Abstract
The Platform and Evolvability course discusses the approach to achieve Evolvable Product Families. Prerequisites for this course are Systems Architecing and Multi-Objective System Architecting and Design, because we start from the assumption that we know how to design and architect individual systems. In this course we address how to harvest synergy and its consequences. We also add the time dimension: markets, customers, stakeholders and technologies are all changing around us, while we architect the next generation product family.
Abstract
The course Platforms and Evolvability addresses the architecting of evolvable product families based on a common platform.

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.
Prerequisites for Evolvable Product Family Architectures

- **mono-disciplinary engineering**
- **mono-system design**
- **mono-system architecting**
- **multi-disciplinary abstraction synthesis**
- **repositories**
- **baselines**
- **change procedures**
- **release management**
- **quality assurance**
- **product family architecting**
- **value propositions**
- **market segmentation**
- **portfolio management**
- **synergy**
- **user context**
- **product life cycle**
- **stakeholders**
- **evolvable product family architecting**
- **responsiveness**
- **anticipated trends**
- **disruptive change**
- **multi-site**
- **multi-supplier**
- **multi-vendor**
## Program

<table>
<thead>
<tr>
<th>1 Why &amp; What Evolvable Product Families</th>
</tr>
</thead>
<tbody>
<tr>
<td>exercise:</td>
</tr>
<tr>
<td>identify products in family</td>
</tr>
<tr>
<td>identify platform boundary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 Market analysis</th>
<th>(stakeholders&amp;concerns, market segments, key drivers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>exercise:</td>
<td></td>
</tr>
<tr>
<td>take 2 most distant products</td>
<td></td>
</tr>
<tr>
<td>make key driver graph, one for each product</td>
<td></td>
</tr>
<tr>
<td>identify tensions in interests</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Engineering &amp; Design</th>
<th>(repositories, configuration management, testing, configurability, resource management, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>exercise:</td>
<td></td>
</tr>
<tr>
<td>show repository structure and quantify</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 Process &amp; People</th>
<th>(development lifecycle, product lifecycle, goods flow, supply chain, creation chain, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>exercise:</td>
<td></td>
</tr>
<tr>
<td>make map of processes &amp; people involved; be specific (names) and quantify</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5 Reference architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>exercise:</td>
</tr>
<tr>
<td>make top 3 views</td>
</tr>
<tr>
<td>identify next 7 views</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6 Assessment &amp; Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>exercise:</td>
</tr>
<tr>
<td>define 3 change cases</td>
</tr>
<tr>
<td>determine impact of 1 change case</td>
</tr>
</tbody>
</table>
1 Why & What Evolvable Product Families

exercise:
- identify products in family
- identify platform boundary
Abstract
Product lines or product families are used to serve a broad market with a limited
development investment. In theory this is easily said, in practice managing product
lines effectively turns out to be significant challenge. In this paper we clarify
when platform strategies towards product lines make sense. Crucial for success
is scoping of product line and the shared assets.

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.

July 24, 2014
status: planned
version: 0

logo TBD
Multiple markets:
- different customers
- different applications
- different products

Electron microscopes:
- Material sciences
- Life sciences
- Manufacturing, e.g. semiconductors

Shared platform:
- Shared concepts
- Shared technology

Electron microscopes:
- E-beam sources, optics
- Vacuum
- Acquisition control
Complementing Systems for Same Market

Single market:
- different stakeholders
- different applications
- interoperable products

health care, e.g. cardiology:
- analysis
- diagnosis
- treatment
- administration

Shared components:
- shared concepts
- shared technology

health care, e.g. cardiology:
- patient support
- patient information
- image information
- storage & communication
- user interface
market segmentation

Customer
What
Customer
How
Product
What

C
Customer objectives
A
Application
F
Functional

market taxonomy
customer classification
stakeholder classification
inventarization applications
functions
features
performance

synergy analysis

Product
How

C
Conceptual
R
Realization

shared functionality
analyse characteristics
analyse differentiators
functionality
characteristics

©2006, Gerrit Muller
Roadmapping: Impact of Future Future

**market**

- **Customer What**
- **Customer How**
- **Product What**

- Commercial
- Financial
- Legal
- Social
- Economical
- Domain specific
  - Example: Clinical
- Managerial

