensions: business value and technical risk. Each dimension is scored H/M/L.



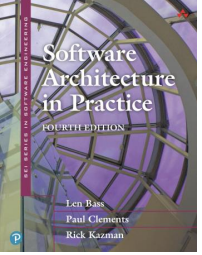
Example Utility Tree

Utility	Performance	Transaction response time	User updates a patient's account in response to a change notification while system is at peak load. The transaction completes in < 0.75 sec.	(H,M)
			User updates a patient's account in response to a change notification while system is at double peak load. Transaction completes in < 4 sec.	(L,M)
		Throughput	At peak load, the system can complete 150 transactions/sec.	(M,M)
	Usability	Proficiency training	A new hire with two+ years' experience in the business becomes proficient in Nightingale's core functions in <1 week .	(M,L)
			A user asks for help, and the system provides help for their context, within 3 secs.	(H,M)
		Normal operations	A payment officer initiates a payment plan for a patient and completes the process without the system introducing delays.	(M,M)
	Configurability	User-defined changes	A hospital increases the fee for a service. The configuration team makes the change in 1 working day; no source code is changed.	(H,L)
	Maintainability	Routine changes	A maintainer encounters search- and response-time deficiencies, fixes the bug, and distributes the fix in < 3 person-days of effort.	(H,M)
			A reporting requirement requires a change to the report-generating metadata. Change is made in 4 person-hours of effort.	(M,L)
		COTS upgrades	The database vendor releases a new version that must be installed in < 3 person-weeks.	(H,M)



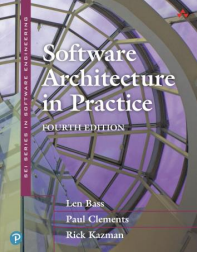
Capturing ASRs in a Utility Tree

- Once you have a utility tree filled out, you can use it to make important checks. For instance:
  - A QA without any ASR scenario is an area you should investigate.
  - ASR scenarios that receive a (H, H) rating are obviously the ones that deserve the most attention from you.



Change Happens

- Requirements—whether captured or not—change all the time.
- Architects have to adapt and keep up, to ensure that their architectures are still the right ones that will bring success to the project.
- Always keep a channel open to the key stakeholders who determine the ASRs so you can keep up with changing requirements.



Summary

- Architectures are driven by architecturally significant requirements. An ASR must have:
  - A profound impact on the architecture.
  - A high business or mission value.
- ASRs can be extracted from a requirements document, captured from stakeholders during a workshop (e.g., a QAW), captured from the architect in a utility tree, or derived from business goals.