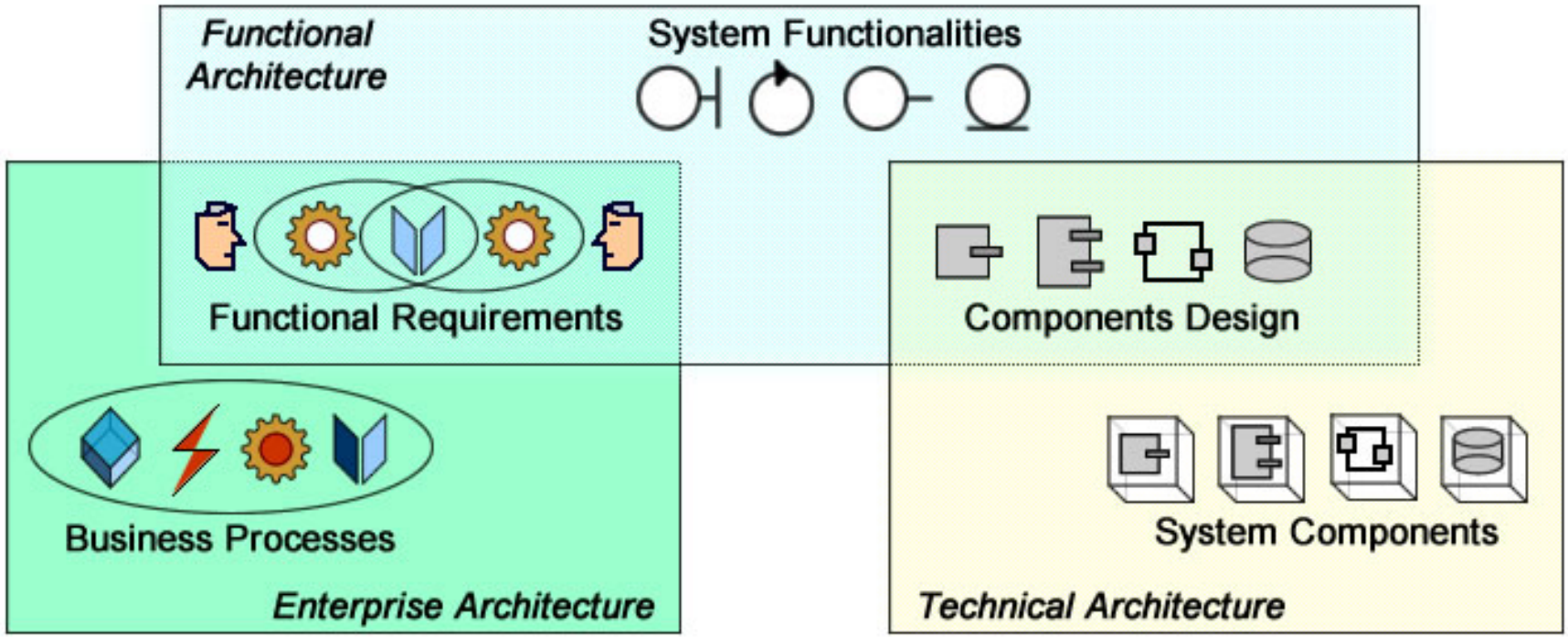
Evaluating
a software architecture

Agenda

- Evaluating a software architecture: why?
- Evaluating a software architecture: how?
- Architectural review: ATAM
- Cost benefits: CBAM

Kinds of “architectures”

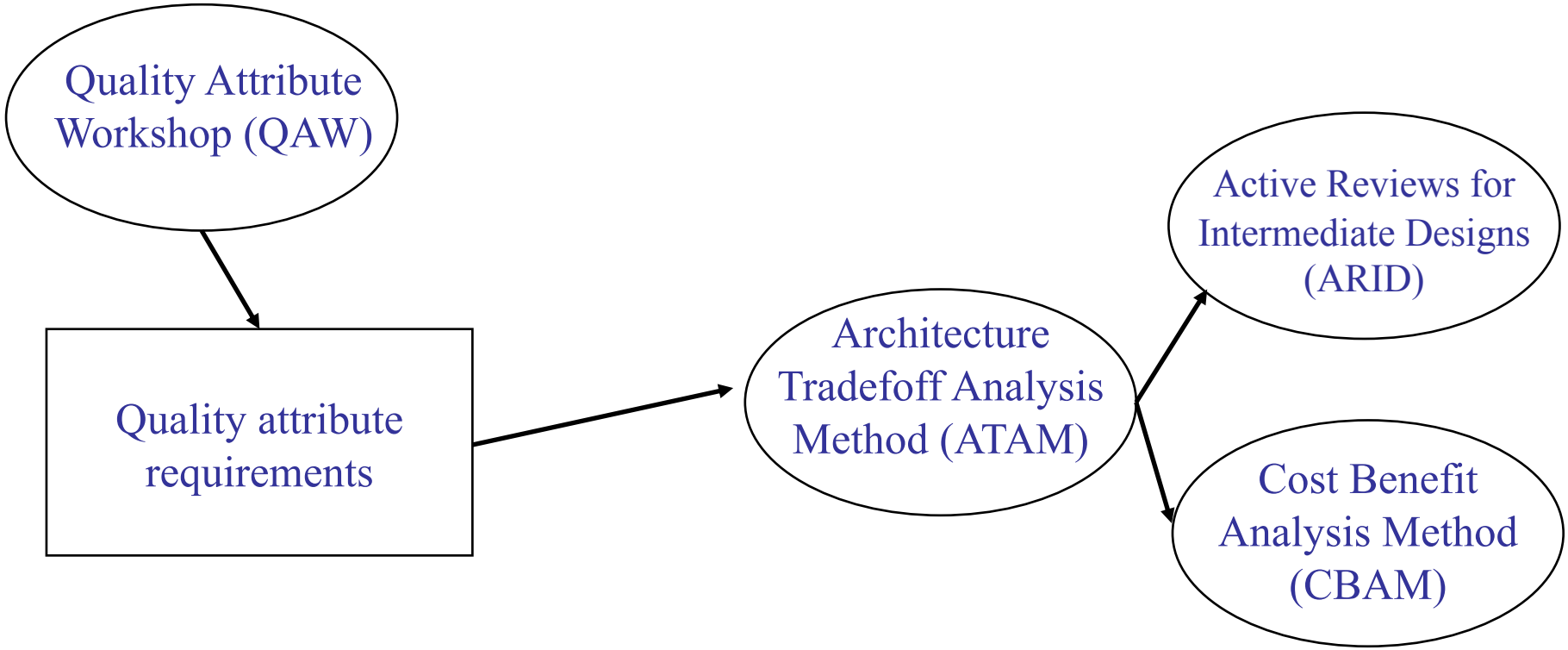


A functional architecture supports the enterprise architecture and is implemented by a technical architecture

