Software Architecture

A business perspective
Mattijs van den Hoed
December 2014

Experience the commitment®

Introductions

- 25 years in the IT industry
  - Developer/designer
  - Technical manager
  - Solution Architect
- Solution architect at CGI
- Telecoms, Finance, Energy & Utilities
- Key knowledge areas
  - Systems integration
  - Contract Delivery
  - Central Market System
- High lights:
  - Telfort Mobile
  - KPN Wholesale
  - EnergyNet.DK

CGI

Goal

Provide an overview of what Software Architecture implies in the role as IT service provider.

Contents

- Software architecture
  - What are they?
  - Why are they important
  - What drives software architectures
- Role of the architect
- Quality attributes
  - How to define
  - Some examples
  - Case study

What is software architecture?

“The software architecture of a program or computing 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.”

- Software Engineering Institute

“Architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution”

- ANSI/IEEE Std 1471-2000, ISO 42010 Recommended Practice for Architectural Description of Software-Intensive Systems

Other definitions

"Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled."

- Eoin Woods, enterprise architect

"Architecture is that which is the hardest to change."

- Chris Verhoef, VU University

For more definitions, check out the SEI site:
http://www.sei.cmu.edu/architecture/definitions.html

Software Engineering Institute | Carnegie Mellon

CGI

Solution/software Architecture is not Enterprise Architecture

Enterprise Architect	**Solution Architect**
Advisor / Consultant	Authority
Building Bridges	Technical Decision Maker
Business/IT Alignment	Requirements ➔ Architecture
Governance over multiple “problems”	Single “Problem”
“City Planning”	“Building Design”
References:	References:
Zachman	SEI: ATAM, CBAM, QAW
TOGAF	Kruchten: 4+1 Views
DYA, IAF, GEM, BASIC,…	Fowler: Architectus Oryzus
IEEE 1471	IEEE 1471

Why is architecture important

Communication among stakeholders

- The architecture must support the goals of the stakeholders and this needs to be regularly aligned

Early design decisions

- Architecture defines constraints on implementation
- Architecture dictates the organizational structure
- Architecture inhibits or enables a system’s quality attributes
- Architecture supports the prediction of system qualities by study
- Architecture makes it easier to reason about and manage change
- Architecture enables more accurate cost and schedule estimates

Transferable abstraction of a system

- Software product lines share a common architecture
- Systems can be build using large, externally developed elements
- It pays to restrict the possible design alternatives
- Architecture permits template based development

Architecture structures and views

Module structures

- Decompositions
- Uses and layers
- Class diagrams

Component-connector structures

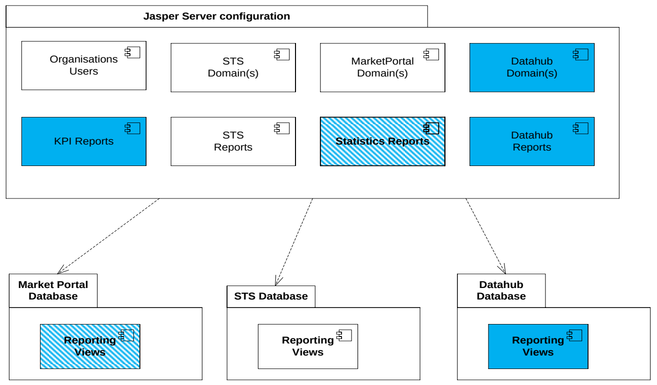
- Client server, Concurrency, Shared data
- Process
- Sequence diagrams, State diagrams

