Software Architecture

Lecture 10
Architecture Analysis

João Pedro Sousa
George Mason University

previously
QA scenarios

- architectural drivers shape the architecture
  - high-level functional requirements
  - constraints
  - quality attributes (QAs)
- QA names are vague:
  need to characterize QAs using scenarios
- QAW is a method to elicit
  and prioritize QA scenarios
- can't have it all:
  architectural design is about balancing tradeoffs

today
architectural analysis

- inspections and reviews
  - ATAM
- model-based analysis
  - simulation: XTEAM etc.
  - analytic: patterns

Acknowledgment
some of the material presented in this lecture is adapted from the companion slides to the Taylor et al. textbook

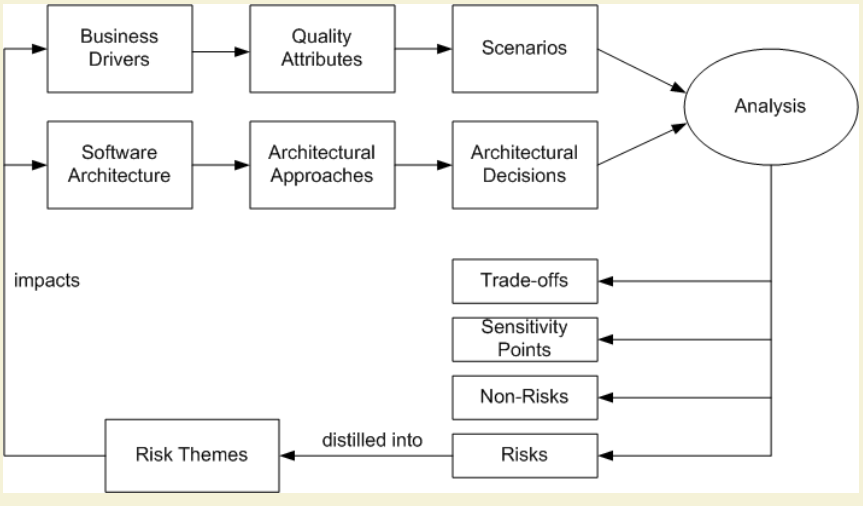
inspections and reviews

- architectural models studied by human stakeholders
- the stakeholders define specific analysis objective
- suited to informal architectural descriptions
- may consider multiple stakeholders' objectives and multiple architectural properties
- useful for making qualitative comparisons
  - e.g., comparisons of scalability or adaptability
- complement formal techniques
  - drilldown of best candidates
  - more work

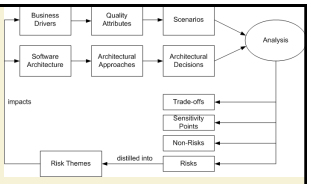
inspections and reviews
example: ATAM

- Architectural Trade-off Analysis Method
- human-centric process for identifying risks early on
- emphasizes quality attributes
  - modifiability
  - security
  - performance
  - reliability
- reveals
  - how well an architecture satisfies quality goals and
  - how those goals trade off

ATAM Process

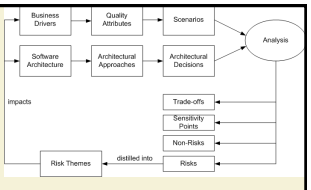


drivers

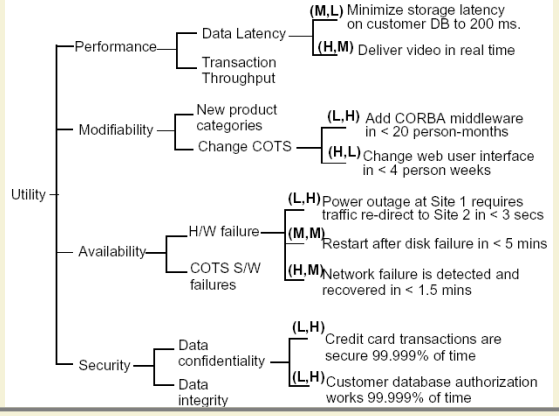


- critical functionality
- constraints set by major stakeholders
  - technical, managerial, economic, or political
  - business goals and context
- QA goals

QAs
elicited with a Utility Tree

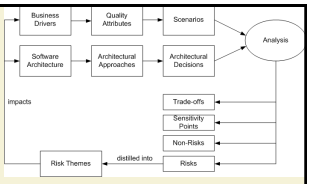


- top-down decomposition of key QAs
- output: characterization and prioritization of QA scenarios
  - scenarios are the leaves of the utility tree