Reviewing a functional architecture

- Systems grow and change continuously their form and functions
- A software architect not only creates new systems,
  but also reviews and improves existing systems
- An architecture evaluation review consists of the following phases:
  - **Scoping**: establishing the goal of the review, as well as from one to three key questions the review should answer. For example, a goal could be to validate if a new system architecture implements appropriately the desired dependability quality attributes
  - **Analysis**: read documents, interview stakeholders, check code and test cases, watch demos, so that we are able to answer the key questions
  - **Evaluation**: investigate the strengths, weaknesses, opportunities and threats imposed by the current system and related to the review goal. If there are risks in the system, reviewers should define recommendations to mitigate these risks
  - **Feedback**: all information (findings, recommendations) provided by the review is returned back to the team responsible for system development

Architecture evaluations (SEI)



Reviews

There are many different types of reviews

- If quality attributes are in the main scope, the *Architecture Tradeoff Analysis Method* (ATAM) exploits scenarios as a context for the actual architecture, thus determining the risk themes
- For obtaining information from stakeholders on architecture issues *Active Reviews for Intermediate Designs* (ARID) are an additional means instead of relying on interviews only
- Quantitative assessments such as metrics, prototypes, and simulators, help to obtain more detailed information about the system under review and its capabilities and limitations
- Code and design reviews help reviewers to gain more insights about the details of the system (these reviews are constrained to the code and design parts relevant for the overall review goal)

Evaluating architectures – Why?

- Evaluating a candidate functional architecture before it becomes the blueprint for the technical architecture can be of great economic value
- Useful to:
  - Assess if a candidate architecture can deliver the expected benefits
    - Assessment against stakeholders' requirements/concerns
  - Evaluate a large software system with a long expected lifetime before acquiring it
  - Compare alternatives
  - Architectural refactoring

Evaluating architectures – When?

- Evaluation can take place at many points
- The earlier, the better
  - Software quality cannot be added late in a project, it must be built in from the beginning
- Typically, when the architecture is finished, before the project commits to expensive development
- But also:
  - Compare two competing architectures
  - Evaluate a legacy system undergoing evolution
  - Evaluate the architecture of a system to be acquired
  - Evaluate OTS (Off-The-Shelf) subsystems

Symptoms of architectural deficiency

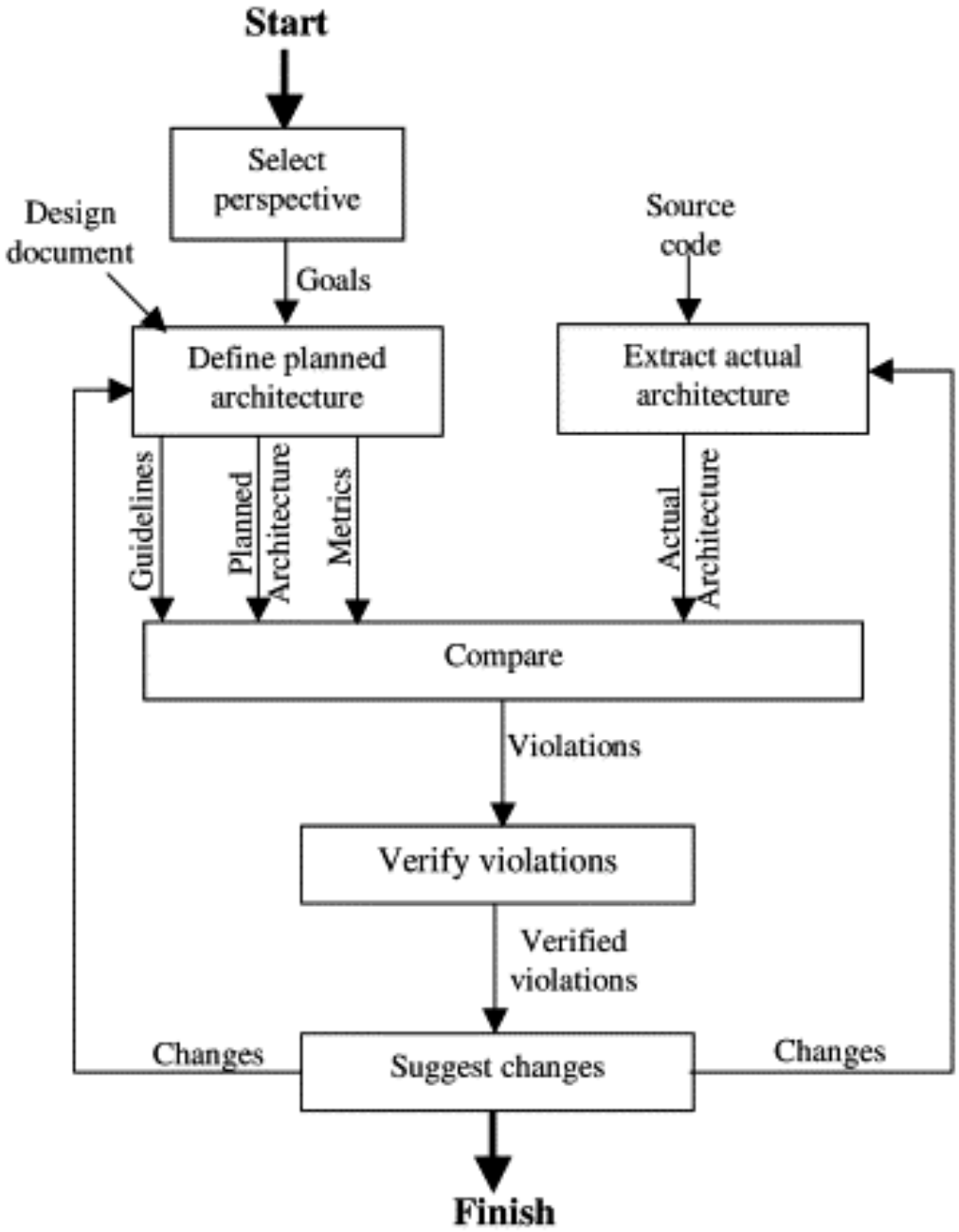
- Operational
  - Communication bottlenecks under various load conditions in systems or throughout system of systems
  - Systems that hang up or crash; portions that need rebooting too often
  - Difficulty synching up after periods of disconnect and resume operations
  - Judgment by users that system is unusable for variety of reasons
  - Database access sluggish and unpredictable
- Developmental
  - Integration schedule blown, difficulty identifying causes of problems
  - Proliferation of patches and workarounds during integration and test
  - Integration of new capabilities taking longer than expected, triggering breaking points for various resources
  - Significant operational problems ensuing despite passage of integration and test
  - Anticipated reuse benefits not being realized

