Software Architecture

Hans van Vliet
Vrije Universiteit, Amsterdam
email: hans@cs.vu.nl

Overview

- **What is it, why bother?**
- Issues only touched upon
  - Architectural styles & patterns
  - Product lines
  - Architecture design
- Architecture representations
- Architecture assessment

Anything New?

- **New wine in old bottles?**
- **Software architecture ≅ global design?**
- **Architect ≅ designer?**

Software architecture, definition (1)

The architecture of a software system defines that system in terms of computational components and interactions among those components.

(from Shaw and Garlan, *Software Architecture, Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.)

Software Architecture

statement
⇓
procedure
⇓
module
⇓
(design) pattern
⇓
architecture

Software Architecture, definition (2)

The software architecture of a system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

(from Bass, Clements, and Kazman, *Software Architecture in Practice*, SEI Series in Software Engineering. Addison-Wesley, 2003.)

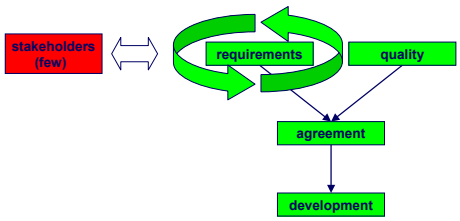
Other points of view

- Architecture is high-level design
- Architecture is overall structure of the system
- Architecture is the structure, including the principles and guidelines governing their design and evolution over time
- Architecture is components and connectors

Why Is Architecture Important?

- **Architecture is the vehicle for stakeholder communication**
- **Architecture manifests the earliest set of design decisions**
  - Constraints on implementation
  - Dictates organizational structure
  - Inhibits or enable quality attributes
- **Architecture is a transferable abstraction of a system**
  - Product lines share a common architecture
  - Allows for template-based development
  - Basis for training

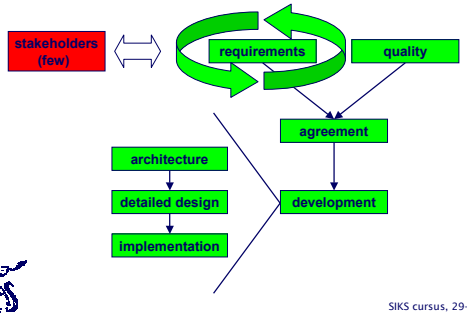
Classic development view



Characteristics

- **Iteration mainly on functional requirements**
- **Few stakeholders involved**
- **No balancing of functional and quality requirements**

Adding architecture, the easy way



Why this is not the way to go

- **Software development does not work that way; and it never did**

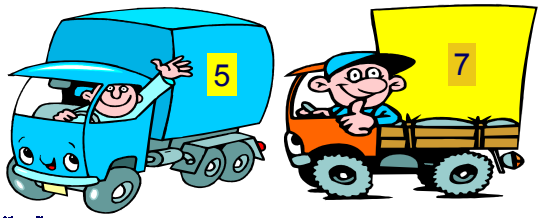
Activity versus phase

Activity \ Phase	Design	Implementation	Integration testing	Acceptance testing
Integration testing	4.7	43.4	26.1	25.8
Implementation (& unit testing)	6.9	70.3	15.9	6.9
Design	49.2	34.1	10.3	6.4

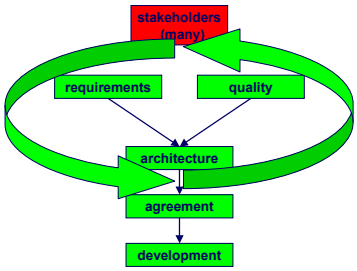
Why this is not the way to go

- **Software development does not work that way; and it never did**
- **Requirements and subsequent phases influence each other:**
  - You need to **see** how it works
  - COTS inclusion

Which car to take?



Architecture in the life cycle



Characteristics

- **Iteration on both functional and quality requirements**
- **Many stakeholders involved**
- **Balancing of functional and quality requirements**

Characteristics (cnt'd)

- **Architecture is about quality**
- **Requirements are negotiable**
- **Architecture is about making decisions, and coordinating those with all stakeholders**
- **Decisions can be evaluated at the architecture level**

Where did it start?