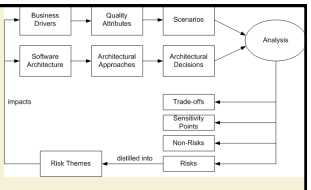
scenarios annotated with (importance to success, risk/difficulty to achieve)

brainstorm different scenarios



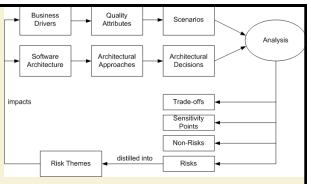
- use-case scenarios
  - how the system is envisioned to be used
- growth scenarios
  - planned and envisioned evolution
- exploratory scenarios
  - establish the limits of adaptability wrt
    - functionality
    - operational profiles
    - underlying execution platforms
- prioritize scenarios based on
  - importance to stakeholders
  - risk/difficulty to achieve

architectural thinking architectural approaches



- identify features that are key for realizing QA goals
- identify candidate architectural styles & patterns
  - client-server
  - 3-tier
    - load balancing
    - first-to-respond
    - …
  - publish-subscribe
  - pipe-and-filter
  - …

analysis
relate approaches to QAs

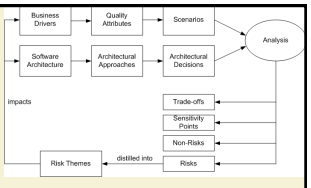


- for each approach, formulate a set of analysis questions
  - driven by the scenarios
- identify
  - risks → distilled into risk *themes*
  - non-Risks
  - sensitivity points
  - tradeoffs

more in a bit

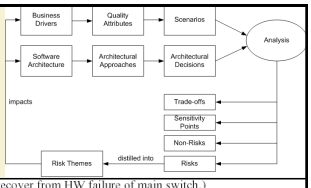
- identify strategies
  - to address risks and tradeoffs
  - further analysis may be needed

analysis
relate approaches to QAs



- sensitivity points
  relate design decisions to important QAs
  - confidentiality is sensitive to encryption key size
    - the larger the key the more confidentiality
  - reliability is sensitive to sensor redundancy
    - the more redundant sensors the more confidence in the value
- tradeoffs
  relate QAs to each other
  - the more confidentiality (heavier encryption) the worse performance
  - the more reliability (the more sensors) the worse development and maintenance costs

example analysis
relate approaches to QAs



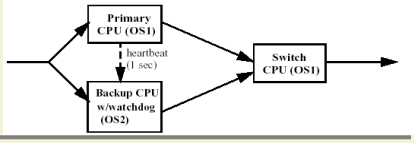
- for each approach, formulate a set of analysis questions
  - driven by the scenarios
- identify
  - risks → distilled into risk *themes*
  - non-Risks
  - sensitivity points
  - tradeoffs
- identify strategies
  - to address risks and tradeoffs
  - further analysis may be needed

Scenario: S12 (Detect and recover from HW failure of main switch.)
Attribute: Availability
Environment: normal operations
Stimulus: CPU failure
Response: 0.999999 availability of switch

Architectural decisions	Risk	Sensitivity	Tradeoff
Backup CPU(s)	R8	S2	
No backup Data Channel	R9	S3	T3
Watchdog		S4	
Heartbeat		S5	
Failover routing		S6	

Reasoning:
- ensures no common mode failure by using different hardware and operating system (see Risk 8)
- worst-case rollover is accomplished in 4 seconds as computing state takes …
- guaranteed to detect failure with 2 seconds based on rates of heartbeat and watchdog …
- watchdog is simple and proven reliable
- availability requirement might be at risk due to lack of backup data channel … (see Risk 9)