Evaluating architectures – How?

- Some architecture evaluation methods:
  - ATAM www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm
  - CBAM www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm
- Both methods are repeatable and structured

- Planned
  - Scheduled well in advance
  - Built into the project's plan and budget
- Unplanned
  - When the management perceives that the project has risks of failure or needs a correction

A process to evaluate architectures



Types of Architecture Evaluation

- Technical: evaluation against system quality attributes, e.g. performance, security and modifiability, suitability of design decisions
  Eg. Architecture Tradeoff Analysis Method (ATAM)

- Economic: tradeoffs in large systems usually have to do with economics, cost, and benefits associated with architectural design decisions
  Eg. Cost Benefit Analysis Method (CBAM)

ATAM

- ATAM is a scenario-based architecture evaluation method for assessing quality attributes such as: modifiability, portability, variability, and functionality
- ATAM analyses how well software architecture satisfies particular quality goals. It also provides insight into quality attribute interdependencies meaning how they trade-off against each other
- ATAM is based on the Software Architecture Analysis Method (SAAM), another method by SEI

Prerequisites and inputs for ATAM

Prerequisites:

- The evaluators must understand the system architecture, recognize its parameters, define their implications with respect to the system quality attributes
- Problem areas are so called “sensitivity points”, “tradeoff points” and risks. These must be carefully identified
- ATAM is a context-based evaluation method in which quality attributes of the system must be understood. This can be achieved employing descriptive scenarios for evaluating the quality attributes

Inputs:

- The initial requirements of the system
- The software architecture description of the system

ATAM (Architectural Tradeoff Analysis Method)

- ATAM has been defined by the Sw Eng Institute (SEI)
- To conduct a detailed evaluation, one should divide the evaluation process into the following phases:
  - Presentation
  - Analysis
  - Testing
  - Report

ATAM Phases

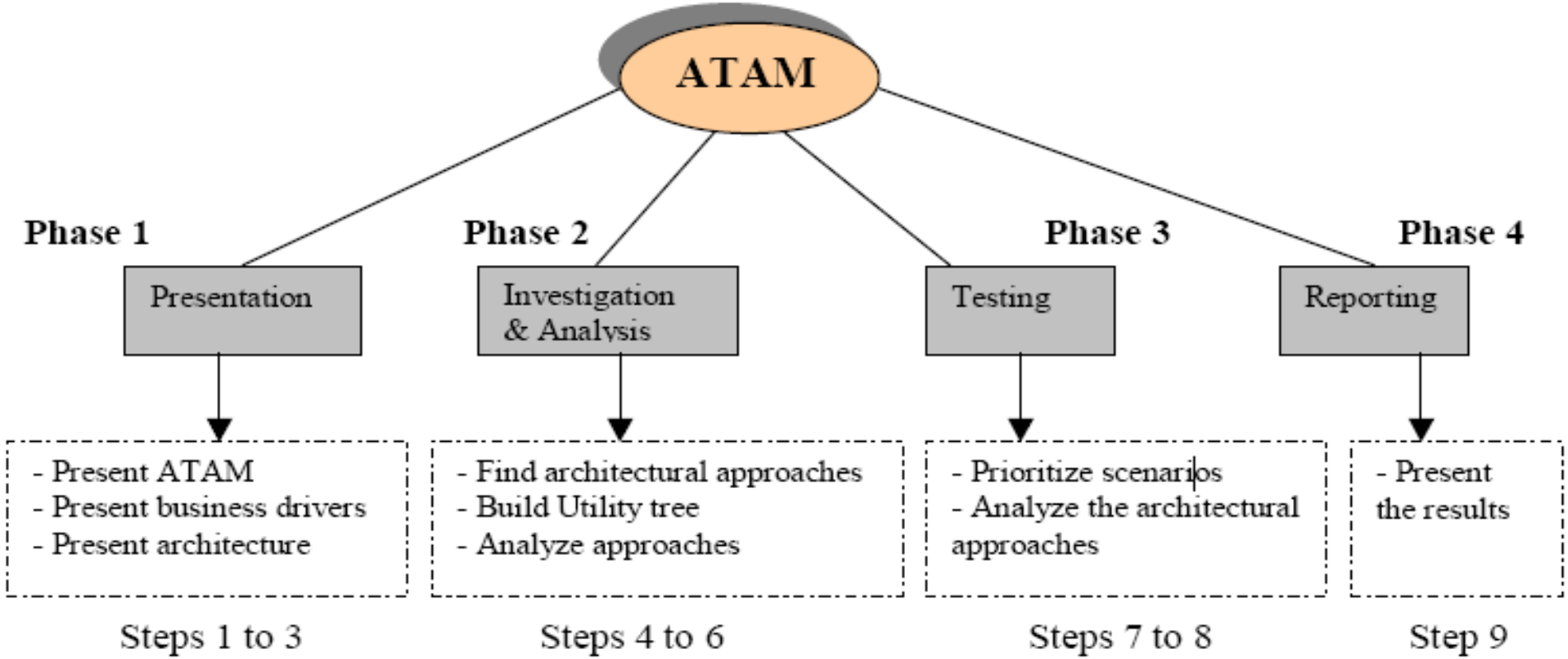


Figure 1: A simple pictorial showing the main steps and sub-steps of the ATAM

Presentation

During this phase, the architect presents

- the process of architectural evaluation by ATAM,
- the business drivers behind the project, and
- a high-level overview of the architecture under evaluation, explaining how it achieves the stated business needs

