State of the Art of Software Architecture in Research and Industry

Michiel Perdeck, LogicaCMG
Michel Chaudron, TUE/SAN

- Experimental Physicist
- LogicaCMG
- Programmer, designer, software engineer (C, C++, Pascal, Ada, …)
- IT Architect
  - Banks, Industry, Public transport, Social Security
  - LogicaCMG Architecture Research Centre (FACE)
- Architecture-assessments
- Middleware (Information Bus)
- Service Oriented Architectures
- GRID (VL-E)
- Menselijke Maat (www.it4humans.org)
• System Architecture & Networking Group, TU Eindhoven
• Research in Software Architectures
  – Predicting system quality during architecture design
  – Component-based software engineering
  • Joint industrial projects (Robocop, Space4U, Trust4All) with Philips, Nokia, CSEM, Fagor/Ikerlan
– Empirical research
  • How are architectures designed in practice?
  • How are architectures represented?

Agenda
• Introduction to Software Architecture
  – Basic concepts, standards, terminology
  – Literature

• Software Architecture Research directions

• Issues and trends in Software Architecture practice
Questions…

• What is (a) “Software Architecture”?
• Why would one be interested in Software Architecture?
• How do you design a SA to fit its target environment?
• How do you assess a SA?
• How do you maintain a SA?
• How do you tell others about a SA?
• How do you describe a SA?
• How do you get support for a SA?
• How do you use and implement a SA?
• How do you keep an SA simple?

Main issue

How does Software Architecture add value to an organisation?
Software Architecture Books

(1)

- Software Architecture in Practice, Second Edition,
  L. Bass, P. Clements, R. Kazman,
  SEI Series in Software Engineering,
  Addison-Wesley, 2003

- Software Architecture: Perspectives on an Emerging Discipline
  Mary Shaw, David Garlan,
  242 pages, 1996, Prentice Hall

- Recommended Practice for Architectural Description,
  IEEE STD 1471-2000, 23 pages

(2)

- Evaluating Software Architecture
  (SEI Series in Software Engineering)
  Paul Clements, Rick Kazman, Marc Klein,
  386 pages, 2002, Addison-Wesley
  ISBN 020170482X

- Software Architecture: Organizational Principles and Patterns
  David M. Dikel, David Kane, James R. Wilson
  308 pages, 2001, Prentice-Hall,
  ISBN: 0-13-029032-7

- IT Architectures and Middleware (2nd edition)
  Chris Britton, Peter Bye
  384 pages, 2004, Addison-Wesley
  ISBN: 0321246942
The Importance of Architecture

10% of the lifecycle determines 90% of cost and risk

SW fault repair costs

10 20% 30% 30%

Architect Design Implement Testing Deployment Operation

Forces in Software

Functionality
Cost Compatibility
Capacity Fail safe
Availability Fault tolerance
Performance Throughput
Technology churn Resilience

The challenge over the next 20 years will not be speed or cost or performance; it will be a question of complexity.

Bill Redfield, Chief Strategy Officer, Sun Microsystems

Our enemy is complexity, and it’s our goal to kill it.

Jan Baan
Forces in Software Architecture

vision

organisation structure

culture

parts whole

information flow

strategy

individual

Source: Daan Rijsevrij, CapGemini

Business Promises of Software Architecture

• Reduce development cost
  – early assessment of system risks; validate early
  – improved communication between developers
  – earlier assessment of design alternatives and

• Reduce time-to-market
  – through enabling reuse and gradual evolution

• Reduce maintenance cost
  – through incorporation of foreseeable changes

• Improve product quality
  – increase fitness for use through stakeholder involvement;
  – reduce errors through enforcement of conceptual integrity
Development Promises of Software Architecture

- **Management of Complexity**
  - Define a model of a system that is intellectually manageable

- **Answering of what-if questions**
  - Allows stakeholders to evaluate alternative architectural solutions and their consequences

- **Feasibility study & risk analysis**
  - Analysis of various (non-)functional features of the future product; identification of possible problems during development, production & operation

- **Project estimation, planning & organization**
  - Allocation of components to concurrent teams

When to use Software Architecting?