**technology**

- **Product How**
- **Conceptual**
- **Realization**

Trends from gradual to disruptive paradigm transition
Criteria and Forces for Synergy

unification

development cost
development effort
logistics cost

market share
time to market
installed base evolution
future (potential) value
market approach
(luminary sites, price fighter)

fit to customer
fit to stakeholder
fit to application

dedication
Possible Levels of Sharing

**intangible assets**
- vision, objectives
- specifications, interfaces
- designs, concepts

**tangible assets**
- realized components
- integrated (sub)systems
- test suites
- processes
- tools
- infrastructure

*Not everything that can be shared should be shared!*
Reuse is needed ... as part of the solution

<table>
<thead>
<tr>
<th>trends</th>
<th>consequences</th>
<th>solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>features</td>
<td>feature interaction</td>
<td>new methods</td>
</tr>
<tr>
<td>performance</td>
<td>interaction</td>
<td>new tools</td>
</tr>
<tr>
<td>expectations</td>
<td>complexity</td>
<td>hardware performance</td>
</tr>
<tr>
<td>number of products</td>
<td>amount of software</td>
<td>new software technology</td>
</tr>
<tr>
<td>release cycle time</td>
<td>integration effort</td>
<td>new standards</td>
</tr>
<tr>
<td>years → months</td>
<td>reliability</td>
<td>reuse</td>
</tr>
<tr>
<td>openness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interoperability</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract
Systems evolve from mostly mechanical or physical devices into multi-disciplinary integrated systems. This evolution takes years or decades. The evolution occurs simultaneously with changes in the markets and in the organization. We describe this evolution and illustrate it with a X-ray systems and wafersteppers.
Evolution of X-ray Systems


- **1980**: Evolution of X-ray Systems starts with electro-mechanical components and autonomous subsystems.
- **1985**: The focus shifts towards vascular systems, introducing synergy and system control.
- **1990**: Cardiovascular systems emerge, showcasing the integration of multiple components.
- **1995**: The concept of a system of systems is introduced, highlighting the complexity and interconnectivity.
- **2000**: Evolution continues with an emphasis on interfaces and PC's in cardiography and radiology.
- **2005**: The timeline concludes with an overview of the advancements in the field, including the integration of PC's and monitoring systems.

From Autonomous Subsystems to Integrated System

©2006, Embedded Systems Institute

version: 0.1
July 24, 2014
FAxrayTimeline

©2006, Gerrit Muller
..~1980
many independent modules most Philips, some 3rd party
sales: all configurations are possible
system integration (SI) in factory
many adaption boxes
SI is mostly electro mechanical
innovation elapsed time many years  (f.i., 10 years for new imaging chain)
Organization in 1980

innovation departments
Roentgen Electronics Laboratory
Mechanical Electronics Laboratory
Physics Technical Laboratory

facilitating departments: drawing office; construction office; workshops
Staff in 1980

small teams

3 key persons:
  application
  senior designer
  cardiologist (outside Philips)

application and domain technology implicit in most staff

staffing mostly domain technology driven
..~1985
autonomous subsystems: Geo Acquisition Imaging X-ray generation

sales: preferred configurations; arbitrary configurations are more expensive
system integration (SI) in R&D
   SW in all subsystems
   SI is electro mechanical and configuration parameters
innovation elapsed time several years (f.i., 2 years for digital imaging chain)
Organization in 1985: Product/Business Oriented

<table>
<thead>
<tr>
<th>mammography</th>
<th>surgery</th>
</tr>
</thead>
<tbody>
<tr>
<td>radiography</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Geo
- Acquisition
- Imaging
- X-ray generation

Most products:
- Successful application oriented
- Little synergy or commonality
- Struggling with software
medium sized teams

strong subsystem focus

software depends on few good SW engineers (often with HW background)

project leader is also system designer

significant System Integration effort
Synergy drive ca 1990

Cardio and Vascular are merged. Digital imaging gets dominant

<table>
<thead>
<tr>
<th>surgery</th>
</tr>
</thead>
<tbody>
<tr>
<td>digital radiography</td>
</tr>
<tr>
<td>radiography</td>
</tr>
</tbody>
</table>

