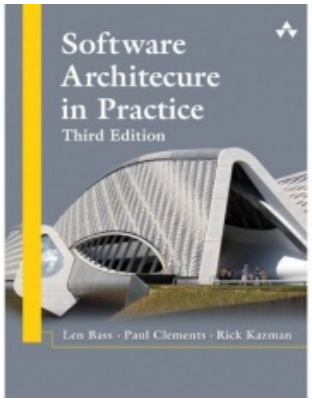
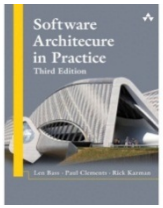
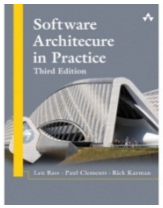


Chapter 16:
Architecture and Requirements



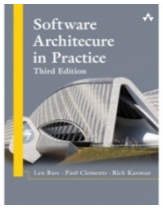
Chapter Outline

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



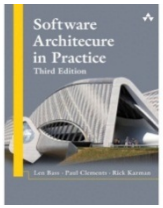
Requirements

- Architectures exist to build systems that satisfy requirements.
- But, to an architect, not all requirements are created equal.
- An *architecturally significant requirement* (ASR) is a requirement that will have a profound effect on the architecture.
- How do we find those?



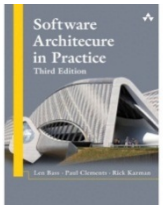
ASRs and Requirements Documents

- An obvious location to look for candidate ASRs is in the requirements documents or in user stories.
- Requirements should be in requirements documents!
- Unfortunately, this is not usually the case.
- Why?



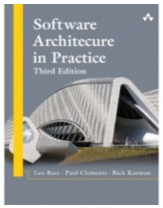
Don’t Get Your Hopes Up

- Many projects don’t create or maintain the detailed, high-quality requirements documents.
- Standard requirements pay more attention to functionality than quality attributes.
- Most of what is in a requirements specification does not affect the architecture.
- No architect just sits and waits until the requirements are “finished” before starting work. The architect *must* begin while the requirements are still in flux.



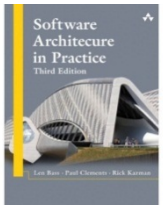
Don’t Get Your Hopes Up

- Quality attributes, when captured at all, are often captured poorly.
  - “The system shall be modular”
  - “The system shall exhibit high usability”
  - “The system shall meet users’ performance expectations”
- Much of what is useful to an architect is not in even the best requirements document.
  - ASRs often derive from business goals in the development organization itself
  - Developmental qualities (such as teaming) are also out of scope



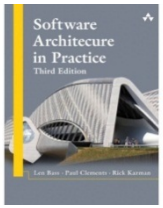
Sniffing Out ASRs

Design Decision Category	Look for Requirements Addressing . . .
Allocation of Responsibilities	Planned evolution of responsibilities, user roles, system modes, major processing steps, commercial packages
Coordination Model	Properties of the coordination (timeliness, currency, completeness, correctness, and consistency)
	Names of external elements, protocols, sensors or actuators (devices), middleware, network configurations (including their security properties)
	Evolution requirements on the list above
Data Model	Processing steps, information flows, major domain entities, access rights, persistence, evolution requirements
Management of Resources	Time, concurrency, memory footprint, scheduling, multiple users, multiple activities, devices, energy usage, soft resources (buffers, queues, etc.)
	Scalability requirements on the list above
Mapping among Architectural Elements	Plans for teaming, processors, families of processors, evolution of processors, network configurations
Binding Time Decisions	Extension of or flexibility of functionality, regional distinctions, language distinctions, portability, calibrations, configurations
Choice of Technology	Named technologies, changes to technologies (planned and unplanned)



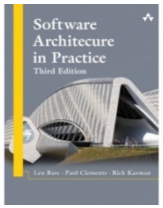
Gathering ASRs from Stakeholders

- Say your project won't have the QAs nailed down by the time you need to start your design work.
- What do you do?
- Stakeholders often have *no idea* what QAs they want in a system
  - if you insist on quantitative QA requirements, you're likely to get numbers that are arbitrary.
  - at least some of those requirements will be very difficult to satisfy.
- Architects often have very good ideas about what QAs are reasonable to provide.
- Interviewing the relevant stakeholders is the surest way to learn what they know and need.