- 1992: Perry & Wolf
- 1987: J.A. Zachman; 1989: M. Shaw
- 1978/79: David Parnas, program families
- 1972 (1969): Edsger Dijkstra, program families
- 1969: I.P. Sharp @ NATO Software Engineering conference: "I think we have something in addition to software engineering [..] This is the subject of software architecture. Architecture is different from engineering."

Overview

- What is it, why bother?
- Issues only touched upon
  - Architectural styles & patterns
  - Product lines
  - Architecture design
- Architecture representations
- Architecture assessment

Architectural styles

- An architectural style is a description of component and connector types and a pattern of their runtime control and/or data transfer.
- Examples:
  - main program with subroutines
  - data abstraction
  - implicit invocation
  - pipes and filters
  - repository (blackboard)
  - layers of abstraction

Why architectures and patterns?

- from cognitive psychology: problem solving, chunking, programming plans
- Why is 'stack' a useful concept?
  - embodies concept useful in variety of settings
  - we have notations and mechanisms to support their use
  - we organize related related concepts in searchable networks
- Goal of architectures and patterns: identify and describe components at a still higher level of abstraction

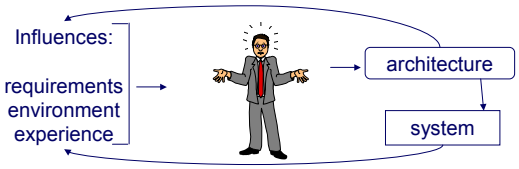
Software product lines

Set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way

Product line issues

- Scoping: what systems are in, what systems are out
- Identifying variation points
- Adoption: proactive or reactive

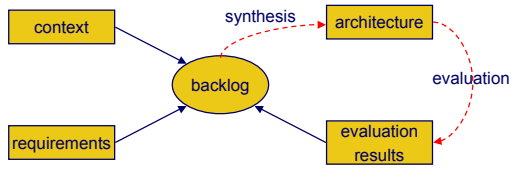
Architecture Business Cycle -- ABC



Architecture influences

- architecture affects structure of the developing organization
- architecture affects goals of the developing organization
- architecture affects requirements for the next system
- building a system affects architect's experience
- some systems/architectures are really influential

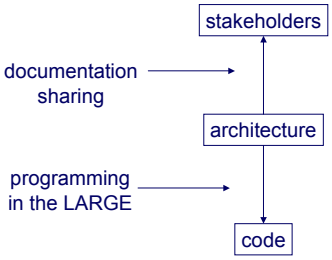
Global workflow in architecture design



Overview

- What is it, why bother?
- Issues only touched upon
  - Architectural styles & patterns
  - Product lines
  - Architecture design
- **Architecture representations**
- Architecture assessment

Architecture: two flavors



Characteristics (cnt'd)

- **Architecture is about quality**
- **Requirements are negotiable**
- **Architecture is about making decisions, and coordinating those with all stakeholders**
- **Decisions can be evaluated at the architecture level**

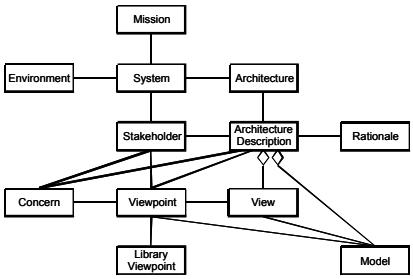
Analogy with building architecture

- Overall picture of building (client)
- Front view (client, “beauty” committee)
- Separate picture for water supply (plumber)
- Separate picture for electrical wiring (electrician)
- etc

Architecture representations in practice

- By and large two flavors:
  - Powerpoint slides – for managers, users, consultants, etc
  - UML diagrams, for technicians

IEEE model for architectural descriptions



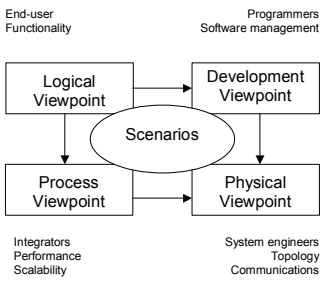
Some terms (from IEEE standard)

- System stakeholder: an individual, team, or organization (or classes hereof) with interests in, or concerns relative to, a system.
- View: a representation of a whole system from the perspective of a related set of concerns.
- Viewpoint: A viewpoint establishes the purposes and audience for a view and the techniques or methods employed in constructing a view.

