How to Create a Manageable Platform Architecture?

by Gerrit Muller       Buskerud University College

e-mail: gaudisite@gmail.com

www.gaudisite.nl

Abstract

Today’s fast pace of the market and the technology development forces the product creators to rethink their development approach. One of the directions is to maximize the return on investments of frequently used functions, for instance by re-use, component based design or by a platform approach. The architecting effort is a key success factor to combine re-use approaches with fast and innovative product creation.

In this presentation we will present a case, discuss the role of the architecture, and elaborate the essential architecture ingredients for a successful platform creation, and evolution, and innovative product creation.
Q: How to manage platform architectures?

Recommendations

- case
- architecting
- platform

- market driven
- process
- time dimension
case:
- company overview
- time line
- technology innovations
context, product(s) and design

Q: How to manage platform architectures?

Recommendations

architecting
platform
market driven
process
time dimension

How to Create a Manageable Platform Architecture?
version: 1.0
March 6, 2013
Gerrit Muller
Philips Medical Systems, schematic organization

- Philips Medical Systems
  - Non X-ray modalities
    - US
    - MR
    - CT
  - Medical Imaging
  - Common X-ray Components
  - Conventional X-ray
    - Cardio Vascular
    - URF
    - Surgery

How to Create a Manageable Platform Architecture?
4 Gerrit Muller

version: 1.0
March 6, 2013
PMSorganization
Phases of Medical Imaging

1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

Advanced Development
("Common Viewing")

Development of 1st product

Parallel Development of 2nd product

Family Development

Transformation in re-useable components

Basic Application plus toolboxes

Easyvision RF

Easyvision RF R2

Easyvision CT/MR

Easyvision Xray R1

Easyvision CT/MR R2

Easyvision RAD

EasyReview

Medical Imaging Platform
Technology innovations by Common Viewing

- standard UNIX based workstation
- full SW implementation, more flexible
- object oriented design and implementation (Objective-C)
- graphical User Interface, with windows, mouse et cetera
- call back scheduling, fine-grained notification
- data base engine, fast, reliable and robust
- extensive set of toolboxes
- property based configuration
- multiple co-ordinate spaces
X-ray rooms from examination to reading around 1990

Examination Room | Control Room | Corridor or closet

Examination Room | Control Room | Reading Room
X-ray rooms with Medical Imaging applied as printserver

Examination Room

Control Room

Corridor or closet

Examination Room

Control Room

Reading Room

How to Create a Manageable Platform Architecture?
9 Gerrit Muller

version: 1.0
March 6, 2013
XRayRoomsPlusPrintServer
Comparison *screen copy vs optimized film*

old: screen copy

new: SW formatting

20 to 50% less film needed
How to Create a Manageable Platform Architecture?

11 Gerrit Muller

version: 1.0
March 6, 2013
HMPAswLayers1992
How to Create a Manageable Platform Architecture?

Gerrit Muller
Example CT/MR department

MR Examination room  Control room

CT Examination room  Control room

"MPR" room

Reading Room

How to Create a Manageable Platform Architecture?

version: 1.0
March 6, 2013
Gerrit Muller
How to Create a Manageable Platform Architecture?

Gerrit Muller

version: 1.0
March 6, 2013
MedicalImagingHealthcare
How to Create a Manageable Platform Architecture?

17    Gerrit Muller
Architecting:
- What is Architecture?
- Architecture vs Description
- My View on Architecture
- "Guiding How"
- The Art of Architecting
- More than Decomposition
What is Architecture?

How to Create a Manageable Platform Architecture?

Mark all applicable boxes

- specifications
- indicators
- high level rules
- concepts
- components (implementations)
- and
- infrastructure

- API's
- overarcheing vision
- guidance monitoring
- specifications
- indicators
- high level rules
- concepts
- components (implementations)
- infrastructure
- overarcheing vision
- guidance monitoring
- domain codification
- other...