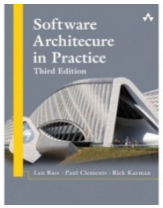
Gathering ASRs from Stakeholders

- The results of stakeholder interviews should include
  - a list of architectural drivers
  - a set of QA scenarios that the stakeholders (as a group) prioritized.
- This information can be used to:
  - refine system and software requirements
  - understand and clarify the system's architectural drivers
  - provide rationale for why the architect subsequently made certain design decisions
  - guide the development of prototypes and simulations
  - influence the order in which the architecture is developed.



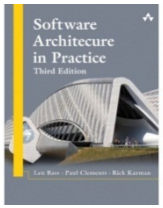
Quality Attribute Workshop

- The QAW is a facilitated, stakeholder-focused method to generate, prioritize, and refine quality attribute scenarios before the software architecture is completed.
- The QAW is focused on system-level concerns and specifically the role that software will play in the system.



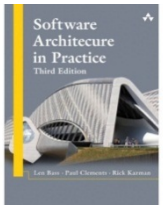
QAW Steps

- **Step 1: QAW Presentation and Introductions.**
  - QAW facilitators describe the motivation for the QAW and explain each step of the method.
- **Step 2: Business/Mission Presentation.**
  - The stakeholder representing the business concerns behind the system presents the system's business context, broad functional requirements, constraints, and known quality attribute requirements.
  - The quality attributes that will be refined in later steps will be derived largely from the business/mission needs presented in this step.
- **Step 3: Architectural Plan Presentation.**
  - The architect will present the system architectural plans as they stand.
  - This lets stakeholders know the current architectural thinking, to the extent that it exists.
- **Step 4: Identification of Architectural Drivers.**
  - The facilitators will share their list of key architectural drivers that they assembled during Steps 2 and 3, and ask the stakeholders for clarifications, additions, deletions, and corrections.
  - The idea is to reach a consensus on a distilled list of architectural drivers that includes overall requirements, business drivers, constraints, and quality attributes.



QAW Steps

- **Step 5: Scenario Brainstorming.**
  - Each stakeholder expresses a scenario representing his or her concerns with respect to the system.
  - Facilitators ensure that each scenario has an explicit stimulus and response.
  - The facilitators ensure that at least one representative scenario exists for each architectural driver listed in Step 4.
- **Step 6: Scenario Consolidation.**
  - Similar scenarios are consolidated where reasonable.
  - Consolidation helps to prevent votes from being spread across several scenarios that are expressing the same concern.
- **Step 7: Scenario Prioritization.**
  - Prioritization of the scenarios is accomplished by allocating each stakeholder a number of votes equal to 30 percent of the total number of scenarios
- **Step 8: Scenario Refinement.**
  - The top scenarios are refined and elaborated.
  - Facilitators help the stakeholders put the scenarios in the six-part scenario form of source-stimulus-artifact-environment-response-response measure.



ASRs from Business Goals

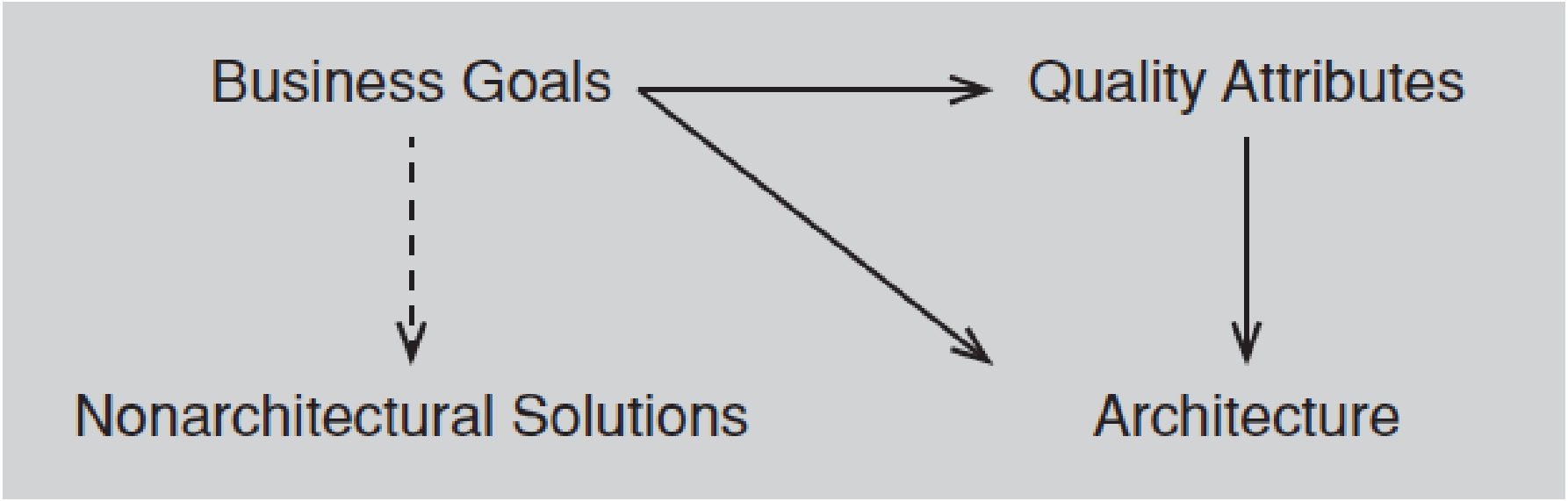
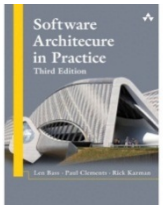
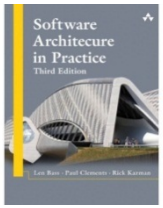