**Legend**

- Geo
- Acquisition
- Imaging
- X-ray generation
Geographical locations in 1990

**Germany**
- Factory and warehouse
- R&D
- Marketing
- Sales
- Management
- Factory
- Warehouse
- Road

**USA**
- R&D
- Marketing
- Sales

**Netherlands**
- Factory and warehouse
- R&D
- Marketing
- Management
- Corporate management

From Autonomous Subsystems to Integrated System
©2006, Embedded Systems Institute
July 24, 2014
version: 0.1
FAlxray1990buildings
Staff in 1990

- matrix organizations within product groups:
  - mechanical
  - electrical
  - software

- application and domain technology know how diluted

- software content is significant

- test and validation time is significant (> 1 year)

- senior designer ~= system designer
Common X-ray components (imaging, generation, collimators)
Common digital infrastructure (workstations, networks, printers)

URF

CT scanner

communication standard

DICOM

printer

workstation

cardio/vascular

MR scanner

kV
mA
kV
mA
Common components are organized as separate groups:
X-ray and PMS-wide

- cardio/vascular
- URF
- RAD
- Surgery

X-ray component suppliers
- Imaging
- X-ray generation

common digital infrastructure supplier
- HIS
- RIS
- viewing
- archiving
- communication
2000: Introduction of central System Control

New: system control = industrial PC + Windows XP + 4 Mloc + 3rd party SW

From Autonomous Subsystems to Integrated System
©2006, Embedded Systems Institute
28 ©2006, Gerrit Muller
version: 0.1
July 24, 2014
FAlxray2000system
Catheterization Laboratory integrates many systems
and is heavily connected to other health care departments and systems.

MR scanner
CT scanner
Cardiology
Radiology
Hospital
PACS
Information System
PC's

Catheterization lab

kV
mA
monitoring
ultrasound
tip positioning
monitoring
### Characterization per Phase

<table>
<thead>
<tr>
<th>Phase</th>
<th>Electro-Mechanical Components</th>
<th>Autonomous Subsystems</th>
<th>Synergy</th>
<th>System Control</th>
<th>System of Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emerging R&amp;D Integration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dominant Concern</td>
<td>Modularity</td>
<td>Configuration</td>
<td>Synergy</td>
<td>Synergy</td>
<td>Market Value</td>
</tr>
<tr>
<td>Staff</td>
<td>All Round</td>
<td>All Round</td>
<td>Disciplines M, E, I + System</td>
<td>Disciplines M, E, I + System</td>
<td>Disciplines M, E, I + System</td>
</tr>
<tr>
<td>Organization</td>
<td>Domain Labs</td>
<td>Products Subsystems</td>
<td>Matrix</td>
<td>Layered Matrix</td>
<td>+ Network</td>
</tr>
<tr>
<td>Size R&amp;D</td>
<td>Tens</td>
<td>Hundred</td>
<td>Several Hundred</td>
<td>Hundreds</td>
<td></td>
</tr>
</tbody>
</table>
Control Hierarchy of a Waferstepper

- Laser
- Illuminator
- Lens
- Measurement
- C&T
- Reticle Stage
- Reticle Handler
- Wafer Stage
- Wafer Handler

Coordination

Vertical Motion
Horizontal Motion

Ethernet

VME

Version: 0.1
July 24, 2014
Frequency of Control Actions

trend with increasing performance requirements

- \(10^{-3}\) seconds for SW sampling per die
- 1 second for per wafer
- \(10^3\) seconds for per batch
- \(10^6\) seconds for preventive maintenance per day
Evolution of System Control

user interface

automation interface

production and installation support

monitoring and optimization

metro

job control

exposure control

data management

dynamic calibration

infrastructure

feedforward

monitoring

1990

150 kloc

2000

2000 kloc
Consequences of Evolution

- **Complexity**: loss of overview (150kloc fits in 1 mind, 2Mloc not) (more than?) exponential increase of coupling
  - 1:1 relation HW:SW becomes n:m relation

- **Reliability**: threatens paradigm shift!

