Architecture Tradeoff Analysis Method

- Introduction
  - Claim: performance, availability and modifiability depend more on architecture than code-level practices.
  - Quality attributes of systems interact.
  - We need a standard strategy for weighing the tradeoffs of multiple attributes during architecture design
  - Attributes
    - modifiability
    - performance
    - reliability
    - security
  - Goals
    - identify tradeoffs
    - facilitate communication between stakeholders
    - clarify and refine requirements
    - framework for ongoing, concurrent process of system design and analysis
- Why Use Architecture Tradeoff Analysis?
  - 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 them
    - these connections are "tradeoff points"
- The ATAM[1]
  - Spiral model of design (like Boehm's model), but implementation isn't required (REALLY???)
  - Newer descriptions are more details:[2]

[1]Figure 1 citation: `http://www.jot.fm/issues/issue_2003_03/article4/`

[2]Figure 2 citation: `https://people.cs.clemson.edu/~johnmc/conferences/splc2010/PublishTest/ProductLineTestingMethod/capabilitypatterns/ATAM_C473A8D6.html_desc.html?nodeId=e4fbe139`

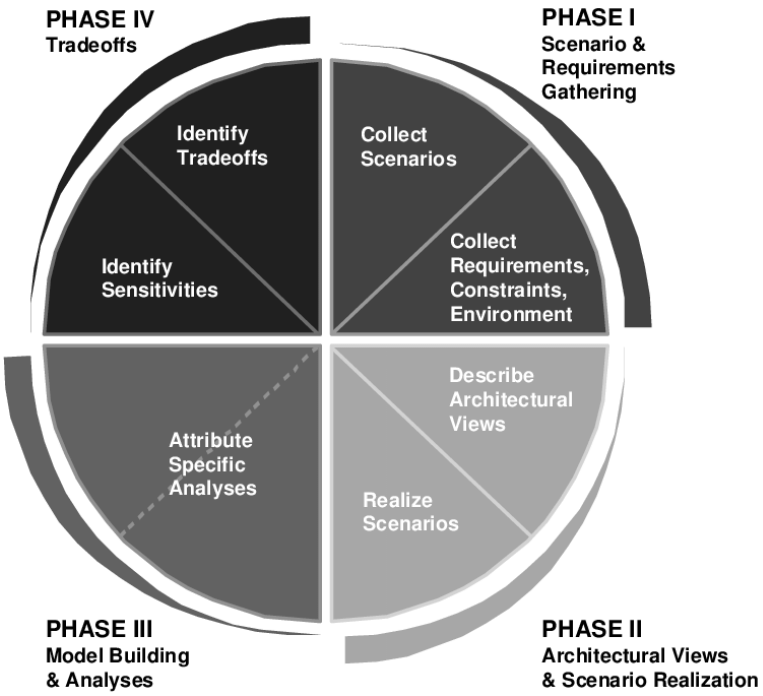

Figure 1: The Spiral View of ATAM

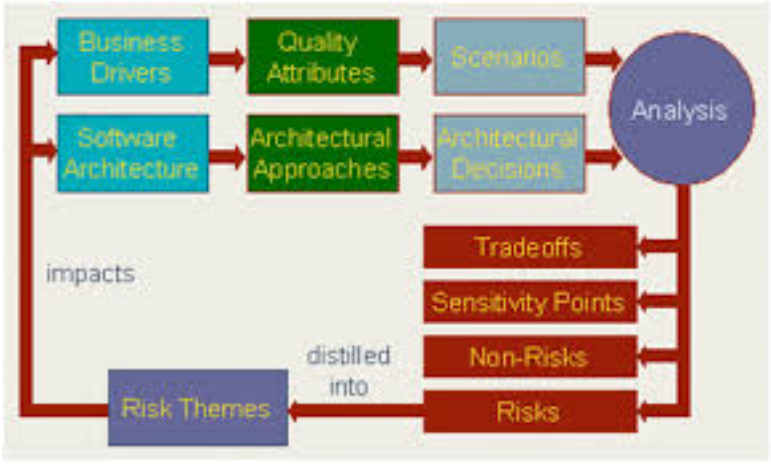

Figure 2: The Newer View of ATAM

  - Goal of an architect: design "an architecture that will lead to system behavior which is as close as possible to the requirements within the cost constraints."

- The Steps of the Method
  - Collect Scenarios
    * We would call these use cases
    * Include requirements, constraints and environmental details
    * This discussion includes appropriate stakeholders
    * Builds a common vision of the system
  - Collect Requirements/Constraints/Environment
    * attribute-based requirements, constraints, and environment
    * sometimes these are called non-functional requirements
  - Describe Architectural Views
    * Generate candidate architectures
    * Constrain the design possibilities
    * Pay attention to what you inherit (existing systems)
    * Specify properties of each architecture as they related to the important quality attributes
    * Look at it from different perspectives
      · module view (the parts of the system)
      · process view - where things run (threads and/or systems) so we can think about performance
      · dataflow view - this is an old technique that modeled how information flowed through a system (before object-oriented design)
      · class view - the OO view
  - Attribute-Specific Analyses
    * For each architecture, analyze each quality attribute in isolation
  - Identify Sensitivities
    * How sensitive the analysis of individual attributes is to particular architectural elements?
    * Vary one or more attributes of the architecture (those perspectives) and evaluate how quality attributes are affected.
    * Sensitivity points: values of attributes that are significantly affected by a change to the architecture
  - Identify Tradeoffs
    * Critique the models in the architectural views and find tradeoff points.
    * Focus on the interaction of attribute-specific analyses particularly at sensitivity points.

- Iterations of the ATAM
  - Continue to do this throughout the lifecycle of the systems
  - And the steps aren't really linear

- Let's go through the example together