- **When developing** a new system
  - Initially and to guide development

- **When changing** a system
  - if an architecture description is not available, or insufficient, as a basis for change
  - adapt the architecture documentation to changes

- **When integrating** existing systems
  - to ensure a consistent philosophy

- **For special communication needs**
  - to provide a common ground for understanding
  - architecture assessments
Stakeholders of IT Architectuur

- Clients
- Users
- Architects
- Company
- System
- Developers
- Project
- Operations

Stakeholders, Concerns, Viewpoints

Part of the IEEE 1471 model

**Stakeholders**
- user
- owner
- developer
- operator
- etc..

**Concerns**
- functionality
- performance
- maintainability
- scalability
- etc..

**Viewpoints**
- technical Architecture
- application landscape
- development
- systems management
- IT organisation
- etc..
**Customer:** Does it support my business goals?
- Schedule & budget estimation
- Feasibility and risk assessment
- Requirements traceability & progress tracking
- Product-line compatibility

**User:** Does it work properly?
- Future requirements growth accommodation
- Support of dependability & other X-abilities

**Service manager:** Can I support and defend it?
- Reliability, availability, maintainability

**System engineer:** Is it a quality system?
- Support of tradeoff analyses
- Completeness of architecture
- Consistency of architecture with requirements

**Developer:** Sufficient detail for design and development
- Workable framework for system construction,
  - e.g. selection/assembly of components & technologies
  - Resolution of development risks

**Maintainer:** Guidance on software modification
- Guidance on architecture evolution
- Interoperability with existent systems
**IEEE - P1471**

**standard for architecture description**

---

**Software Architecture definition 1**

**Definition 1**

The fundamental organization of a system embodied by its components, their relationships to each other and to the environment and the principles guiding its design and evolution

IEEE Standard P1471 Recommended Practice for Architectural Description of Software-Intensive Systems
**Definition 2**

Architecture of software is a collection of design decisions that are expensive to change.

*Alexander Ran, Nokia Research September 2001 European Conference on Software Engineering*

“The things that are fixed”

---

**Definition 3**

IT-Architecture is the collection of principles, guidelines and modeling standards that guide an organisation with the development, change and use of IT-resources within the whole organisation.

*M. Sikkema (ABN-AMRO) Ontwikkelen Onder Architectuur, Informatie, juni 2000*

Some bias towards enterprise systems.
Definition 4 (Chaudron)

- The set of design decisions that determine the quality properties of a system

(Design choices may affect different levels of abstraction.)

Definition 5 (Perdeck)

- Software Architecture is the discipline to create and change software systems that fit the organisations and individuals that use them.
Philippe Kruchten’s Definition

- Software architecture is not only concerned with structure and behaviour, but also with
  - usage
  - functionality
  - performance
  - resilience
  - reuse
  - comprehensibility
  - economic and technological constraints and tradeoffs
  - aesthetics

The Rational Unified Process -- An Introduction
Addison-Wesley, 1999.

IT-Architecture on different scales

Global architecture

Enterprise architecture

Systems architecture

Application architecture

Frameworks (class hierarchies)

Micro architecture

objects & classes

source:
Thomas Mowbray:
Corba Design Patterns
By the **business** of an enterprise is understood the **functional** perspective on the enterprise. It is about the **products** that are delivered to the environment, and about product and market **innovation**. IT has an **enabling** role in creating products.

By the **organization** of an enterprise is understood the **constructional** perspective on the enterprise. It is about the **processes** in which the products are produced, and about operational **performance**. IT is **exploited** for the performance of the processes.