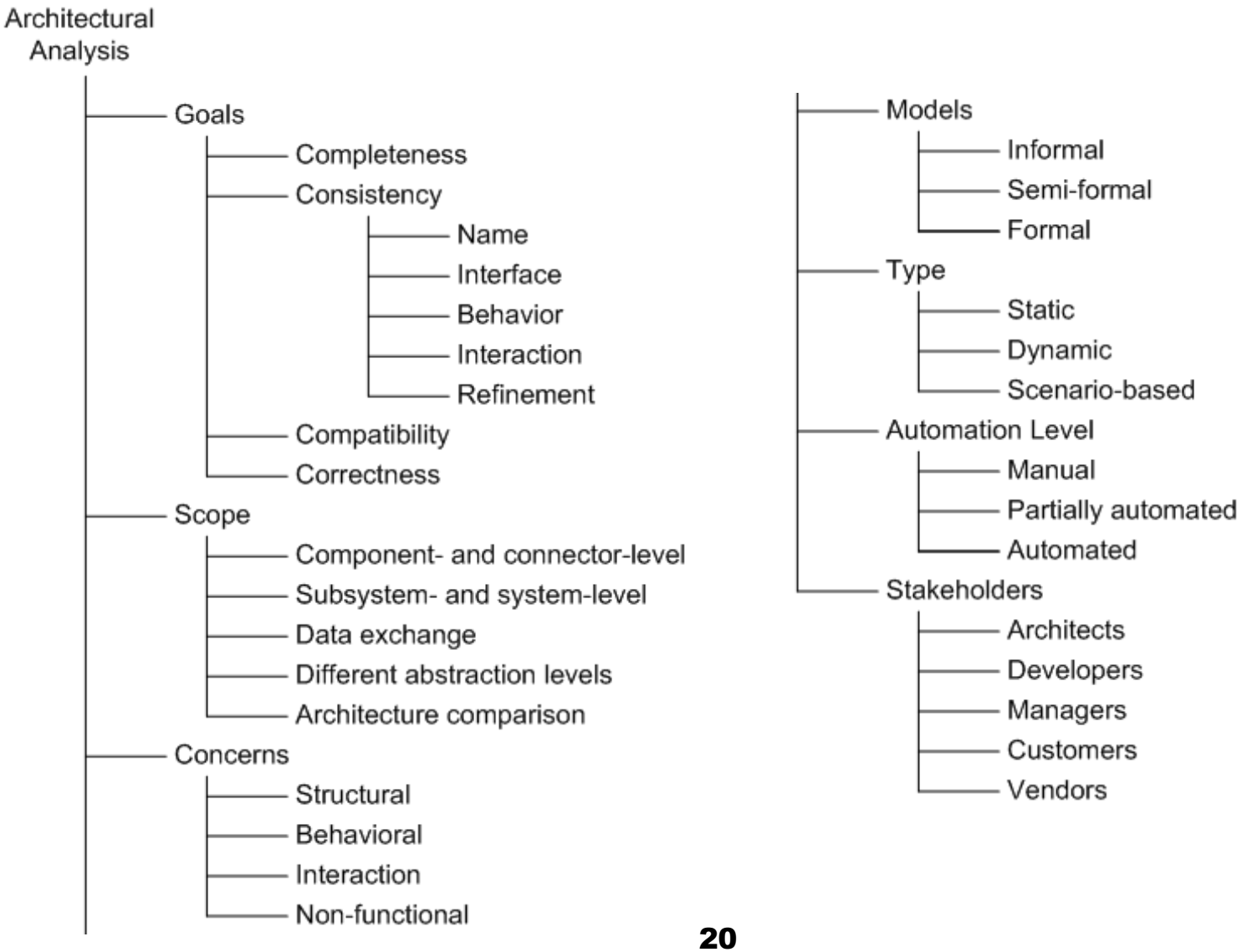
Business drivers presentation

- (~ 12 slides; 45 minutes)
- Description of the business environment, history, market differentiators, driving requirements, stakeholders, current need, and how the proposed system will meet those needs/requirements
- Description of business constraints (e.g., time to market, customer demands, standards, costs)
- Description of the technical constraints (e.g., commercial off-the-shelf products, interoperation with other systems, required hardware or software platform, reuse of legacy code)
- Quality attribute requirements and the business needs from which they are derived
- Glossary

Architecture presentation

- (~ 20 slides; 60 minutes)
- Driving architectural requirements, the measurable quantities you associate with them and any existing standards/models/approaches for meeting them
- Important architectural information: context diagram, module or layer view, component-and-connector view, deployment view
- Architectural approaches, patterns, or tactics employed, including which quality attributes they address and a description of how the approaches address them
  - use of COTS products and how they are chosen/integrated
  - trace of 1 to 3 of the most important use case scenarios
  - trace of 1 to 3 of the most important change scenarios
  - architectural issues/risks with respect to meeting the driving architectural requirements
  - glossary

Architectural Analysis



Analysis

In the analysis phase, the architect:

- discusses possible architectural approaches
- identifies business needs from requirement documents
- generates a quality attribute tree (i.e. an Utility tree) and
- analyzes alternative architectural solutions