FIGURE 16.1 Some business goals may lead to quality attribute requirements (which lead to architectures), or lead directly to architectural decisions, or lead to nonarchitectural solutions.



Categories of Business Goals, to Aid in Elicitation

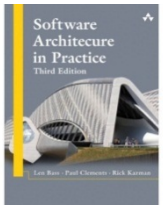
1.	Contributing to the growth and continuity of the organization
2.	Meeting financial objectives
3.	Meeting personal objectives
4.	Meeting responsibility to employees
5.	Meeting responsibility to society
6.	Meeting responsibility to state
7.	Meeting responsibility to shareholders
8.	Managing market position
9.	Improving business processes
10.	Managing the quality and reputation of products
11.	Managing change in environmental factors



Expressing Business Goals

Business goal scenario, 7 parts

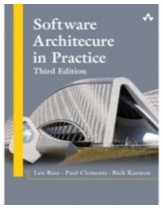
- *Goal-source*
  - The people or written artifacts providing the goal.
- *Goal-subject*
  - The stakeholders who own the goal and wish it to be true.
  - Each stakeholder might be an individual or the organization itself
- *Goal-object*
  - The entities to which the goal applies.
- *Environment*
  - The context for this goal
  - Environment may be social, legal, competitive, customer, and technological



Expressing Business Goals

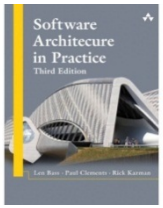
Business goal scenario, 7 parts

- *Goal*
  - Any business goal articulated by the goal-source.
- *Goal-measure*
  - A testable measurement to determine how one would know if the goal has been achieved. The goal-measure should usually include a time component, stating the time by which the goal should be achieved.
- *Pedigree and value*
  - The degree of confidence the person who stated the goal has in it
  - The goal's volatility and value.



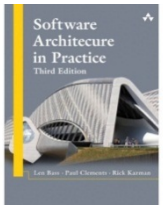
PALM: A Method for Eliciting Business Goals

- PALM is a seven-step method.
- Nominally carried out over a day and a half in a workshop.
- Attended by architect(s) and stakeholders who can speak to the relevant business goals.



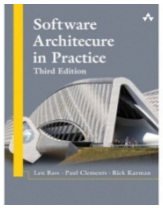
PALM Steps

- *PALM overview presentation*
  - Overview of PALM, the problem it solves, its steps, and its expected outcomes.
- *Business drivers presentation.*
  - Briefing of business drivers by project management
  - What are the goals of the customer organization for this system?
  - What are the goals of the development organization?
- *Architecture drivers presentation*
  - Briefing by the architect on the driving business and quality attribute requirements: the ASRs.
- *Business goals elicitation*
  - Business goals are elaborated and expressed as scenarios.
  - Consolidate almost-alike business goals to eliminate duplication.
  - Participants prioritize the resulting set to identify the most important goals.



PALM Steps

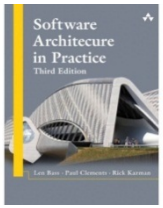
- *Identification of potential quality attributes from business goals.*
  - For each important business goal scenario, participants describe a quality attribute that (if architected into the system) would help achieve it. If the QA is not already a requirement, this is recorded as a finding.
- *Assignment of pedigree to existing quality attribute drivers.*
  - For each architectural driver identify which business goals it is there to support.
  - If none, that's recorded as a finding.
  - Otherwise, we establish its pedigree by asking for the source of the quantitative part.
- *Exercise conclusion*
  - Review of results, next steps, and participant feedback.



