ATAM

09/21/2020

Overview

ATAM

- Quality Attributes:
  - Modifiabilty
  - Performance
  - Reliability
  - Security
- Claim 1: Quality Attributes of systems interact
- Claim 2: Quality attributes depend on architecture more that code level practices

Really?

Do we believe those claims? Under what circumstances?

Goals

- Identify the options for architecture
- Identify the tradeoffs between those architectures
- Facilitate communication between stakeholders
- Clarify and refine requirements
- Results: framework of ongoing, concurrent process of system design and analysis

Why use ATAM?

- Structured method of analysis
  - Helps us ask good questions early
  - Looking for "conflicts in requirements"
- Assumptions
  - Attribute-specific analyses are interdependent
  - Quality attributes are connected to each other through architectural elements
    - Architectural elements are components, their properties and the relationships between then
  - These connections are "tradeoff points"

What is "Architecture"

- Structure of the whole system (software and hardware)
- Architect's Job
  - Lay out the architecture
  - Meeting requirements (aspects like throughput, expected load, latency)
  - Withing cost constraints
- Meeting requirements depends on architecture:
  - policy for allocating processes to processors
  - scheduling of concurrent processes on one processor
  - managing access to shared data stores

Spiral Model of Design

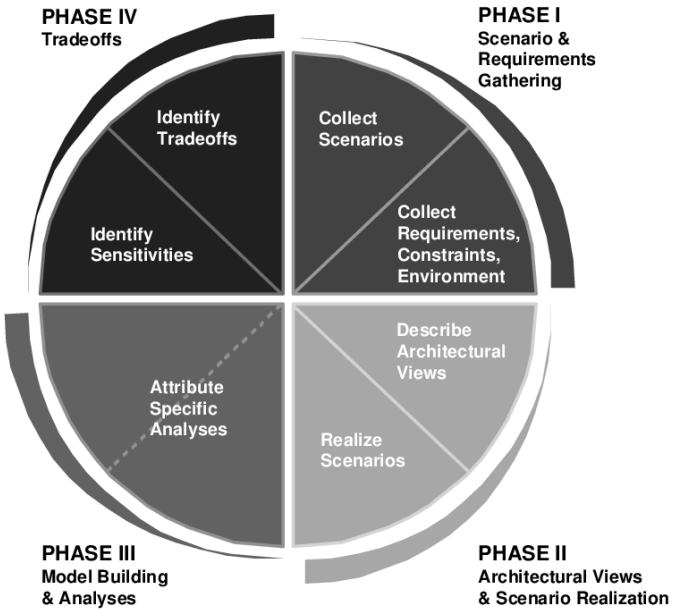

Figure: ATAM Process

- Spiral model like Boehm's
- Each iteration increases the depth of understanding of system and risks
- Implementation isn't required {really?}