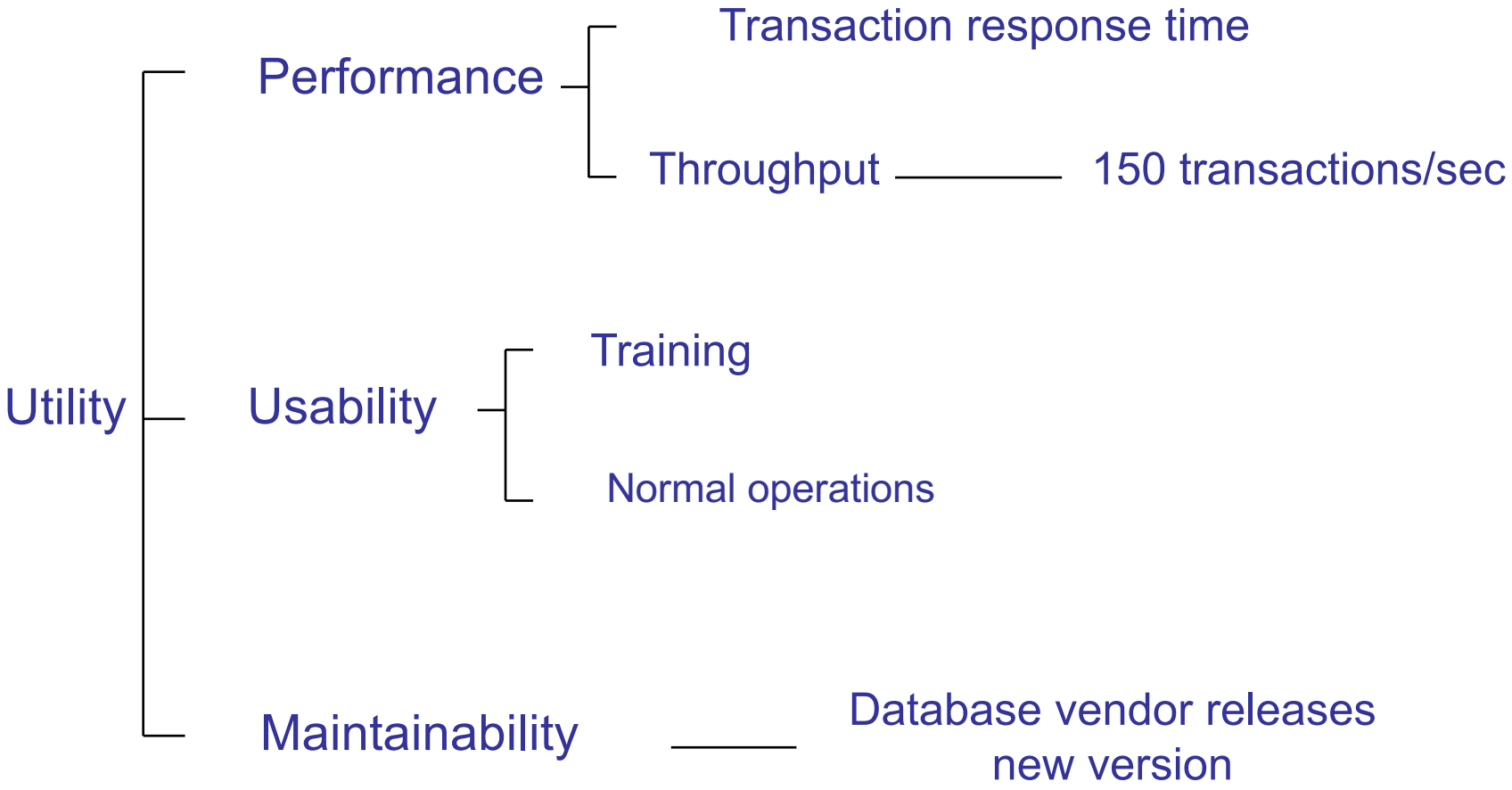
Utility tree

- The root is labeled Utility
- Select the general, important quality attributes to be the high-level nodes: E.g. performance, modifiability, security, availability.
- Refine them to more specific categories
- All leaves of the utility tree are “scenarios”: for each leaf in the utility tree, write a scenario; a scenario has the form of context, stimulus, and response.
  - For example, “Under normal operation, perform a database transaction in fewer than 100 milliseconds.”
- Prioritize scenarios with pairs (*importance*, *risk*)
- Present the quality attribute goals in detail

Key properties in sw intensive systems

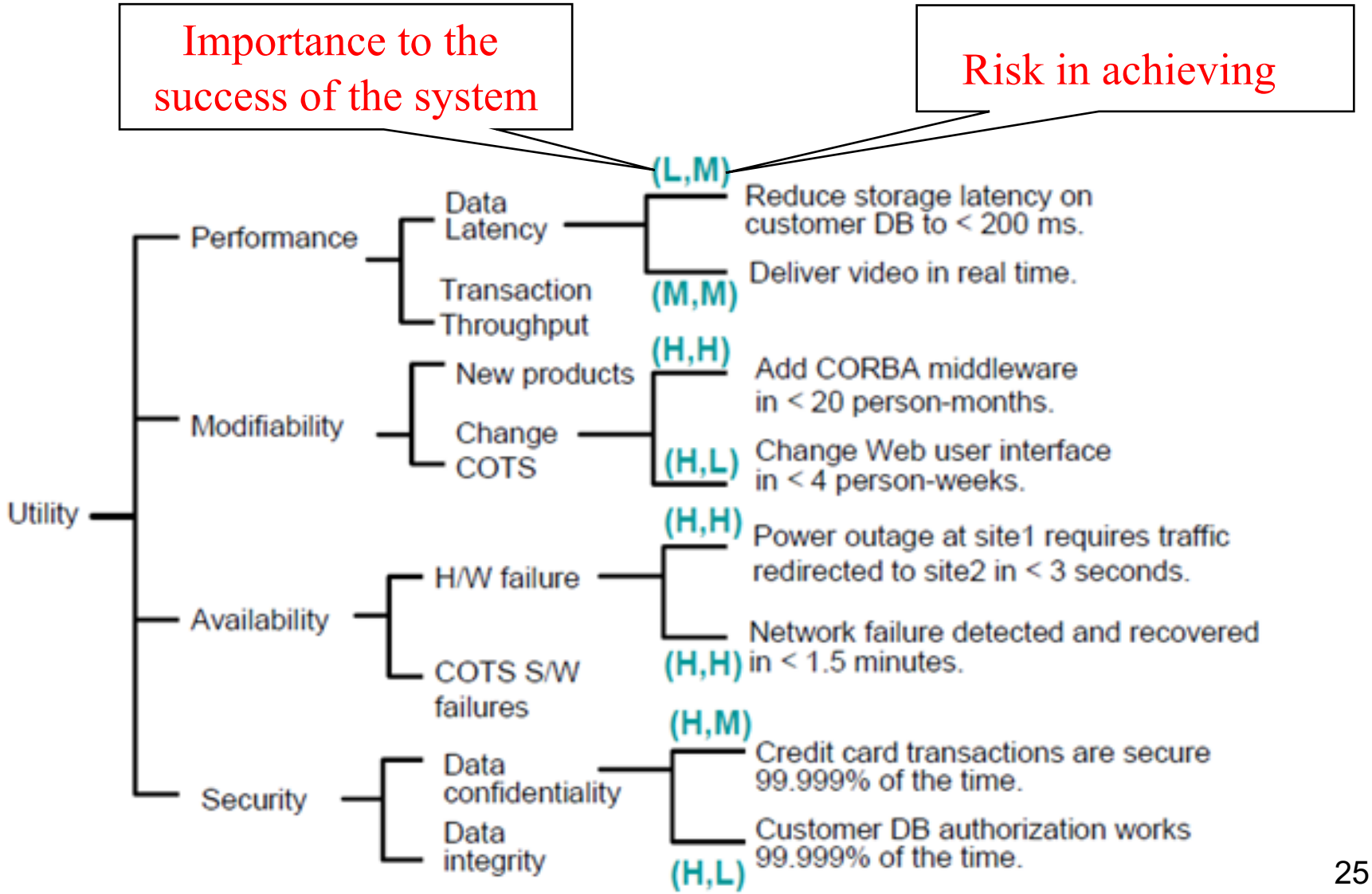
Design time	Run time	Sw in its environment
Time to market	Performance	Usability
Cost and benefit	Security	Supportability
Projected lifetime	Reliability	Configurability
Modifiability	Availability	Sustainability
Maintainability	Scalability	Buildability
Reusability	Interoperability	etc
Portability	Throughput	
Testability	Capacity	
etc	etc	