View models

- Kruchten’s (now RUP) 4+1 model
- Siemens’ 4 views model
- Philips BAPO/CAFCR (5 views)
- Zachman (36 views)
- …

Kruchten’s 4+1 view model



Bass' set of architectural views (view = representation of a structure)

- **Module views**
  - Module is unit of implementation
  - Decomposition, uses, layered, class
- **Component and connector (C & C) views**
  - These are runtime elements
  - Process (communication), concurrency, shared data (repository), client-server
- **Allocation views**
  - Relationship between software elements and environment
  - Work assignment, deployment, implementation

Module views

- **Decomposition: units are related by "is a submodule of", larger modules are composed of smaller ones**
- **Uses: relation is "uses" (calls, passes information to, etc). Important for modifiability**
- **Layered is special case of uses, layer *n* can only use modules from layers <*n***
- **Class: generalization, relation "inherits from"**

Component and connector views

- **Process: units are processes, connected by communication or synchronization**
- **Concurrency: to determine opportunities for parallelism (connector = logical thread)**
- **Shared data: shows how data is produced and consumed**
- **Client-server: cooperating clients and servers**

Allocation views

- **Deployment: how software is assigned to hardware elements**
- **Implementation: how software is mapped onto file structures**
- **Work assignment: who is doing what**

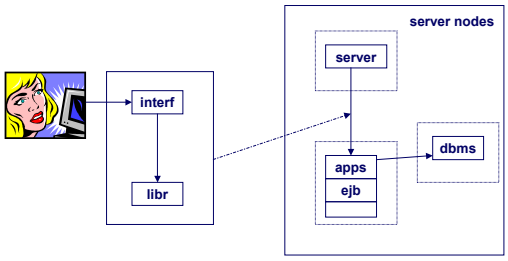
How to decide on which views

- **What are the stakeholders and their concerns?**
- **Which views address these concerns?**
- **Prioritize and possibly combine views**

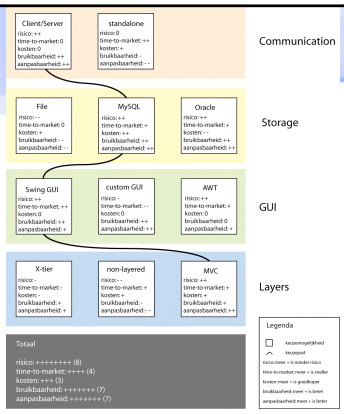
Problems/pitfalls in functional views

- **Poorly defined interfaces**
- **Poorly understood responsibilities**
- **Overloading the view**
- **Just drawing the picture**
- **Inappropriate level of detail**
- **"God" elements**

Example overloaded view



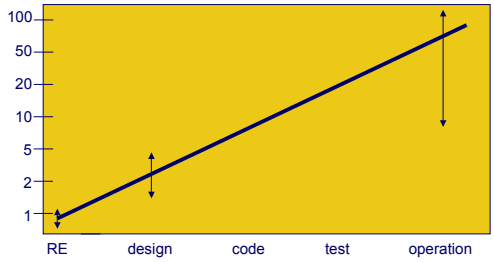
Decision visualization



Overview

- What is it, why bother?
- Issues only touched upon
  - Architectural styles & patterns
  - Product lines
  - Architecture design
- Architecture representations
- **Architecture assessment**

Relative cost of error correction



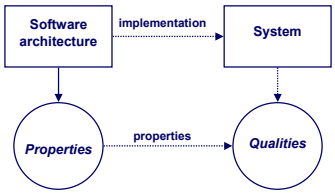
Architecture evaluation/analysis

- **Assess whether architecture meets certain quality goals, such as those w.r.t. maintainability, modifiability, reliability, performance**
- **Mind: the architecture is assessed, while we hope the results will hold for a system yet to be built**

Two kinds of questions

- Is this architecture suitable?
- Which of two or more architectures is the most suitable?