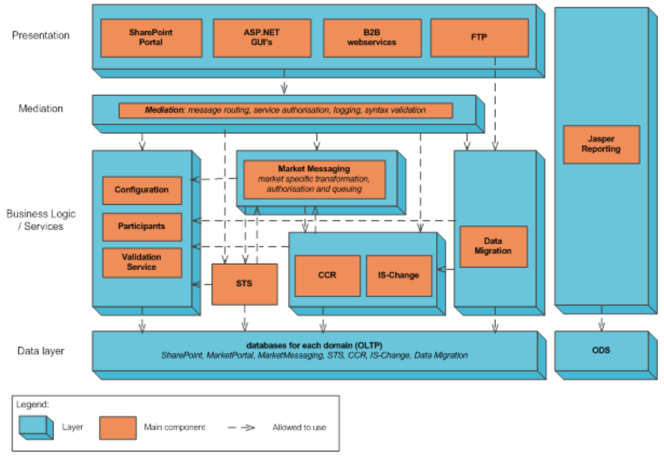
Allocation

- Work assignment
- Deployment
- Implementation

Module structures detail the development view of the system, its dependencies and data models

- **Decompositions** details a hierarchical structure of system in modules and sub-modules
  - Modules are often the products or production artefacts (code)
  - Often in diagrams or tables
  - Details localisation of changes for instance
- **Uses** views are important as these detail the dependencies for correct functioning of the modules.
  - Used modules are required for functioning
- **Layering** defines the different rules related to behaviour and communication between layers
- **Class diagrams** document the data structures used in the system
  - Supports reasoning behaviour and reuse





Presentation
SharePoint Portal
ASP.NET GUI's
B2B webservices
FTP
Mediation
Mediation: message routing, service authorisation, logging, syntax validation
Business Logic / Services
Configuration
Participants
Validation Service
Market Messaging market specific transformation, authorisation and queuing
STS
CCR
IS-Change
Data Migration
Jasper Reporting
Data layer
databases for each domain (OLTP) SharePoint, MarketPortal, MarketMessaging, STS, CCR, IS-Change, Data Migration
ODS
Legend:
Layer
Main component
Allowed to use

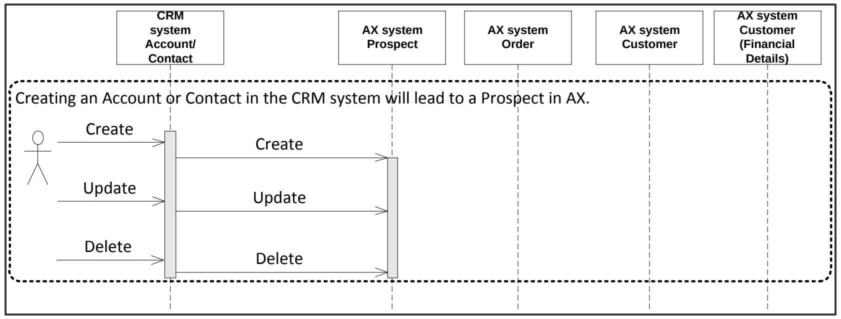
Component and connector views support the analysis of the runtime behaviour of the system

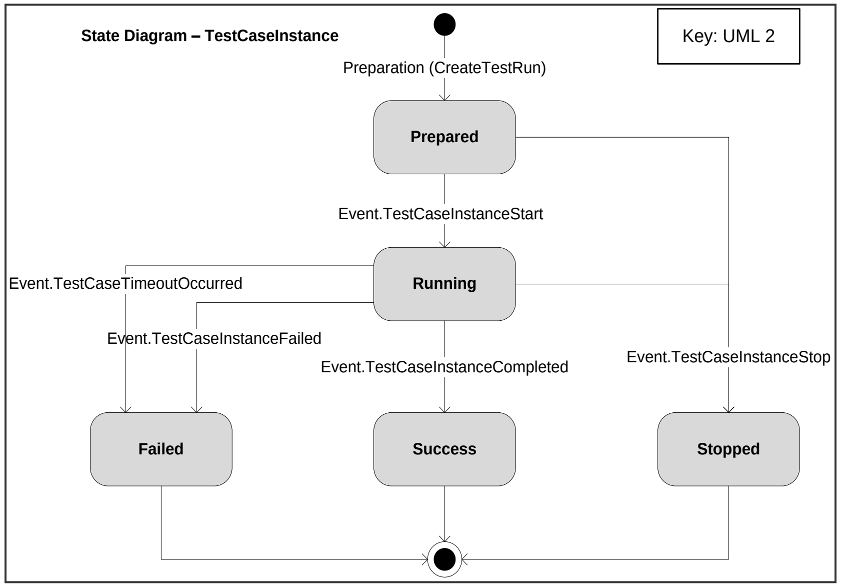
Analysis of runtime behaviour requires careful analysis of the required views may require specifically designed views. Common views include:

