Multi-view Architecting

by Gerrit Muller, Jürgen Müller, Jan Gerben Wijnstra

Buskerud University College, Philips Research

e-mail: gaudisite@gmail.com

www.gaudisite.nl

Abstract

The development of large SW-intensive products needs to take requirements of multiple stakeholders into account. A design of such a system has to address functional and quality requirements adequately. However, for most of the required qualities no straight-forward design method exists even for a single quality.

A multi-view architecting model is described based upon a decomposition of an architecture in 5 architectural views, ranging from customer objectives to realization. It is the task of the architect to keep these views consistent and to balance design decisions in the perspective of the stakeholder needs.

We derived this model from our experience in developing software intensive industrial products, 2 cases are described from the medical domain.
Integrating 5 System Architecture Views

What does Customer need in Product and Why?

Customer What
Customer How

Application

Functional

Product What

Conceptual

Realization

Threads of Reasoning
Toplevel documentation structure  

Medical Imaging Workstation

Functional specifications

- FS cardio
- FS vascular
- FS dental

Design specifications

- design cardio
- design vascular
- design dental

Requirement analysis documents

- Typical cases
- Hazard analysis

Memory Resource Usage

CPU Resource Usage

Safety Design

Aspect designs

Multi-view Architecting

Gerrit Muller, Jürgen Müller, Jan Gerben Wijnstra

version: 1.0
March 6, 2013
ICAFCRdocumentationEV
## Memory usage aspect

**Medical Imaging Workstation**

<table>
<thead>
<tr>
<th>Application</th>
<th>Functional</th>
<th>Conceptual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders</td>
<td>Quality Attributes</td>
<td>Design Concerns</td>
</tr>
<tr>
<td>Clinical Technician</td>
<td>Response time</td>
<td>Memory Usage paging allocation fragmentation</td>
</tr>
<tr>
<td>Throughput</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Department manager</td>
<td>Processing Power usage</td>
<td>Bulk Memory management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fine grain memory management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Caching</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Processing Pipeline</td>
</tr>
</tbody>
</table>

---

Multi-view Architecting

Gerrit Muller, Jürgen Müller, Jan Gerben Wijnstra

version: 1.0

March 6, 2013

MVeasyVisionMemoryUsage
Multi-view Architecting

version: 1.0
March 6, 2013
ICAFCRissuesPerView

Gerrit Muller, Jürgen Müller, Jan Gerben Wijnstra

Issues per view

domain specific

keydrivers

context constraints

application domain entities & operations

stakeholder concerns

functional blocks SW components

functions features

system qualities

design concerns

technical structures and mechanisms

generic

Customer business

Application

Functional

Conceptual

Realization
Zooming in on relations

Customer business

Application
domain specific

keydrivers

realized by

application domain
tenities & operations

within

influences

determine

context constraints
generic

stakeholder concerns

with respect to

Multi-view Architecting

version: 1.0
March 6, 2013
ICAFCRzoomIn
Explicit facts and inherent details per view

- **Explicit specified**
- **Inherently present**

<table>
<thead>
<tr>
<th>Key drivers</th>
<th>Domain</th>
<th>Functions</th>
<th>Components</th>
<th>Construction elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key drivers</td>
<td>Domain</td>
<td>Functions</td>
<td>Components</td>
<td>Construction elements</td>
</tr>
<tr>
<td>Complementors</td>
<td>Entities</td>
<td>Qualities</td>
<td>Interfaces</td>
<td>statements</td>
</tr>
<tr>
<td>Competition</td>
<td>Operations</td>
<td>Values</td>
<td>Parameters</td>
<td>gates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>etcetera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer business</th>
<th>Application</th>
<th>Functional</th>
<th>Conceptual</th>
<th>Realization</th>
</tr>
</thead>
</table>

- Version: 1.0
- March 6, 2013
- ICAFCRlevelOfDetail

Multi-view Architecting

Gerrit Muller, Jürgen Müller, Jan Gerben Wijnstra
One thread of reasoning

Customer business
Application
Functional
Conceptual
Realization

domain specific

generic
Activities in multi-view architecting

3. Maintain Consistency
2. Determine Integrating Threads
1. Decompose

4. Balance Macroscopic and Microscopic
Criteria for thread selection

- Important for customer and the business
- Critical with respect to technical realization
Pitfalls in multi-view architecting