---

**Functional view (What)**

**Functional perspective: black-box model**

- **the driver’s perspective**
  - **function**: (mathematical) relationship between input and output
  - **behavior**: the manifestation of the function in the course of time

**functional decomposition**

- car
  - lighting system
  - power system
  - steering system
  - brake system
**Constructional view (How)**

**Constructional perspective: white-box model**

*the mechanic’s perspective*

**construction**: (active) elements and their interaction relationships

**operation**: the manifestation of the construction in the course of time

**constructional composition**

- car
- chassis
- wheels
- motor
- lamps

---

**Architectural Dimensions**

**Functional dimensions**

- Structure
  - Functionality
  - Component interfaces
  - Component configurations

- Dependability
  - Performance
  - Timeliness
  - Reliability
  - Availability
  - Safety
  - Security
  - Robustness

- Composability
  - Interoperability
  - Reusability

- Local/Global Behavior
  - Control modes
  - Activities/Transactions
  - C&S protocols

- General
  - Scalability
  - Flexibility
  - Maintainability
  - Reusability
  - Openness
  - User-Friendliness
  - Cost
  - ...
IEEE - P1471
standard for architecture description

Mission
Environment
System
Architecture
Stakeholder
Architecture Description
Rationale
Concern
Viewpoint
View
Library Viewpoint
Model

stakeholders, concerns and viewpoints

Stakeholders
Customer, user
builders, Service managers

Concerns
suitable? usability?
durable? performance?
buildable? maintainable?

Viewpoints
Source: Daan Rijsenbrij, CapGemini
4+1 view model
(Philippe Kruchten - Rational)

End-user
Functionality

Programmers
Software management

Functional View

Development View

Scenarios

Process View

Physical View

Integrators
Performance
Scalability

System engineers
Topology
Communications

Hofmeister, Soni, Nord

mid-late 1990's
### Zachman Enterprise Architecture Framework

<table>
<thead>
<tr>
<th>Aspects</th>
<th>What</th>
<th>How</th>
<th>Where</th>
<th>Who</th>
<th>When</th>
<th>Why</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stakeholder</strong></td>
<td>Data</td>
<td>Function</td>
<td>Network</td>
<td>People</td>
<td>Time</td>
<td>Goals</td>
</tr>
<tr>
<td>Context scope</td>
<td>business things</td>
<td>business processes</td>
<td>locations</td>
<td>organisations and units</td>
<td>events</td>
<td>business goals</td>
</tr>
<tr>
<td>Concept</td>
<td>business level ERD</td>
<td>business process model</td>
<td>business location linkage</td>
<td>organisation and tasks</td>
<td>business event schedule</td>
<td>business strategy</td>
</tr>
<tr>
<td>Logical system</td>
<td>logical data model</td>
<td>application model</td>
<td>distribution model</td>
<td>human interaction model</td>
<td>systems event model</td>
<td>functional requirements</td>
</tr>
<tr>
<td>Technical</td>
<td>physical data model</td>
<td>system design</td>
<td>technical distribution architecture</td>
<td>user interface design</td>
<td>control structure</td>
<td>operational requirements</td>
</tr>
<tr>
<td>Components</td>
<td>data definition</td>
<td>program source code</td>
<td>network architecture</td>
<td>identities roles security</td>
<td>timing definitions</td>
<td>specifications</td>
</tr>
<tr>
<td>Operational</td>
<td>database admin</td>
<td>application admin</td>
<td>network admin</td>
<td>training</td>
<td>maintenance schedules</td>
<td>admin requirements</td>
</tr>
</tbody>
</table>

### LogicaCMG’s BASIC framework for change

- **Direction**
- **Diagnose**
- **Define**
- **Execute**
- **Evaluate**
- **Market etc.**
- **Products & Services**
- **Organisation**
- **People & Culture**
- **Process & Information Systems**
- **Technical Infrastructure**
Development of Research Focus

- Single system → product family
- structured notation
- multi-view notation (UML)
- principles (e.g. think client)
- design decisions & rationale
- Artefact oriented
  → value oriented
  economical aspects of Software Engineering

Research problems

- Multi-criteria evaluation of architectures
- Traceability (round-trip)
- Consistency between views
- Empirical research
  - politics dominates engineering
  - reverse engineering of views
- GRID architectures
  - Information services
  - Robustness
  - Security
  - Collaboration
Grid architecture

Virtual Organisations – Resource Sharing

Industrial Issues/Trends

- Reduction of Complexity
- Alignment
  - (Business – IT) (projects – architects) (development – serv mgmt)
- Legacy integration
  - Definition & untangling of services
- Service-oriented architecture
  - Granularity of services
  - Level of flexibility
  - Standardisation
- Model-driven architecture?
  - Use of CASE tools?
The Development Process, simple as this?