- **Sequence diagrams**, to model the runtime collaboration of components in time
- **State models**, to model behaviour of runtime components

The C&C views help analyse quality attributes such as:

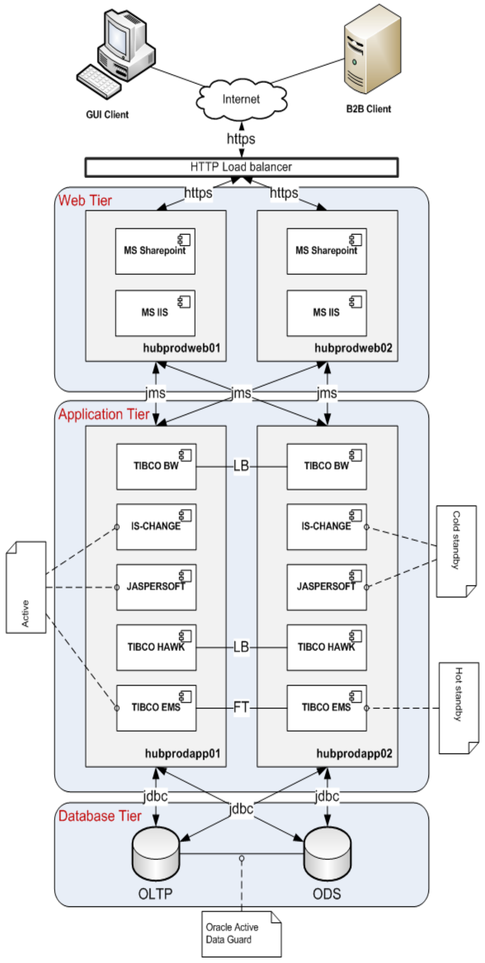
- Performance
- Availability
- Concurrency
- Resource usage


CRM system Account/ Contact
AX system Prospect
AX system Order
AX system Customer
AX system Customer (Financial Details)
Creating an Account or Contact in the CRM system will lead to a Prospect in AX.
Create
Create
Update
Update
Delete
Delete


State Diagram – TestCaseInstance
Key: UML 2
Preparation (CreateTestRun)
Prepared
Event.TestCaseInstanceStart
Running
Event.TestCaseTimeoutOccurred
Event.TestCaseInstanceFailed
Event.TestCaseInstanceCompleted
Event.TestCaseInstanceStop
Failed
Success
Stopped

Allocation view detail how the system sits in the external environment

- **Deployment** views show how the system is assigned to hardware resources and communication structures
  - Analysis and reasoning on performance, availability, etc.
- **Implementation** views detail how the software elements (modules, code, etc.) are mapped to file structures, configuration systems, etc.
  - Supports the effectiveness of the development and deployment processes
- **Work assignment**. This view details how the modules are assigned to development teams. Often called the Work Breakdown Structure (WBS)
  - Primary input to the project plan
  - Analysis of expertise required