Example Utility Tree



Utility tree with priorities

A quality attribute tree with priorities in form of pairs (importance, risk)



Testing

- This phase is similar to the analysis phase
- Here, the architect identifies groups of quality attributes and evaluates possible architectural approaches against these groups of attributes

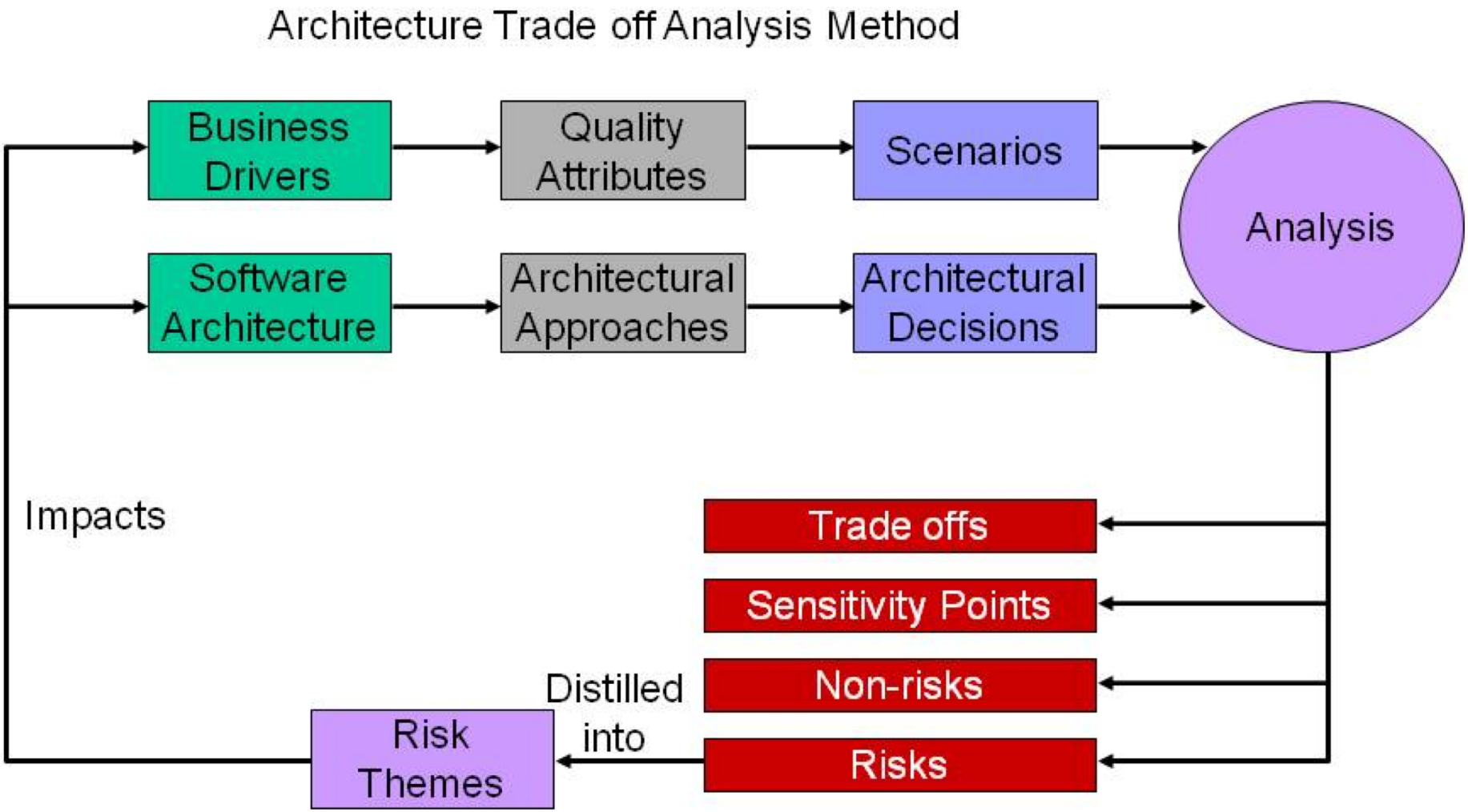
ATAM terminology

Term	Example
Risks are potentially problematic architectural decisions	*The rules for writing business logic modules in the second tier of your three-tier client-server style are not clearly articulated. This could result in replication of functionality, thus compromising modifiability of the third tier*
Nonrisks are good decisions that rely on assumptions that are frequently implicit in the architecture	*Assuming message arrival rates of once per second, a processing time of less than 30 milliseconds, and the existence of one higher priority process, then a one-second soft deadline seems reasonable*
A **sensitivity point** is a property of one or more components (and/or component relationships) that is critical for achieving a particular quality attribute response	*The average number of person-days of effort it takes to maintain the system might be sensitive to the degree of encapsulation of its communication protocols and file formats*
A **trade-off point** involves two or more conflicting sensitivity points	*If the processing of a confidential message has a hard real-time latency requirement, then the level of encryption could be a trade-off point*

Report

- During the report phase, the architect puts together in a document the ideas, views collected, and the list of architectural approaches outlined during the previous phases

ATAM



Conceptual Flow of the ATAM

Who are the participants?