- too few views
- completeness
- general formalization
Continuous feedback during Product Creation

Business bottomline requirement

Compare

Product Creation

Measured result
Stepwise evolution
Qualities checklist

Safety
Security
Reliability
Robustness
Useability
Appeal, Appearance
Throughput or Productivity
Response Time
Image Quality
Reproduceability
Predicatability
Accuracy
Transportability
Wearability
Storability
Manufacturability
Testability
Serviceability
Configurability
Installability
Evolvability
Portability
Upgradeability
Extendability
Maintainability
Logistics flexibility
Lead time
Standards Compliance
Cost price
Cost of operation
Interaction with environment
Power consumption
Consumption rate (water, air, chemicals, etcetera)
Disposability
Size, weight
Resource utilization
SW aspects checklist

Granularity
- Scoping
- Containment
- Cohesion
- Coupling

Interfaces

Allocation
- Budgets

Information model
- Entities
- Relations
- Operations

Characteristics
- Static
- Dynamic

Configuration man.
- Packages
- Components
- Files
- Objects
- Modules
- Interfaces

Meta-functional
- Operational
  - Image processing
  - Handling calls
- Initialization
  - Start-up
  - Shutdown
  - Bootstrap
  - Discovery
  - Negotiation
- Fault handling
  - Exceptions
  - Logs
  - Traces
- Diagnostics
  - Configuration handling
  - Data replication
  - Performance observation
  - Capability query
- Testing
  - Automation
  - Special methods
  - Harness
  - Suites
- Off-line guidance

Supply chain
- Outsource
- Co-design
- Buy
- Interoperate
- Source vs binary

Technology choices
- Lifecycle
- Obsolescence
- Core, key, base

SW development
- Environment
- Repository
- Tools

Feedback tools
- Monitoring
- Statistics
- Analysis
- Call graphs
- Message tracing
- Object tracing

Synchronization
- Signalling
- Messaging
- Call-back scheduling
- Notification
- Active data
- Watchdogs
- Time-outs
- Locking
- Semaphores
- Transactions
- Checkpoints
- Deadlock detection
- Roll-back
- Priorities
- Pre-emption

Identification
- Uniqueness
- Naming
- Data model, registry
- Scoping
- Configuration
- Database
- Inheritance

Resource
- Management
  - Allocation
  - Anti-fragmentation
  - Garbage collection

Concurrency
- Processes
- Tasks
- Threads

Distribution
- Allocation
- Transparency
- Component
- Client/Server
- Multi-tier model

Persistence
- Caching
- Versioning
- Prefetching
- Lazy evaluation

Infrastructure

Multi-view Architecting
15 Gerrit Muller, Jürgen Müller, Jan Gerben Wijnstra
Medical Safety

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Quality Attributes</th>
<th>Design Concerns</th>
<th>Technical Structures and Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>vascular surgeon</td>
<td>reliable fluoroscopy</td>
<td>graceful degradation</td>
<td>independent parts</td>
</tr>
<tr>
<td>patient</td>
<td>radiation dose</td>
<td>error handling</td>
<td>distributed error handling</td>
</tr>
<tr>
<td>radiologist</td>
<td>mechanical safety</td>
<td>error prevention</td>
<td>self-tests</td>
</tr>
</tbody>
</table>

- graceful degradation
- error handling
- error prevention
- dead-man principle
- dose detection
- collision detectors
- power disconnection
IEEE 1471 model

System has Architecture

Architecture Description

Described by

Stakeholder has concern covers viewpoint

Consists of

S covers viewpoint

view conforms to 1

model defines 1
ISO 9126 quality framework

• **Functionality** suitability, accuracy, interoperability, compliance, security, *traceability*

• **Reliability** maturity, fault tolerance, recoverability, *availability*, *degradability*

• **Usability** understandability, learnability, operability, *explicitness*, *customisability*, *attractivity*, *clarity*, *helpfullness*, *user-friendliness*

• **Efficiency** time behaviour, resource behaviour

• **Maintainability**

• **Portability**