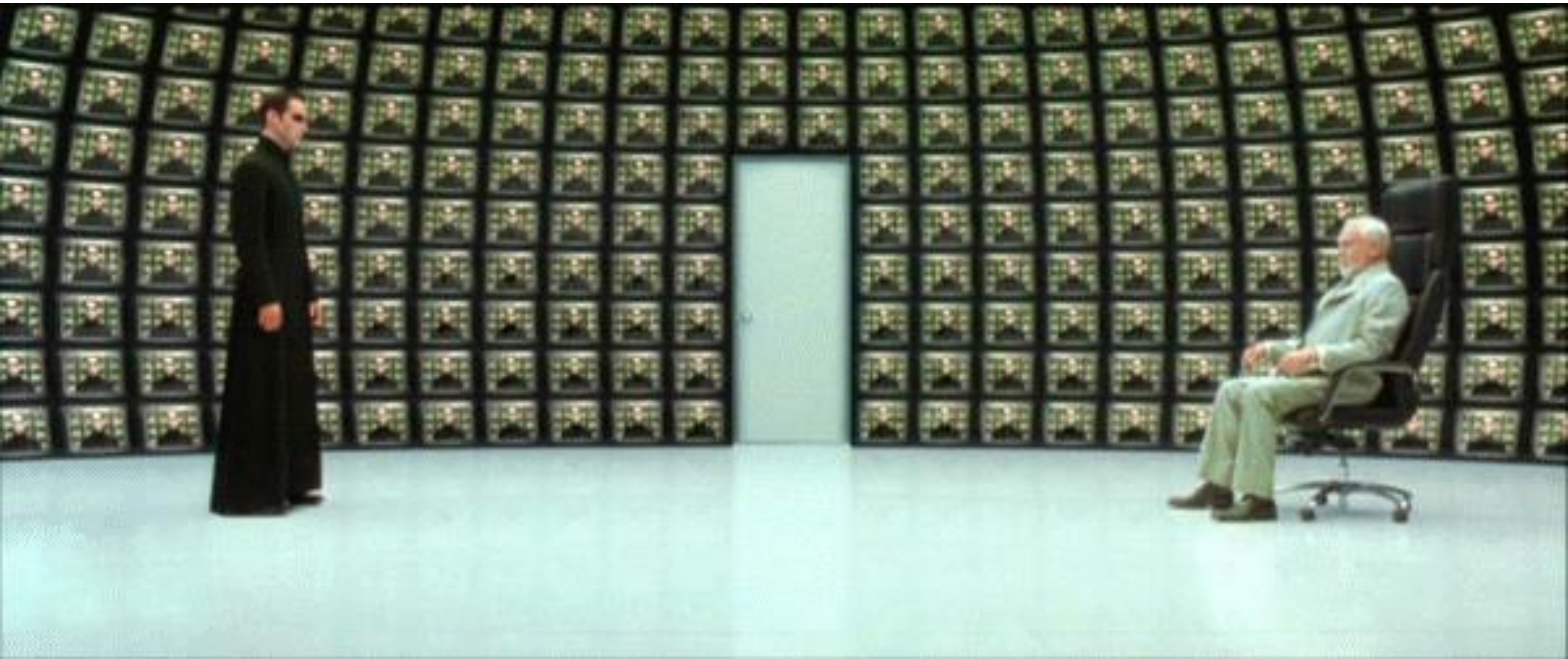
Role of the architect

Experience the commitment®

Common perception of ‘The Architect’

Question: What’s the difference between a terrorist and an architect?

Answer: You can negotiate with terrorists…



Key responsibilities of the architect

Create and communicate the vision that aligns to the requirements

- The vision includes the software architecture

Create and communicate the road to achieve that vision

- In terms of work packages and dependencies

Key responsibilities of the architect

Create and communicate the vision that aligns to the requirements

- The vision includes the software architecture

Create and communicate the road to achieve that vision

- In terms of work packages and dependencies

The Work Breakdown Structure is arguably the single most important deliverable of the Architect

Solution architect – Role responsibilities

Requirements analysis

- Review of requirements to determine how they can be met by a candidate architecture, products and technology

Solution development and optimisation

- Development of the most cost effective way to meet the requirements across the solution lifecycle

Project lifecycle analysis

- Analysis of the solution lifecycle across the project lifecycle and determining the best lifecycle/solution balance

Implementation strategy

- Determining the best Implementation strategy balancing the requirements of stakeholders, dependencies and continuity.