In an ATAM there are:

- Some evaluators (eg. 3 or 4)
- The architect
- A business representative
- Other stakeholders (typically 5 to 30 people)

ATAM outcomes

The outcomes of an ATAM session are:

- The stakeholders understand more clearly the architecture
- Improved software architecture documentation
- Enhanced communication among the stakeholders
- Quality scenarios produced by stakeholders based on the quality attributes requirements

Evaluating architectures – How?

- Preconditions:
  - A few (3 to 5) high-priority goals from the stakeholders
  - Availability of key staff involved
  - Competent evaluation team
- Effort: a few tens of staff-days in a large project
  - ATAM requires approximately 36 staff-days
- Result:
  - A report describing all issues of concern, along with supporting data
  - Report should include costs of the evaluation and estimated benefits if concerns are addressed
  - Report to be first circulated in draft form among participants
  - Issues should be ranked by potential impact on the project if unaddressed

Evaluating architectures – Benefits

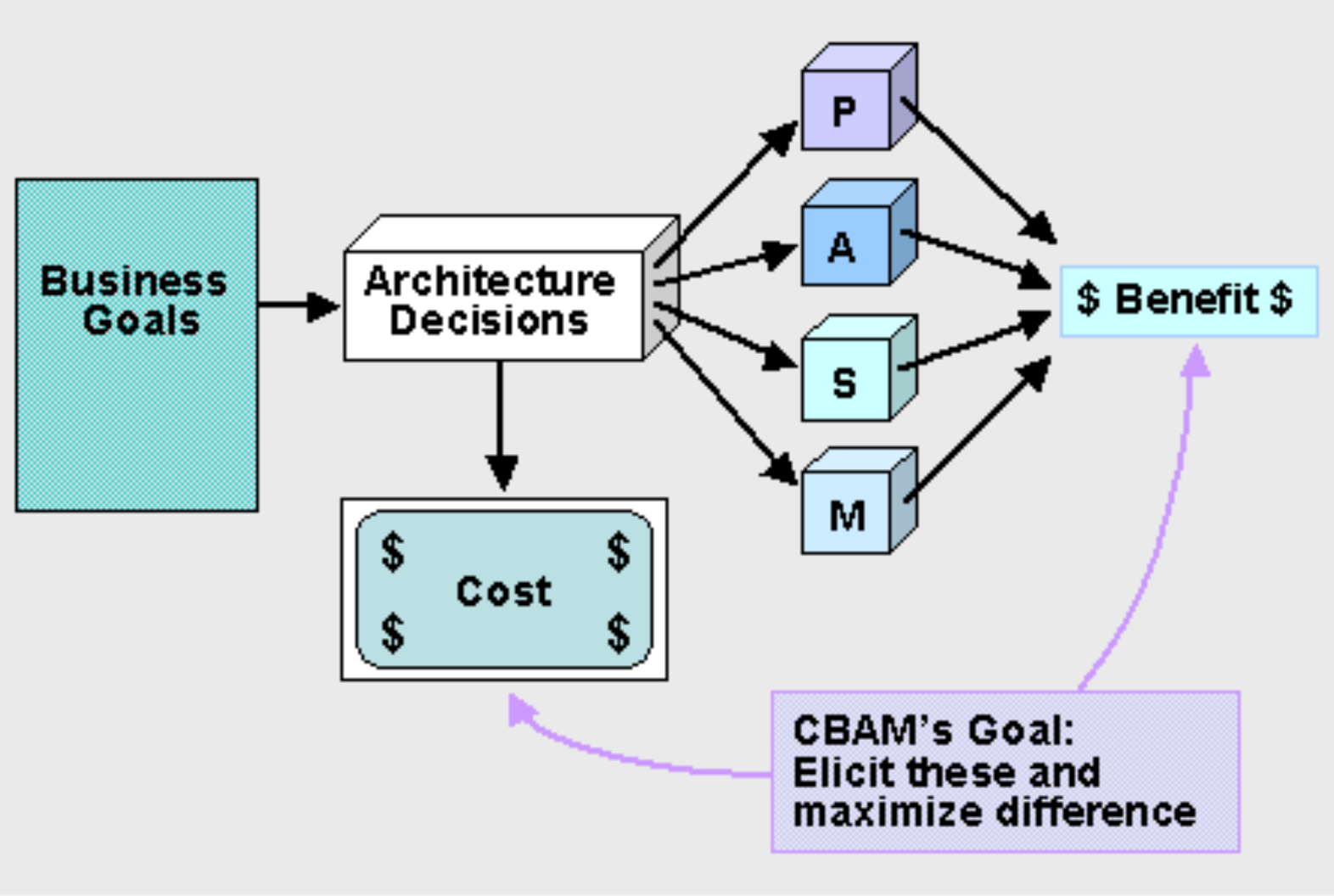
(Especially for planned evaluations)

- Economic / financial
- Preparation for review
- Captured rationale
- Early detection of problems
- Validation of requirements
- Improved final architecture
- Ensuring proper documentation

Cost-Benefit Analysis Method (CBAM)

- CBAM is an architecture-centric **method** for analyzing the costs, benefits and schedule implications of architectural decisions
- ATAM considered the design decisions with respect to architectural quality attributes like performance, availability, security, modifiability, usability, and so on
- CBAM adds the costs (implicit budgets or money) as quality attributes

CBAM



http://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm

Inputs for CBAM

Inputs in a CBAM evaluation session are:

- The presentation of the business goals
- The architectural decisions and possible tradeoffs resulted in a former ATAM session
- The quality attributes expectation level and economical constraints (budget)