Capturing ASRs in a Utility Tree

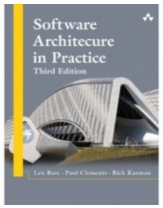
An ASR must have the following characteristics:

- *A profound impact on the architecture*
  - Including this requirement will very likely result in a different architecture than if it were not included.
- *A high business or mission value*
  - If the architecture is going to satisfy this requirement it must be of high value to important stakeholders.



Utility Tree

- A way to record ASRs all in one place.
- Establishes priority of each ASR in terms of
  - Impact on architecture
  - Business or mission value
- ASRs are captured as scenarios.
- Root of tree is placeholder node called “Utility”.
- Second level of tree contains broad QA categories.
- Third level of tree refines those categories.

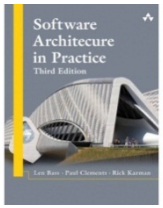


Utility Tree Example (excerpt)

Quality Attribute	Attribute Refinement	ASR
Performance	Transaction response time	A user updates a patient's account in response to a change-of-address notification while the system is under peak load, and the transaction completes in less than 0.75 second. (H,M)
		A user updates a patient's account in response to a change-of-address notification while the system is under double the peak load, and the transaction completes in less than 4 seconds. (L,M)
	Throughput	At peak load, the system is able to complete 150 normalized transactions per second. (M,M)
Usability	Proficiency training	A new hire with two or more years' experience in the business becomes proficient in Nightingale's core functions in less than 1 week. (M,L)
		A user in a particular context asks for help, and the system provides help for that context, within 3 seconds. (H,M)
	Normal operations	A hospital payment officer initiates a payment plan for a patient while interacting with that patient and completes the process without the system introducing delays. (M,M)
Configurability	User-defined changes	A hospital increases the fee for a particular service. The configuration team makes the change in 1 working day; no source code needs to change. (H,L)
Maintainability	Routine changes	A maintainer encounters search- and response-time deficiencies, fixes the bug, and distributes the bug fix with no more than 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)
	Upgrades to commercial components	The database vendor releases a new version that must be…

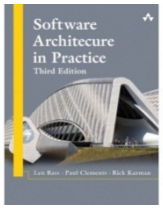
Utility

Key:
H=high
M=medium
L=low



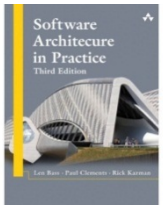
Utility Tree: Next Steps

- A QA or QA refinement without any ASR is not necessarily an error or omission
  - Attention should be paid to searching for unrecorded ASRs in that area.
- ASRs that rate a (H,H) rating are the ones that deserve the most attention
  - A very large number of these might be a cause for concern: Is the system is achievable?
- Stakeholders can review the utility tree to make sure their concerns are addressed.



Tying the Methods Together

- How should you employ requirements documents, stakeholder interviews, Quality Attribute Workshops, PALM, and utility trees in concert with each other?
  - If you have a requirements process that gathers, identifies, and prioritizes ASRs, then use that and consider yourself lucky.
  - Otherwise use one or more of the other approaches.
  - If nobody has captured the business goals behind the system you're building, use PALM.
  - If important stakeholders have been overlooked in the requirements-gathering process, use interviews or a QAW.
  - Build a utility tree to capture ASRs along with their prioritization.
  - You can blend all the methods together
    - Use PALM as a "subroutine call" from a QAW for the business goal step
    - Use a quality attribute utility tree as a repository for the scenarios produced by a QAW.
- Pick the approach that fills in the biggest gap in your existing requirements.



Summary

- Architectures are driven by architecturally significant requirements: requirements that will have profound effects on the architecture.
  - Architecturally significant requirements may be captured from requirements documents, by interviewing stakeholders, or by conducting a Quality Attribute Workshop.
- Be mindful of the business goals of the organization.
  - Business goals can be expressed in a common, structured form and represented as scenarios.
  - Business goals may be elicited and documented using a structured facilitation method called PALM.
- A useful representation of quality attribute requirements is in a utility tree.
  - The utility tree helps to capture these requirements in a structured form.
  - Scenarios are prioritized.
  - This prioritized set defines your “marching orders” as an architect.