Cost benefit analysis

- Balance solution cost with winning price

Risk management

- Analysis and management of the risks introduced and covered by the solution and lifecycles

Business management

- Ensure that the proposed solution adheres to the requirements of the Business management system

Solution architect – skills

Requirements analysis

- Requirements management & development processes
- Defining scope based on requirements
- Mapping of requirements against candidate architectures and technologies
- Management of requirements across subcontractors

Solution development and optimisation

- Architecting skills
  - NFR handling
  - Functional decomposition
  - COTS selection and construction
  - Documenting architectures/solutions
  - Balancing methods and patterns against cost and capabilities
  - Manage and document design decisions
- Solution skills
  - Translate the scoped requirements and architecture into a solution
  - Scope and tweak solution to fit the winning price
  - Map the technical solution to project team and delivery requirements
- Delegate work to subteams and subcontractors and manage the result.

Solution architect – skills (continued)

Project lifecycle analysis

- Understand the pros and cons, risks and criteria of different delivery methodologies (sequential delivery, parallel delivery, RAD, Offshore/nearshore, time box vs. scoped deliverable, etc.)
- Project lifecycle support processes: configuration management, test and test requirements for the solutions, environments

Development strategy

- Solution implementation phasing
- Dependency analysis
- Balancing parallel strategies

Cost benefit analysis

- Estimation capabilities and methodologies
- Understanding the concept of the winning price

Risk management

- Understanding of the risks involved in the above subjects
- Experience in implementing containment strategies
- Understanding of the impact of risks

Drivers for Software architectures

Experience the commitment®

What drives software architectures?

Functional requirements

- The software architecture must provide the platform to implement the functionality
- The software architecture is mostly concerned with the type of functions required and less with each individual requirement

Quality attributes or non functional requirements

- These should always be seen in relation to the functional requirements; they define the quality of the functional requirement.
- Each quality attribute separately is of importance to the software architecture

Non-Functional Requirements – some examples

performance

availability

scalability

usability

security

portability

maintainability

extensibility

safety criticality

Non-Functional Requirements: Models and Standards

ISO 9126

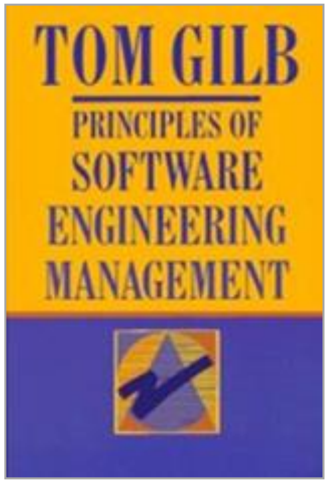
NFR Framework (Chung)

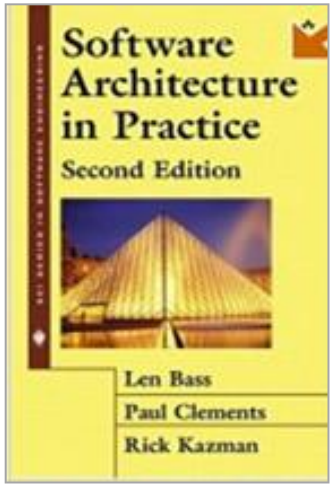
Attribute Specification (Gilb)

Attribute-Driven Design (Software Engineering Institute)

Non-Functional Decomposition (Poort & de With)

NON-FUNCTIONAL REQUIREMENTS IN SOFTWARE ENGINEERING
Lawrence Chung
Brian A. Nixon
Eric Yu
John Mylopoulos
Kluwer Academic Publishers





Resolving Requirement Conflicts through Non-Functional Decomposition

Eltjo R. Poort

Peter H.N. de With

Abstract

[illegible]

1. Introduction

[illegible]

Non-Functional Requirements – some observations

Overlooked

Studiously avoided

Left until it's too late

Poorly understood

Difficult to test:

- sometimes impractically so

Over-specified (usually by customers)

Under-specified (usually by suppliers)

A source of much angst for customers and suppliers alike:

- conflict!!!