Architecture diagram:



outline
architectural analysis

- inspections and reviews
  - ATAM
- model-based analysis
  - simulation: XTEAM etc.
  - analytic: patterns

model-based analysis
complementary to inspections and reviews

- require rigorous architectural descriptions
  - Architecture Description Languages ADL with well-defined semantics
- benefits from tool support
  - syntactic correctness, adherence to a style
  - behavioral checking
    - deadlock freedom
    - model checking
  - simulation
    - QAs
- scalability of tool/size of model may be an issue
- can also be done by hand
  - analytic models of QAs

more in a bit

model-based analysis
ADLs that support analysis

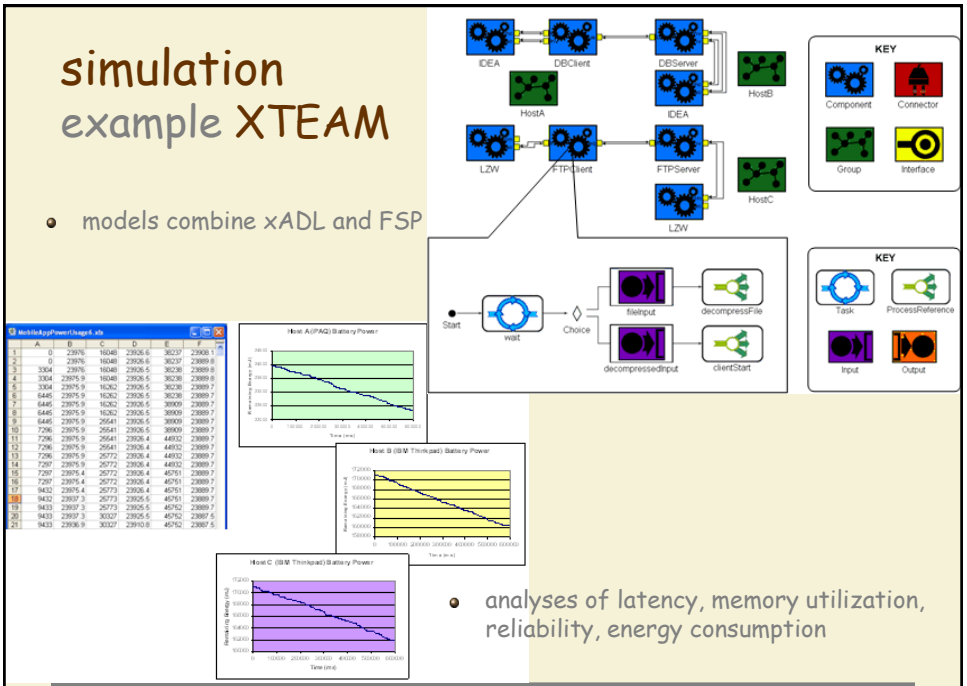
- Armani, Aesop and UML's OCL – enforce style constraints
- SADL and Rapide - architectural refinement
- Rapide - generates executable architectural simulations
- Wright – uses CSP to analyze for deadlocks
- MetaH and UniCon – support schedulability analysis via NFPs such as component criticality and priority
- all – ensure syntactic and semantic correctness

simulation
requires an executable model

- simulation abstracts system behavior
  - hides unnecessary complexity
  - behavior may differ from implementation
  - crucial to have accurate-enough models
- some tools combine ADLs to improve expressive power
- example XTEAM
  eXtensible Tool-chain for Evaluation of Architectural Models
  - combines: xADL and FSP
  - maps architectural model to *adevs*, an o-t-s event simulation engine
  - supports different analyses
    - latency, memory utilization, reliability, energy consumption

simulation
example XTEAM

- models combine xADL and FSP



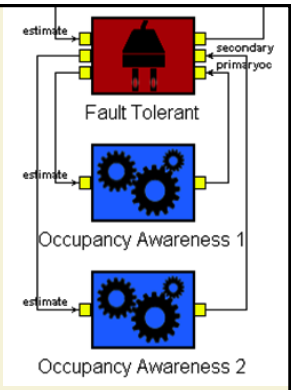
- analyses of latency, memory utilization, reliability, energy consumption

outline
architectural analysis

- inspections and reviews
  - ATAM
- model-based analysis
  - simulation: XTEAM etc.
  - analytic: patterns

analytic modeling
relate patterns to QAs

- structural description of pattern
  - first-to-respond
  - load balancing
  - 2-phase commit
  - …
- analytic model of QA
  - availability
  - execution time aka latency
  - …



xADL

$$a = \mathcal{M}_a(v_1, v_2...)$$
$$e = \mathcal{M}_e(v_1, v_2...)$$

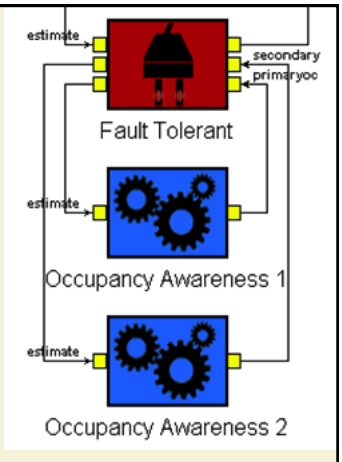
analytic modeling
base

c

- single component, c
  - availability $v_{a,c}$
  - execution time $v_{e,c}$
- asynchronous message passing