- customers
- environment

- Java
- SQL
- FPGA
- technology

- OS
- file system
- bus
- memory
- I/O

- audio pipeline
- message routing
- white communication

- http
- DVB
- IPTV
- WMA
- MP3
- JPEG

- performance
- functionality
- risk
- reliability
- power
- cost

- layer n \( \downarrow \text{calls} \ n-k; \ k>1 \)
- layer n \( \uparrow \text{calls} \ n+k; \ k>0 \)

- market, business, technology
- process, product

- codification
- overarching vision
- guidance monitoring
- domain codification
- other...
Architecture vs Description

Architecture is a subset of which an architect is aware. It is then flattened into an architecture description, which is actually written by architect(s).
My View on Architecture

Understanding Why
Describing What
Guiding How

Do the right things

Do the things right
"Guiding How" by providing rules for:

1. Functional Decomposition
2. Construction Decomposition
3. Allocation
4. Infrastructure
5. Choice of integrating concepts

How to Create a Manageable Platform Architecture?

Gerrit Muller

version: 1.0
March 6, 2013
LWAArchitectureHow
The Art of Architecting

How to Create a Manageable Platform Architecture?

23     Gerrit Muller

version: 1.0
March 6, 2013
LWAArchitecting
Architecting is much more than Decomposition

Decomposition is "easy"
Integration is difficult
Platform:
- Why Platforms?
- What is a Platform?
- Platform Source Deliverables
- Example of Platform Efficiency
- Embedding Costs of Purchased SW

Q: How to manage platform architectures?

Recommendations

- case
- architecting
- market driven
- process
- time dimension

Platform:
- Why Platforms?
- What is a Platform?
- Platform Source Deliverables
- Example of Platform Efficiency
- Embedding Costs of Purchased SW
Why Platforms?

Customer value
- application adaptability
- availability variations
- new features originating from different products
- timely availability
- reliability

Internal benefits
- asset creation
- increase economy of scale

Extrovert driver
- availability of accumulated feature set
- design for configurability
- shared architectural framework
- quality increase
- predictability
- availability integrated base product
- maturity

Introvert driver
What is a Platform?

Huge product integration effort
Very flexible
Low coupling
Configuration management???

No product integration effort
Not flexible
High coupling
Configuration management

Product implementation

Applications + integration glue

Components

Infrastructure

P1
P2
P3

Pre-integrated platform

Common

Legend

Product
Platform
Platform Source Deliverables

development process  code  specifications
configuration management  development environment  documentation tools
infrastructure
And now in More Detail...

<table>
<thead>
<tr>
<th>development process</th>
<th>code</th>
<th>specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>test code &amp; data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>source code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>target OS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>purchased SW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>generation recipes</td>
</tr>
<tr>
<td>configuration management</td>
<td>development environment</td>
<td>documentation tools</td>
</tr>
<tr>
<td>code problem reports</td>
<td>code compiler, linker, ...</td>
<td></td>
</tr>
<tr>
<td>change requests documentation</td>
<td>dev. cluster OS meta data (review, metrics) customization</td>
<td></td>
</tr>
<tr>
<td>documentation</td>
<td>dev process support</td>
<td>word processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>drawing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spreadsheets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>publishing management</td>
</tr>
</tbody>
</table>

How to Create a Manageable Platform Architecture?

version: 1.0
March 6, 2013
HMPAplatformDeliverables
# Example of Platform Efficiency

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Number of inputs (a.o. modalities)</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>52</td>
<td>62</td>
<td>72</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>People per application</td>
<td>13</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Purchased SW Requires Embedding

How to Create a Manageable Platform Architecture?

version: 1.0
March 6, 2013
HMPAembedding

31 Gerrit Muller
Embedding Costs of Purchased SW

- Installation
- Configuration
- Customization
- Start up, shutdown
- Specifications
- Interface to application SW
- Exception handling
- Resource allocation and monitoring provision
- Resource tuning, see above
- Safety design
- Security design

- System monitor
- Error propagation
- Logging
- CPU
- Memory
- Disk

- Use of appropriate low level mechanisms
- Match to high level mechanisms:
  - Notification, scheduling
  - Job requests, subscriptions

- Add semantics level

- Functional
- System design
- Sw design

- Installation
- Configuration
- Customization
- Start up, shutdown
- Specifications
- Interface to application SW

How to Create a Manageable Platform Architecture?
Gerrit Muller
version: 1.0
March 6, 2013
Example of Embedding Problems

Architectural mismatch:
- wrappers, translators, conflicting controls
- Poor performance; additional resource usage
- additional code and complexity, no added value

Problems
- Architecture
- Reuse
- non problem

How to Create a Manageable Platform Architecture?

Gerrit Muller

version: 1.0
March 6, 2013
ARmergeProblems
Q: How to manage platform architectures?