Non-Functional Requirements - some trends

Customers are increasingly focused on NFRs:

- possibly due to outsourcing (SLA) pressure
- punitive conditions for breach of SLA

Some recent requirements:

- 99.999%, 100% availability
- guaranteed zero data loss
- 100 percentile response times

…any comments?

The three most 'troublesome' NFRs:

- performance
- availability
- scalability

The Impact of Non-Functional Requirements

Architecturally disproportionately influential

- major driver for componentisation
- hard/impossible to change later on

Major influence on project parameters

- frequent cause for project failure
- the delivery scheme will be significantly influenced
- the estimates will be significantly influenced
- often the processes will be, too

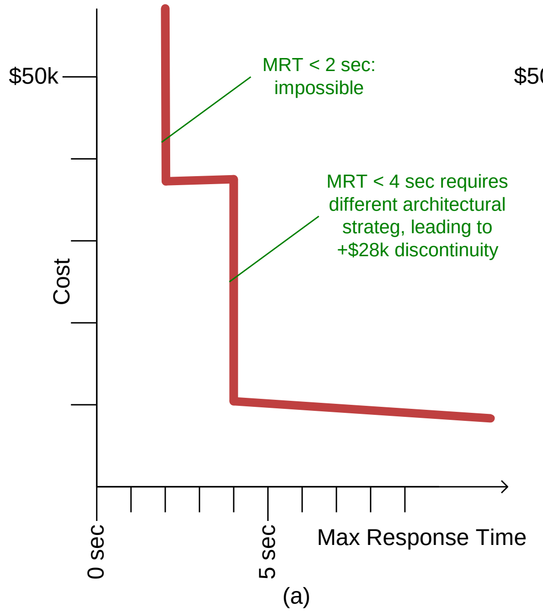
Strongly constrain acceptability

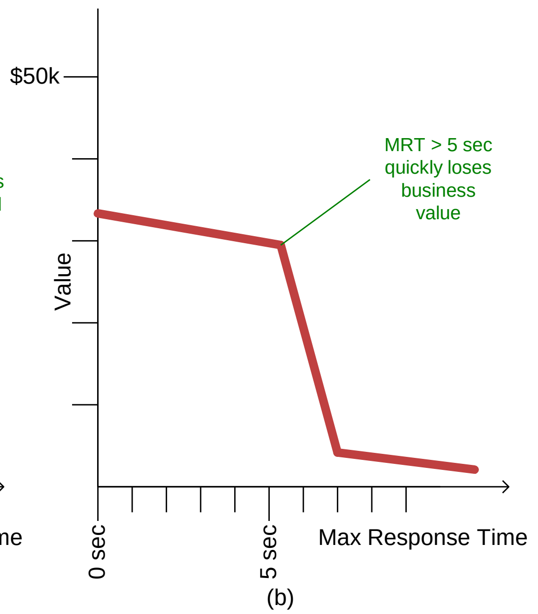
- even when not explicitly stated

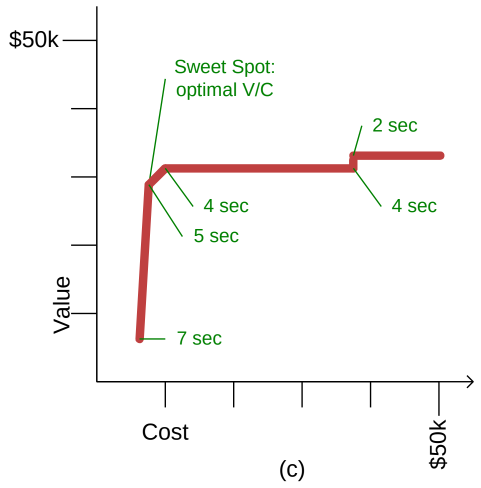
CGI

The Cost and Value of NFRs

- Architectural choices cause jumps in cost/attribute relationship
- Business value/attribute relationship is stakeholder dependent
- Balance of affordability: finding the sweet spot
  - Know everything
  - Elaborated architecture