$$a = \mathcal{M}_a(v_{a,c}) = v_{a,c}$$
$$e = \mathcal{M}_e(v_{e,c}) = v_{e,c}$$

analytic modeling
load balancing

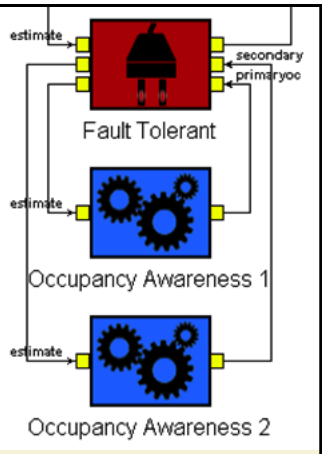


- connector
  - connector sends request to each component Cj in turn e.g. round-robin
  - sends each reply back to client

$$a = \mathcal{M}_a(v_{a,C_1}, \cdots, v_{a,C_C}) = \frac{1}{C}\sum_{j=1}^{C} v_{a,C_j}$$

$$e = \frac{1}{C}\sum_{j=1}^{C} v_{a,C_j} \times v_{e,C_j}$$

analytic modeling
parallel invocation

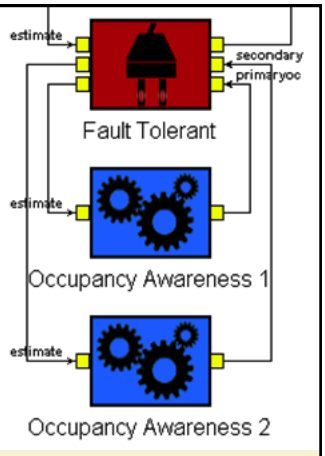


- connector
  - connector breaks request prior-proc into C sub-requests and sends to each component Cj
  - merges the replies of all components post-proc

$$a = v_{a,\text{connector}} \prod_{j=1}^{C} v_{a,C_j}$$

$$e = v_{e,\text{connector}-\text{prior}} + \max\{v_{e,C_1}, \cdots, v_{e,C_C}\} + v_{e,\text{connector}-\text{post}}$$

analytic modeling
fault-tolerant first-to-respond



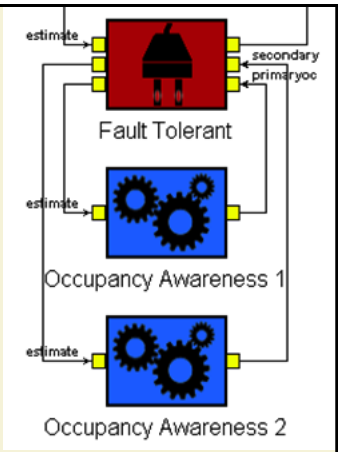
- connector
  - sends request estimate to all C components
  - takes first response

$$a = \mathcal{M}_a(v_{a,C_1}, \cdots, v_{a,C_C}) = 1 - \prod_{j=1}^{C}(1 - v_{a,C_j})$$

$$e = \frac{1}{a}\left[v_{a,C_1}(1 - v_{a,C_2})v_{e,C1} + v_{a,C_2}(1 - v_{a,C_1})v_{e,C2} + v_{a,C_1}v_{a,C_2}\min\{v_{e,C1}, v_{e,C2}\}\right]. \quad (5)$$

analytic modeling
fault-tolerant 2-phase commit

- connector
  - sends request estimate to all C components
  - waits for all to reply
  - sends commit to all and replies to client



$$a = \mathcal{M}_a(v_{a,C_1}, \cdots, v_{a,C_C}) = \prod_{j=1}^{C} v_{a,C_j}$$

$$e = 2 \times \max\{v_{e,C_1}, \cdots, v_{e,C_C}\}$$

in summary
analysis

- is neither easy nor cheap
- Early information about the system's key characteristics is indispensable
- Multiple analysis techniques often should be used in concert
- "How much analysis?"
  - This is the key facet of an architect's job
  - Too many will expend resources unnecessarily
  - Too few will carry the risk of propagating defects into the final system
  - Wrong analyses will have both drawbacks
- The benefits typically far outweigh the drawbacks