Recommendations

- Who is First: Platform or Product?
- Platform Stability
- The First Time Right?
- Evolution of Easyvision Platform
- Lifecycle Differences
Who is First: Platform or Product?

How to Create a Manageable Platform Architecture?

version: 1.0
March 6, 2013
HMPAReleaseModel
Myth: Platforms are Stable

How **stable** is a platform or an architecture?

Dynamic Market

Fast changing Technology
The First Time Right?

Many years 1 10 100 1000

First time right? maybe unlikely miracle impossible
Feedback

stepsize: 3 months
elapsed time: 25 months
stepsize:
elapsed time

3 months
25 months

Target

2 months
12 months

Target

Start

Start
Small feedback cycles result in Faster Time to Market

1991

Growth

Change

Growth

3rd generation components are mature, active maintenance needed.
Growth and change continues, some "old" components become obsolete

1992

1994

1996

Last changed in:

1991

1992

1994
Lifecycle Differences

- **problem response**
  - 3 months
- **clinical prototype**
- **procedural change**
- **legislation change**
- **workstation useful life**
  - 10 years
- **MR scanner useful life**

- **commodity hardware and software**
- **minor SW release**
- **major SW release**
- **new generation of magnets gradients detectors**

42 Gerrit Muller
<table>
<thead>
<tr>
<th>information handling</th>
<th>archiving</th>
</tr>
</thead>
<tbody>
<tr>
<td>entirely distributed</td>
<td>service business</td>
</tr>
<tr>
<td>wide variation due to &quot;socio-geographics&quot;:</td>
<td>not health care specific</td>
</tr>
<tr>
<td>psycho-social, political, cultural factors</td>
<td>extreme robust</td>
</tr>
<tr>
<td>imaging and treatment</td>
<td>fire, earthquake, flood proof</td>
</tr>
<tr>
<td>localised</td>
<td>life time</td>
</tr>
<tr>
<td>patient focus</td>
<td>100 yrs (human life)</td>
</tr>
<tr>
<td>safety critical</td>
<td>not health care specific</td>
</tr>
<tr>
<td>limited variation due to &quot;nature&quot;:</td>
<td>short life-cycles</td>
</tr>
<tr>
<td>human anatomy</td>
<td>rapid innovation</td>
</tr>
<tr>
<td>pathologies</td>
<td>imaging physics</td>
</tr>
<tr>
<td>imaging physics</td>
<td>base technology</td>
</tr>
<tr>
<td>not health care specific</td>
<td>system technology</td>
</tr>
<tr>
<td>short life-cycles</td>
<td>rapid innovation</td>
</tr>
<tr>
<td>rapid innovation</td>
<td>entire distributed</td>
</tr>
</tbody>
</table>

How to Create a Manageable Platform Architecture?

Gerrit Muller

version: 1.0
March 6, 2013
MICAFReferenceModel
Process View:
- Simplified Process Decomposition
- Financial View on Process Decomposition
- Value and Feedback Flow
- Propagation Delay
- Sources of Failure
- Models for Generic Development

Q: How to manage platform architectures?

Recommendations

- case
- architecting
- platform
- time dimension
- process
- market driven
How to Create a Manageable Platform Architecture?

Simplified Process Decomposition

customer

supplying business

strategy
process

customer oriented (sales, service, production) process

product creation process

people, process and technology management process

version: 1.0
March 6, 2013
RSPprocessDecomposition
Modified Simplified Process Decomposition

How to Create a Manageable Platform Architecture?

Gerrit Muller

version: 1.0
March 6, 2013

SWRprocessDecompositionFamily
Financial View on Process Decomposition

How to Create a Manageable Platform Architecture?