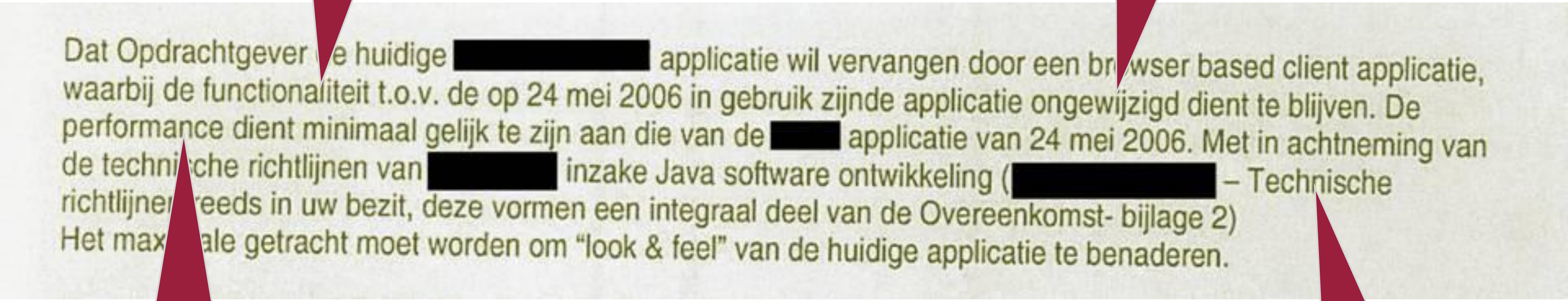




Real-life example 1: excerpt from Contract (page 1)

Implicit functional requirements

Always unfeasible requirement



Dat Opdrachtgever de huidige [redacted] applicatie wil vervangen door een browser based client applicatie, waarbij de functionaliteit t.o.v. de op 24 mei 2006 in gebruik zijnde applicatie ongewijzigd dient te blijven. De performance dient minimaal gelijk te zijn aan die van de [redacted] applicatie van 24 mei 2006. Met in achtneming van de technische richtlijnen van [redacted] inzake Java software ontwikkeling ([redacted] – Technische richtlijnen reeds in uw bezit, deze vormen een integraal deel van de Overeenkomst- bijlage 2)
Het maximale getracht moet worden om "look & feel" van de huidige applicatie te benaderen.

NFR guarantee

Customer dictated architectural solutions

Quality attributes

Source of Stimulus, entity that generates the stimulus

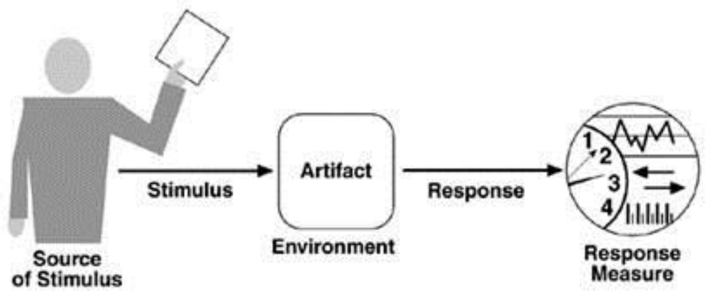
Stimulus, the condition arriving at the system

Environment, the condition on the system when the stimulus arrives

Artifact, the part of the system stimulated

Response, the effect of the artifact under the stimulus

Response measure, The response should be measureable such that the response can be tested.



Source: Software Engineering Institute

Performance examples

Attribute	Description
Source of stimulus	System
Stimulus	Initiation of Peek transaction
Environment	Normal operations (3 transactions per second average load)
Artefact	System
Response	Acknowledgement of receipt of transaction
Response measure	< 200msec, 95% < 1000msec, 99%

Our commitment to you

We approach every engagement with one objective in mind: to help clients succeed

CGI

Experience the commitment®