Newer Description

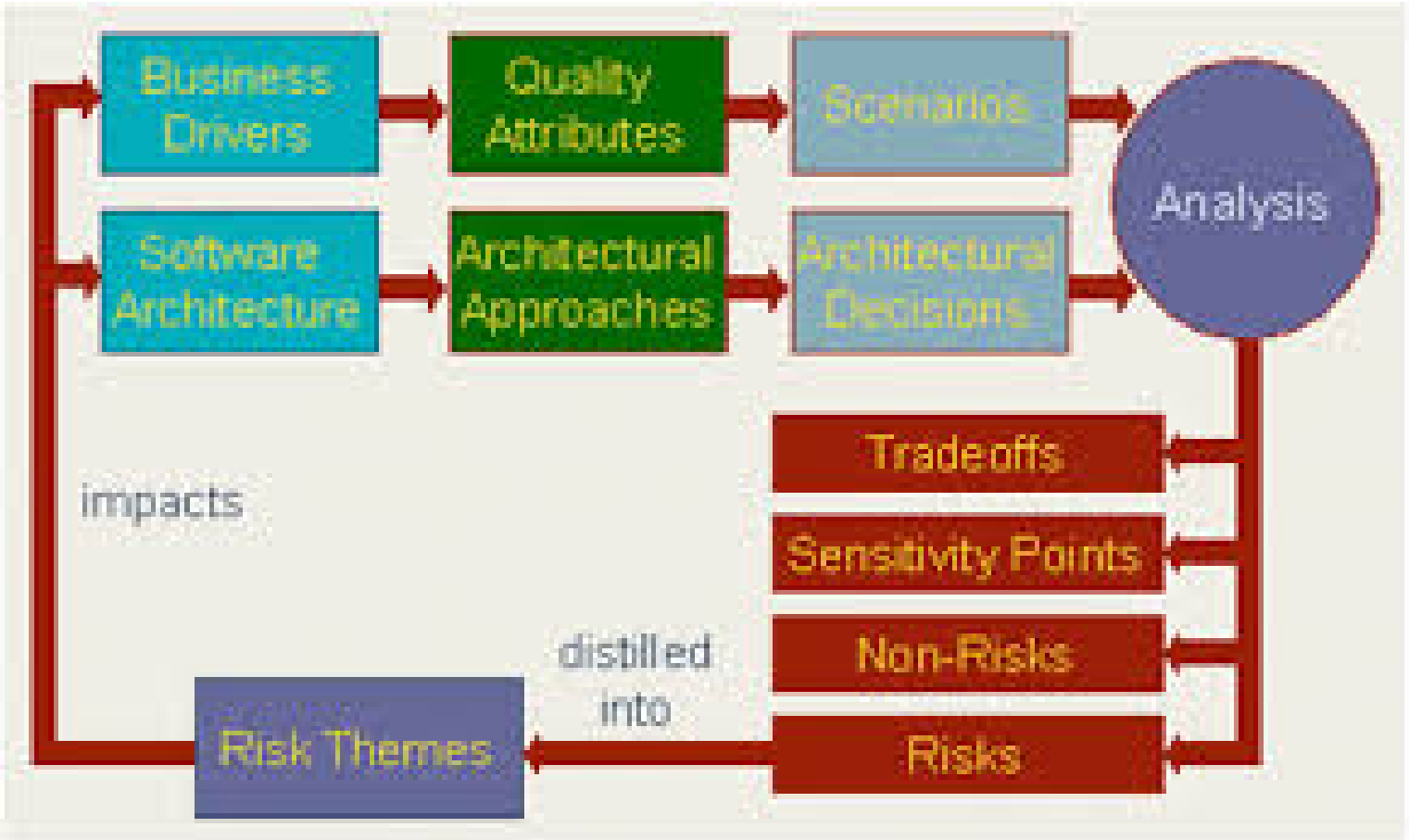

Figure: Newer View of ATAM

Step 1 Collect Scenarios

- Elicit system usage scenarios (things a client does to the system and how it reacts)
- Can be concurrent with Step 2
- Goal
  - Operationalize both functional and quality requirements
  - Facilitate communication between stakeholders
  - Develop a common vision of the important activities the system should support

Step 2 Collect Requirements/Constraints/Environment

- identify quality-attribute-based requirements, constraints and environment of the system
- Requirements may have specific targets or be more generic
- I can't tell exactly what he means by "environment"
  - From the example, it seems like they are requirements that are not specific to one quality attribute

Step 3 Describe Architectural Views

- Describe "candidate" architectures
  - Architectural elements related to quality attributes
  - Example: Reliability requirements could lead to an architecture that includes voting schemes
- Architectural Views
  - Module View (Class/Layer/Package/etc.)
  - Process View (threads/systems)
  - Dataflow View (how information flows through the system)

Step 4 Attribute-Specific Analysis

- For each architecture
  - Analyze each quality attribute
- Recognize that there may be different experts for different quality attributes
- My concern here is how is this valid with the initial claims they made?

Step 5 Identify Sensitivities

- For each architecture, determine how sensitive it is with respect to each quality attribute
- Vary the models to weigh different designs
- Model values that significantly affect an attribute are called "sensitivity points"

Step 6 Identify Tradeoffs

- Here we are focusing our analysis on the interactions between quality attributes
- Tradeoff points are architectural elements to which multiple attributes are sensitive
- Often, change the architecture improves performance against one attribute at the cost of performance against another attribute.

Go through the example together (ATAMMarkedUp.pdf)

Trading Cost for Knowledge

- He means there are costs to making the system's performance more precise (knowledge = know it will happen in time)
- We are trying to find what aspects are really important and be explicit about what those things may cost
- He admits we can't believe the numbers - orders of magnitude are the best we can hope for
- This makes the relationships between quality attributes explicit - ex: adding retries on a comm link improves reliability, but then we have to re-do the performance analysis

Where does this fit in the development process?

- Plan-driven: It is part of Requirements Analysis (prior to High-Level Design)
- Agile:
  - Early iterations - let's us know what decisions we can defer to later
  - Later iterations - now we can use some real numbers
  - Risk - some big architectural decisions have to be made early - you can't refactor everything . . .