- **Performance and functionality demands**
  - autonomous subsystems
  - integrated system

From Autonomous Subsystems to Integrated System

version: 0.1
July 24, 2014
FAIevolutionConsequences
2 Market analysis (stakeholders&concerns, market segments, key drivers)

exercise:
- take 2 most distant products
- make key driver graph, one for each product
- identify tensions in interests
Module Platform Business Analysis

by Gerrit Muller    Buskerud University College

  e-mail: gaudisite@gmail.com

  www.gaudisite.nl

Abstract
This module provides an approach to analyse market and business to help in defining the platform scope.
<table>
<thead>
<tr>
<th>Approach to Platform Business Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>explore markets, customers, products and technologies</td>
</tr>
<tr>
<td>study one customer and product</td>
</tr>
<tr>
<td>make map of customers and market segments</td>
</tr>
<tr>
<td>identify product features and technology components</td>
</tr>
<tr>
<td>make maps:</td>
</tr>
<tr>
<td>market segments - customer key drivers</td>
</tr>
<tr>
<td>customer key drivers - features</td>
</tr>
<tr>
<td>features - products</td>
</tr>
<tr>
<td>products - components</td>
</tr>
<tr>
<td>determine value of features</td>
</tr>
<tr>
<td>identify synergy and (potential) conflicts</td>
</tr>
<tr>
<td>create roadmap and short term plan</td>
</tr>
</tbody>
</table>
Explore Markets, Customers, Products and Technologies

market segments  customers  products  technology

Asian country  Asian city  African  US private  US social  EU
Won Lan  JJ express  Pretoria  Johnson
express  national  USA
EU  Columbia  EU

成本 体积 交通 速度
质量

P1800  P1900  P2200  P2600
基本 送电
喂料
高频 喂料
送电

1800k/hr  2100k/hr  2100k/hr  3000k/hr

P1800

喂料

brain storm and discuss time-boxed
Study one Customer and Product

What does Customer need in Product and Why?

Customer

What

Customer

How

Product

What

Product

How

What does Customer need in Product and Why?

Customer objectives

Application

Functional

Conceptual

Realization

Key drivers

Derived application drivers

Requirements

Early hazard detection

with warning & signalling

Maintain safe road condition

Classify and track dangerous goods vehicles

Detect and warn non-compliant vehicles

Enforce speed compliance

Enforce weight compliance

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

Smooth Operation

- Anticipate on future traffic condition

- Ensure Traceability

- Ensure prop

- Ensure sys

Environment

- Reduce emissions

key-driver graph

configuration

functional model

physical model

Which is only partially elaborated on drivers and requirements

Module Platform Business Analysis
©2006, Embedded Systems Institute

version: 0.2
July 24, 2014

MPBAProductMarket
Make Map of Customers and Market Segments

- **Functionality**: P2600
- **Performance**: P2200
- **Mature**: P1900
- **Niche**: P2000

Many changes and variations
identify product features and technology components
Mapping From Markets to Components

market segments

features

components

1. cost, power, traffic, volume
2. basic
3. 1800k/hr, 2100k/hr, 3000k/hr
4. buffer

adjust, order, workflow
prepare, packing, process
browse, fast imaging
buffering, cooling, heating, cleaning, feeding
drivers, drivers, scheduler
store, conveyor
climate subsystem, handling subsystem

products

P1800
P1900
P2200

customer key drivers

market segments

changing cost, mature cost, mature
changing performing, performing

key drivers

1800k/hr, 2100k/hr, 3000k/hr

climate subsystem, handling subsystem

modules

CPU, RAM, etc.

software

file-system, networking

hardware

drivers, conveyor, store

control subsystem, feeding, bufferiing, heating, cooling, cleaning

Web services

execute, order

market segments

key drivers

P1800
P1900
P2200

changing cost, mature cost, mature
changing performing, performing

CPU, RAM, etc.

file-system, networking

drivers, conveyor, store

control subsystem, feeding, bufferiing, heating, cooling, cleaning

execute, order

mapping
Example Criteria for Determining Value

- Value for the customer
- (dis)satisfaction level for the customer
- Selling value (How much is the customer willing to pay?)
- Level of differentiation w.r.t. the competition
- Impact on the market share
- Impact on the profit margin