Gerrit Muller

version: 1.0
March 6, 2013

SWRprocessDecompositionFamilyByValue
Value and Feedback Flow

Philips business

policy and planning

customer oriented process
(sales, service, production)

PCP

create generic components

people and technology management process
How to Create a Manageable Platform Architecture?

version: 1.0
March 6, 2013
GDPropagationDelay

Gerrit Muller
## Sources of Failure in Platform Developments

<table>
<thead>
<tr>
<th>Technical</th>
<th>Process/People/Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Too generic</td>
<td>• Forced cooperation</td>
</tr>
<tr>
<td>• Innovation stops (stable interfaces)</td>
<td>• Time platform feature to market</td>
</tr>
<tr>
<td>• Vulnerability</td>
<td>• Unrealistic expectations</td>
</tr>
<tr>
<td></td>
<td>• Distance platform developer to customer</td>
</tr>
<tr>
<td></td>
<td>• No marketing ownership</td>
</tr>
<tr>
<td></td>
<td>• Bureaucratic process (no flexibility)</td>
</tr>
<tr>
<td></td>
<td>• New employees, knowledge dilution</td>
</tr>
<tr>
<td></td>
<td>• Underestimation of platform support</td>
</tr>
<tr>
<td></td>
<td>• Overstretching of product scope</td>
</tr>
<tr>
<td></td>
<td>• Nonmanagement, organizational scope increase</td>
</tr>
<tr>
<td></td>
<td>• Underestimation of integration</td>
</tr>
<tr>
<td></td>
<td>• Component/platform determines business policy</td>
</tr>
<tr>
<td></td>
<td>• Subcritical investment</td>
</tr>
</tbody>
</table>

How to Create a Manageable Platform Architecture?

50   Gerrit Muller
Models for Platform Development

How to Create a Manageable Platform Architecture?

51  Gerrit Muller
Market Driven:
- The “CAFCR” model
- Example Platform Scoping
- Customer Key Drivers

Q: How to manage platform architectures?

Recommendations
The “CAFCR” model

What does Customer need in Product and Why?

Customer What
Customer How
Product What
Product How

C = Customer objectives
A = Application
F = Functional
C = Conceptual
R = Realization

drives, justifies, needs
enables, supports

How to Create a Manageable Platform Architecture?
53    Gerrit Muller
Five viewpoints for an architecture

What does Customer need in Product and Why?

Customer What

C - Customer objectives

Customer How

A - Application

Product What

F - Functional

Conceptual

R - Realization

Objective driven

intention

context understanding

opportunity

constraint awareness

know how based

How to Create a Manageable Platform Architecture?

version: 1.0
March 6, 2013
AMOintegratingCAFCR

Gerrit Muller
Example Platform Scoping

heterogeneous domains and application

intelligent buildings
motorway management
railway stations
airport terminals

shared core technology
Closed Circuit TV
audio broadcasting
access control
networking
Customer Key Drivers Motorway Management

Key-drivers

Safety
- Reduce accident rates
- Enforce law
- Improve emergency response

Effective Flow
- Reduce delay due to accident
- Improve average speed
- Improve total network throughput
- Optimize road surface
- Speed up target groups
- Anticipate on future traffic condition

Smooth Operation
- Ensure traceability
- Ensure proper alarm handling
- Ensure system health and fault indication

Environment
- Reduce emissions

Derived application drivers

Early hazard detection with warning and signaling
- Maintain safe road condition
  - Classify and track dangerous goods vehicles
  - Detect and warn noncompliant vehicles
  - Enforce speed compliance
  - Enforce red light compliance
  - Enforce weight compliance

Requirements

- Automatic upstream accident detection
- Weather condition dependent control
- Traffic speed and density measurement
- Cameras
- Deicing
- Traffic condition dependent speed control

Note: the graph is only partially elaborated for application drivers and requirements
Finally All Design Decisions are Related to Market

Philips operational view (manufacturing, service, sales)

Customer objectives: diagnostic quality
Application: image quality
Functional: IQ spec
Conceptual: render engine
Realization

C

U"

Useable
diagnosis
effective
time efficient
efficient
economic sound

A

T

Application

throughput
purchase price
CoO

F

IQ spec
typical case
CPU budget

C

Moore’s law
memory budget

R

processing
library
pixel depth
memory limit

P'

M'

Realization

P

M

operational constraints

Profit margin
standard workstation

Philips operational view (manufacturing, service, sales)

Cost revisited in context of clinical needs and realization constraints;

note: original threads are significantly simplified
Recommendations

Q: How to manage platform architectures?

to create successful products
educate artful architects
accept heterogeneous solutions
implement agile lifecycle decoupling
stimulate evolution, and continuous refactoring

Recommendations

identify key drivers
maintain focused scope
ensure market and business feedback

Q: How to manage platform architectures?

educate artful architects
to create successful products
accept heterogeneous solutions

实施敏捷生命周期解耦

stimulate evolution, and continuous refactoring