1. Concept
2. Business Process
3. Functions

Business Benefits

IPA Lente dag 2005-03-30 Software Architecture in Research and Industry

Two major forces

Complexity

Business Goals

Organisation
Culture
Conflicting interests

IPA Lente dag 2005-03-30 Software Architecture in Research and Industry
Culture’s consequences

- Certain IT solutions may not work because of the organisation culture and politics

Examples:
  - Enterprise Service Bus
    - Resistance to change, extra party on board
  - Configuration Management Database
    - Threat of position
  - Planning system
    - Organisation not ripe

How to architect?

The most important software engineering skills we must learn are:

- the skills involved in dealing with a **plurality of goals** which may be at odds with each other,

- the skills of coordinating the application of a **plurality of means**, each of which provides a varying degree of help or hindrance in achieving a given goal, depending on the situation.

B. Boehm, Software Engineering Economics
Architecting is a Wicked Problem

Some characteristics of Wicked Problems (Rittel & Webber)

- There is no definitive formulation of a Wicked Problem (W.P.)
  - A wicked problem is a set of interlocking issues and constraints which change over time, embedded in a dynamic social context.
  - The causes of a wicked problem can be explained in numerous ways. There are many stakeholders who have various and changing ideas about what might be a problem and what might be causing it.

- W.P.s do not have a well-described set of potential solutions.

- There is no immediate and no ultimate test of a solution to a W.P.
  - Solutions to wicked problems are not true-or-false but good-or-bad.

Dealing with Wicked Problems

- Adaptive Processes
  - Get stakeholder feedback early and often
  - Regular evaluation of risks and priorities
  - Alternate doing and thinking in order to continue making progress;
  - Make a 'map' of the problem to avoid ‘oscillation’

- Discuss the problem. Consensus emerges through the process of laying out alternative understandings of the problem, competing interests, priorities and constraints.
Forces pulling at the IT-Architect

Company
Long term

Architects
Content, Quality

Projects
Process, Short term

The Alignment Problem

Business
Architecture

align

Technical ICT
Architecture

align

Information
Architecture

align

PEOPLE
AND
POLITICS

align

Organisation’s Information System
<table>
<thead>
<tr>
<th>Level</th>
<th>Management</th>
<th>Definitions</th>
<th>Role Architect</th>
<th>Process</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Begraven</td>
<td>Improvement focus</td>
<td>Initiates improvements</td>
<td>Optimised</td>
<td>Effective</td>
</tr>
<tr>
<td></td>
<td>Embedded</td>
<td>Known and lived by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Beleid</td>
<td>Result focus</td>
<td>Responsible for result</td>
<td>Integrated</td>
<td>Consistent</td>
</tr>
<tr>
<td></td>
<td>Policy</td>
<td>Known and lived by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Beprooefd</td>
<td>Process focus</td>
<td>Responsible for process</td>
<td>Disciplines</td>
<td>Coordinated</td>
</tr>
<tr>
<td></td>
<td>Evaluated</td>
<td>Known and lived by</td>
<td></td>
<td>coordinated</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Belegd</td>
<td>Person focus</td>
<td>Workable drafts</td>
<td>Draft on discipline</td>
<td>Draft</td>
</tr>
<tr>
<td></td>
<td>Assigned</td>
<td></td>
<td></td>
<td>basis</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Bewust</td>
<td>Searching</td>
<td>Attempts</td>
<td>None</td>
<td>Hero</td>
</tr>
<tr>
<td></td>
<td>Aware</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>