Use relative scale, e.g. 1..5 1=low value, 5 -high value

Ask several knowledgeable people to score

Discussion provides insight  (don't fall in spreadsheet trap)
Determine Value of Features

<table>
<thead>
<tr>
<th>Features</th>
<th>P1800</th>
<th>P1900</th>
<th>P2200</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfaction</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>customer</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>sales price</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>market share</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- **feeder**: 1 5 4
- **hf feeder**: 4 3 4
- **buffer**: 4 3 4
- **sunpower**: 2 2 1

Module Platform Business Analysis
©2006, Embedded Systems Institute
©2006, Gerrit Muller

version: 0.2
July 24, 2014
PFproductFeatureMapWithNumbers
Example Platform Scoping

- Intelligent buildings
- Motorway management
- Railway stations
- Airport terminals

shared core technology
- Closed Circuit TV
- Audio broadcasting
- Access control
- Networking

heterogeneous domains and application
3 Engineering & Design  (repositories, configuration management, testing, configurability, resource management, ...)
exercise:
    show repository structure and quantify
What is a Platform?

huge product integration effort
very flexible
low coupling
configuration management???

no product integration effort
not flexible
high coupling
configuration management

<table>
<thead>
<tr>
<th>product implementation</th>
<th>applications + integration glue</th>
<th>applications + integration glue</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>concepts</td>
<td>components</td>
<td>components</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>infrastructure</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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</td>
<td>compiler, linker, ...</td>
</tr>
<tr>
<td>change requests</td>
<td>meta data (review, metrics)</td>
<td>dev. cluster OS</td>
</tr>
<tr>
<td>documentation</td>
<td>customization</td>
<td>dev process support</td>
</tr>
<tr>
<td>infrastructure</td>
<td>documentation</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>management</td>
</tr>
</tbody>
</table>
Who is First: Platform or Product?

platform baseline

platform as consolidation baseline
Abstract

Many products today are developed for highly dynamic markets while the products and functions get more and more integrated. The product and service realization is based on fast changing technologies that come together in complex value chains. The challenge for modern companies in innovative domains is to survive in this dynamic world.

In this paper we explore the contribution of architecting and standardization to the company success. We look at the *why*, *when*, *who* and *how* questions of standardization and at the role of architecting in the standardization process.

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.
Problem Statement

How to survive in innovative domains?
- Fast moving market
- Complex value chains
- Fast moving technology
- Increased integration
That is easy...

How to survive in innovative domains? fast moving market fast moving technology complex value chains increased integration

By being the fittest in your ecological (economical) niche!
1. employ skilled system architects

2. apply an agile system architecting process

3. determine the right subjects and moments for standardization

4. apply a sensible standardization process
How to survive in innovative domains?

standardization

what

why

how

when

who
How to survive in innovative domains?

**standardization**

- **what**
- **how**
- **who**
- **when**
- **why**
Classification of Standardization Tactics

system of systems
  provides
    system
      interoperates with
        component
          uses

+ provide choice to customer
+ compete on performance and functionality
+ enlarge application potential
+ customer value
+ focus on core value
+ use of commodity components
Focus on Core; not on Key or Base Technology?

- **Core**
  - Own value
  - IP
  - Critical for final performance
- **Key**
  - Technology life cycle
- **Base**
  - Commodity

Options for Total Product:
- **make**
- **outsource**
- **buy**
- **refer customer to 3rd party**
- **Partnering**
How to survive in innovative domains?

standardization

what

why

how

when

who
When to Standardize

too early  right moment  too late

problem is understood  
domain structure is clear  
broadening set of stakeholders  
technology is ripe  
requirements unknown  
technological compromises  
loss of competitive edge  
insufficient and uncertain facts  
wrong expectations  
intuition not calibrated  
caught in proprietary legacy  
poor interoperability  
customer demands standards  
focus on key i.s.o. core  
market does not take off  
(Metcalfe's law)
Roadmapping as Tool