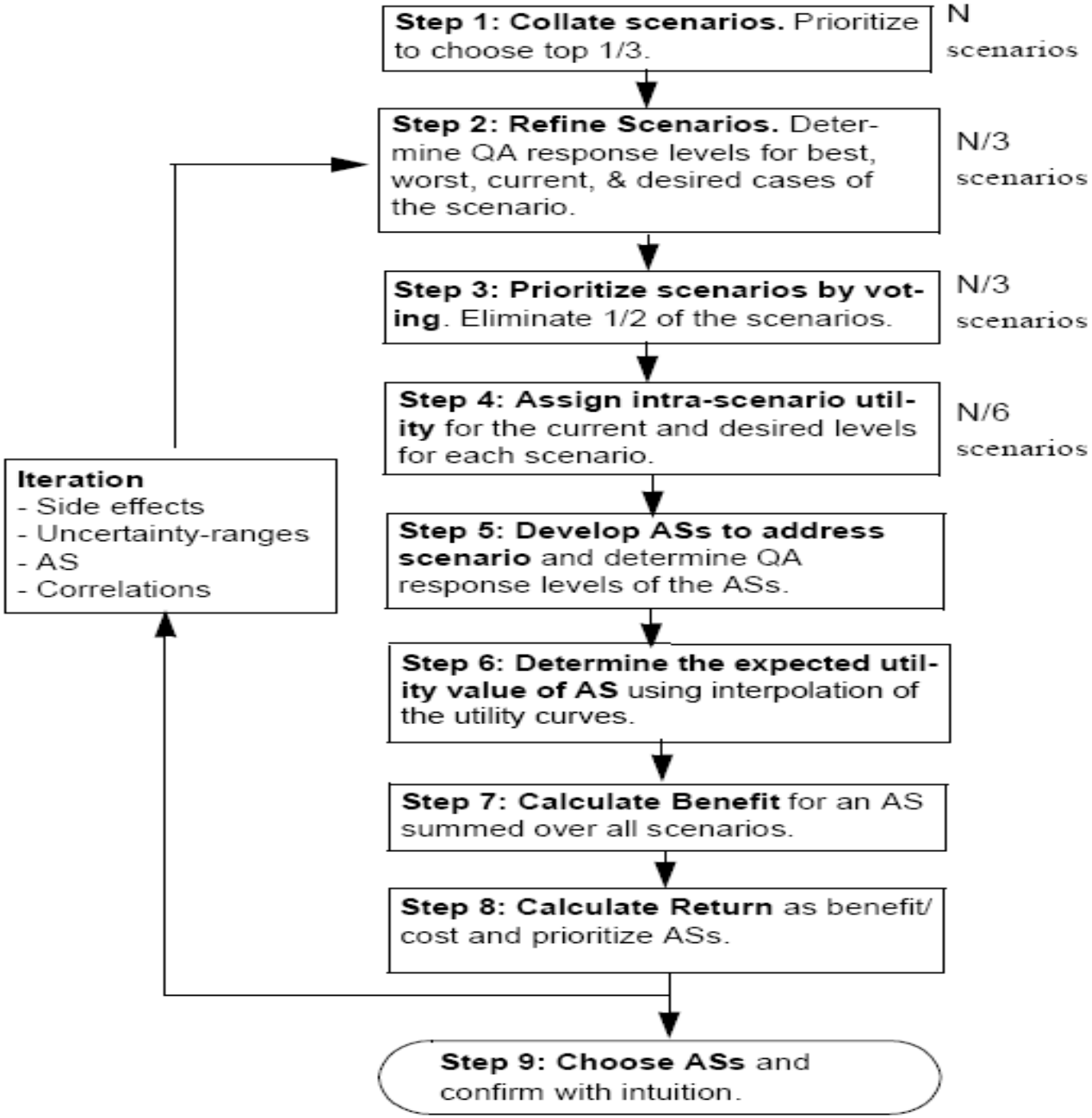


Figure 2: Process Flow Diagram for the CBAM

CBAM outcomes

- CBAM provides the basis for a rational decision making process in applying some architectural strategies
- The method provides a business measure that can determine the level of return on investment of a particular change to the system
- The method helps architects in analyzing and pre-evaluating the resource investment in different directions by adopting those architectural strategies that are maximizing the gains and minimize the risks

Summary

- Software architecture is a matter of social consensus, so it is its evaluation
- The main method for architecture evaluation is ATAM, based on scenarios evaluated via a utility tree

Template of the report

1. Descrizione del problema o sistema
2. Descrizione dell'architettura
    1. Contesto (in quali ambiti si usa questa architettura?)
    2. Proprietà rilevanti (quali requisiti specie non funzionali sono rilevanti?)
    3. Struttura (quali componenti e connettori? se sono noti, quali package?)
    4. Funzioni (come si mappano i requisiti sui componenti, ovvero cosa fa ciascun componente)
    5. Comportamento (come funziona l'architettura?)
    6. Razionale (perché è fatta cosi? che stili usa?)
3. Aspetti analitici (quali sono i punti critici dell'architettura? che test si eseguono di solito?
4. Descrizione di architetture simili o stili architetturali derivati
5. Bibliografia

Self questions

- In which way we evaluate a software architecture?
- What is ATAM?
- What is a sensitivity point in ATAM?
- What is CBAM?

References

- Clemens & Kazman & Klein, *Evaluating Software Architectures: Methods and Case Studies*, AddisonWesley, 2001
- Bass & Clemens & Kazman, *Software Architecture in practice*, AW 2012
- Spinellis & Gousios, *Beautiful Architecture: Leading Thinkers Reveal the Hidden Beauty in Software Design*, O'Reilly, 2009
- Roy & Graham, Methods for Evaluating Software Architecture: A Survey, TR Queen's University at Kingston Ontario, Canada, 2008
- Tesoriero Tvedt & Costa & Lindvall, Evaluating Software Architectures, *Advances in computers*, 61:1-43, 2004
- Nord et al., Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM), SEI 2003

Sites, blogs and wikis

- www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm
- www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm
- www.gaudisite.nl
- etutorials.org/Programming/Software+architecture+in+practice,+second+edition/

Questions?