Abstract

The case study is evaluated: the resulting product and its design and the way the method has been used by the product creation team. The evaluation is done by means of the predefined hypothesis and criteria.

Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.
Hypothesis and criterions as basis for the evaluation

A rich collection of submethods fitting in a multi-view framework complemented with reasoning methods enables successful architecting of technology and software intensive complex systems in heterogeneous environments by means of generic insights grounded in specific facts.

1. product is a commercial success
2. product family is sustainable commercially successful
3. architects benefit from deploying submethods in multi-view framework
4. project leaders, product managers and engineers are able to use the outcome of the submethods
Evaluation of the product

**Customer objectives**
- ++ usability film layout
- ++ film efficiency
- + operator efficiency printing
- + ease of auto-printing
- - concurrent viewing and auto-printing

**Application**
- + throughput
- + image quality
- + interoperability URF
- - interoperability vascular

**Functional**
- + sales volume
- + selling price
- + margin
- + time to market
- - return on investment
- ~ network installation

**Operational feedback**
- + manufacturability
- + option handling
- ~ network installation

**Legend**
- + good or
- ++ very good
- ~ doubt
- - problem

---

Evaluation of the Architecting Method
3 Gerrit Muller

version: 0.9
June 21, 2020
AREproductEvaluation
### Evaluation of the design

<table>
<thead>
<tr>
<th>Conceptual</th>
<th>Realization</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ notification</td>
<td>+ processing pipeline</td>
</tr>
<tr>
<td>+ Objective-C</td>
<td>+ graphics</td>
</tr>
<tr>
<td>+ standard workstation</td>
<td>+ UI toolbox</td>
</tr>
<tr>
<td>+ X bypass</td>
<td>+ PMSnet</td>
</tr>
<tr>
<td>+ Unix</td>
<td>+ database engine</td>
</tr>
<tr>
<td>+ memory management</td>
<td>+ memory management</td>
</tr>
<tr>
<td>+ DB based communication</td>
<td>+ SW keys</td>
</tr>
<tr>
<td>+ OIT</td>
<td>+ OIT</td>
</tr>
</tbody>
</table>

- modularity
- distance internal and external information model
- some bloating due to over-genericity
- property handling
- dependency structure
- interface management

Lots of discussions about:
- language choice (why not C++)
- windowing system
- platform re-use

Based upon technology assessment in "Technology Improvement Plan"
### Coverage of submethods

<table>
<thead>
<tr>
<th>Customer objectives</th>
<th>Application</th>
<th>Functional</th>
<th>Conceptual</th>
<th>Realization</th>
</tr>
</thead>
<tbody>
<tr>
<td>key drivers</td>
<td>context diagram</td>
<td>case descriptions</td>
<td>construction</td>
<td>budget</td>
</tr>
<tr>
<td>value chain</td>
<td>stakeholders and concerns</td>
<td>commercial</td>
<td>decomposition</td>
<td>benchmarking performance analysis</td>
</tr>
<tr>
<td>business models</td>
<td>entity relationship models</td>
<td>service decomposition</td>
<td>functional decomposition</td>
<td>granularity determination</td>
</tr>
<tr>
<td>suppliers</td>
<td>dynamic models</td>
<td>goods flow decomposition</td>
<td>designing with multiple decompositions</td>
<td>value and cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td>function and feature specifications</td>
<td>execution architecture</td>
<td>safety analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>performance external interfaces standards</td>
<td>internal interfaces performance</td>
<td>reliability analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>standards</td>
<td>start up shutdown integration plan</td>
<td>security analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>legend</td>
<td>explicitly addressed</td>
<td>addressed only implicitly</td>
<td>not addressed</td>
<td></td>
</tr>
</tbody>
</table>

Coverage based on documentation status of first product release
Documentation of qualities in 1996

- **Usable**
  - Usability
  - Attractiveness
  - Responsiveness
  - Image quality
  - Wearability
  - Storability
  - Transportability

- **Interoperable**
  - Connectivity
  - 3rd party extendible

- **Serviceable**
  - Serviceability
  - Configurability
  - Installability

- **Ecological**
  - Ecological footprint
  - Contamination
  - Noise
  - Disposability

- **Liable**
  - Liability
  - Testability
  - Traceability
  - Standards compliance

- **Future proof**
  - Evolvability
  - Portability
  - Upgradeability
  - Extendibility
  - Maintainability

- **Down to earth attributes**
  - Cost price
  - Power consumption
  - Consumption rate (water, air, chemicals, et cetera)
  - Size, weight
  - Accuracy

- **Reliable**
  - Safety
  - Security
  - Reliability
  - Robustness
  - Integrity

- **Efficient**
  - Resource utilisation
  - Cost of ownership

- **Effective**
  - Throughput or productivity

- **Consistent**
  - Reproducibility
  - Predictability

- **Logistics friendly**
  - Manufacturability
  - Logistics flexibility
  - Lead time

Legend:
- *in separate document*
- *implicit in other documents*
Coverage profile of qualities

- **Coverage** based on amount of explicit and implicit available information in documentation;
- **Relevance** based on retrospective experience-based assessment.

Legend:
- **Coverage**: Completely covered, Very relevant, Relevant, Slightly relevant, Not relevant
- **Relevance**: Completely covered, Very relevant, Relevant, Slightly relevant, Not relevant
Users and usage of the results of the architecting method

**Results used by:**
- product management
- application
- project leaders
- engineers
- test engineers
- purchasing
- manufacturing
- suppliers

**Results used for:**
- detailed specifications
- testing
- communication
- derived documentation (manuals)
- used for succeeding products

**Engineers' critics**
- too abstract
- too late finalised

**Too late**
- project start
- product release

**Gap**
- system level
- submethods, qualities
- module level

Evaluation of the Architecting Method
8 Gerrit Muller

version: 0.9
June 21, 2020
AREmethodUsability
The conclusion of the case evaluation

1. product is a commercial success
   - + sales volume
   - + selling price
   - + margin
   - + time to market
   
   √ derived from Figure 18.3

2. product family is sustainable commercially successful
   - + 3 products
   - + 10 releases
   - in 5 years

   √ derived from Figure 14.10

3. architects benefit from deploying submethods in multi-view framework

<table>
<thead>
<tr>
<th>submethods</th>
<th>CA</th>
<th>F</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>qualities checklist</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>story telling</td>
<td>√</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   derived from sections
   18.4.1
   18.4.2
   18.4.3

   integration of the method
   18.4.4
   18.4.5

4. project leaders, product managers and engineers are able to use the outcome of the submethods

   results used by stakeholders for many purposes
   
   √ too late too abstract

   derived from Figure 18.7

version: 0.9
June 21, 2020
AREconclusions