Software Architecture Analysis



Analysis techniques

- Questioning techniques: how does the system react to various situations; often make use of scenarios
- Measuring techniques: rely on quantitative measures; architecture metrics, simulation, etc

Scenarios in Architecture Analysis

- Different types of scenarios, e.g. use-cases, likely changes, stress situations, risks, far-into-the-future scenarios
- Which stakeholders to ask for scenarios?
- When do you have enough scenarios?

Architecture Tradeoff Analysis Method (ATAM)

- Reveals how well architecture satisfies quality goals, how well quality attributes interact, i.e. how they trade off
- Elicits business goals for system and its architecture
- Uses those goals and stakeholder participation to focus attention to key portions of the architecture

Benefits

- Financial gains
- Forced preparation
- Captured rationale
- Early detection of problems
- Validation of requirements
- Improved architecture

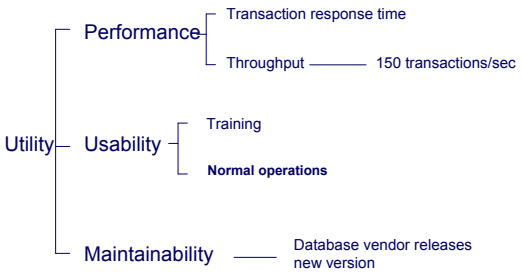
Participants in ATAM

- Evaluation team
- Decision makers
- Architecture stakeholders

Phases of ATAM

- Present method to stakeholders
- Present business drivers (by project manager of system)
- Present architecture (by lead architect)
- Identify architectural approaches/styles
- Generate quality attribute utility tree
- Analyze architectural approaches
- Brainstorm and prioritize scenarios
- Analyze architectural approaches
- Present results

Example Utility tree



Outputs of ATAM

- Concise presentation of the architecture
- Articulation of business goals
- Quality requirements expressed as set of scenarios
- Mapping of architectural decisions to quality requirements
- Set of sensitivity points and tradeoff points
- Set of risks, nonrisks, risk themes

Important concepts in ATAM

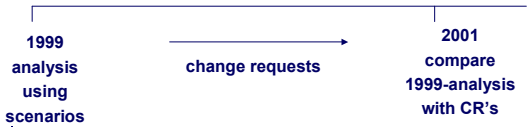
- Sensitivity point: decision critical for certain quality attribute
- Tradeoff point: decision that affects more than one quality attribute
- Risk: potential problem
- These concepts are overlapping

State of the practice

- Review process is mostly informal
- Techniques: experience-based reasoning, prototyping, scenarios, checklists
- Only 25% know about ATAM and SAAM
- Mostly done by internal people (architects and design team)
- Common stakeholders: architect, designer, manager, developer

How well can we predict changes?

- PhD research Nico Lassing, 1997-2001
- Theme: Architecture-Level Modifiability Analysis (ALMA)



Conclusions from this analysis

- Evolution is, to a large extent, unpredictable
- Certain changes concern complex components
- Analysis would improve if we explicitly challenge the requirements
- Fundamental modifiability-related decisions are sometimes not visible in viewpoints available

Summary

- A software architecture is important:
  - stakeholder communication
  - **early** evaluation of a design
  - transferable abstraction
- Software architecture is intertwined with requirements engineering
- Attention shifts from programming-in-the-large aspects to modeling of decisions

Further reading

- IEEE Software, March/April 2006, special issue on Software Architecture
  - Philippe Kruchten, Henk Obbink, Judith Stafford: The Past, Present, and Future of Software Architecture
  - Mary Shaw, Paul Clements: The Golden Age of Software Architecture