Customer objectives: customer needs, expectations, trends
Application: Market

Functional: Products

Conceptual: Technology

Realization: People

standardization process, tactics, deployment

standardization concern
provides interoperability
provides interoperability
use of standards

time, ca 5 years
Purchased SW Requires Embedding

- Purchased software
- Proprietary software
- SW architecture
- Purchased OS

Architecting and Standardization
©2006, Embedded Systems Institute
©2006, Gerrit Muller

version: 1.0
July 24, 2014
HMPAembedding
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

functional
system design
sw design
add semantics level
use of appropriate low level mechanisms
match to high level mechanisms:
- notification, scheduling
- job requests, subscriptions

System monitor
Error propagation
Logging
CPU
Memory
Disk
Balance of Considerations and Trends

- Innovation from outside
- Focus on core technology
- Initial cost reduction
- Faster to market
- Interoperability
- Functional integration

---

- Transition cost
- Required know how
- Release propagation
- Integration effort
- Embedding
- Flexibility
- Resource use
- Performance
- License costs

---

Architecting and Standardization
©2006, Embedded Systems Institute

version: 1.0
July 24, 2014
ECMAbalance
Example of Lifecycle Reference Model

**information handling**
- entirely distributed
- *wide* variation due to "socio-geographics":
  - psycho-social,
  - political, cultural factors

**imaging and treatment**
- localised
- patient focus
- safety critical
- *limited* variation due to "nature":
  - human anatomy
  - pathologies
  - imaging physics

**image handling**
- distributed
- *limited* variation due to "nature":
  - human anatomy
  - pathologies
  - imaging physics

**base technology**
- not health care specific
- short life-cycles
- rapid innovation

**archiving**
- service business
- not health care specific
- extreme robust
- fire, earthquake, flood proof
- life time
- 100 yrs (human life)
Evolution from Proprietary to Standard

- high innovation rate
- global standardization takes more than 5 years
- high interoperability

Legend:
- Applications
- Product family
- Vendor
- World standard

Applications: CT, MRI, vascular, URF, medical imaging

Vendor: Siemens, GE, Philips

Standardization:
- ACR/NEMA
- DICOM
How to survive in innovative domains?

standardization

what
why
how
who
when
Standards describe what

black box (interface) level:
- protocols
- functions
- parameters
- formats
- behavior
- characteristics

white box (implementation) level:
- protocols
- functions
- parameters
- formats
- realizations
- limitations
- constraints
- opportunities
- behavior
- characteristics

standard → complying implementations →
Input from implementation know how

white box know how:

current and future realization:

- design choices
- technology capabilities
- domain concepts
- limitations
- constraints
- opportunities

what needs to be defined

- functions
- parameters
- formats
- protocols
- behavior
- characteristics

realism/acceptance level

- time
- effort
- cost
Towards a Standard

**market**
- needs
- expectations
- concerns

**black box level:**
- functions
- parameters
- formats
- protocols
- behavior
- characteristics

**white box know how:**
- current and future realization:
  - design choices
  - technology capabilities
  - domain concepts
  - limitations
  - constraints
  - opportunities

future proof; room for innovation
market enabler; room for added value
not locked into specific technology constraints
realistic and acceptable; time, cost, effort
What Should be in a Standard

**Standard: what**

requirements at conceptual level,

*no design or implementation*

as minimal as possible

ambitious but cautious

the minimal set of (interface) requirements to:

1) ensure interoperability
2) foster innovation and
3) maximise the room for added value.
Embedding in a Reference Architecture

reference architecture

context + system model:
  function allocation
  composition guidance
  emerging characteristics
  processes

is this a standard?

framework for

standards

conform to

implementations
How to survive in innovative domains?

**standardization**

- **why**
- **when**
- **who**
- **what**
- **how**
Flow of Standardization

**explore**
- market needs
- stakeholders (competitors, suppliers, partners, customers, ...)
- existing realizations
- implementation issues

**analyze**
- manage and facilitate
  (heterogeneous stakeholders, create support and acceptance)
- write and debate
  (scoping, negotiation)
- prototype and validate
- iterate

**standardize**
- decide
- publish
- provide reference implementation (optional)

**deploy**
- push
- manage compliance
- evolve standard
Who Contributes and Participates?

How to survive in innovative domains?

standardization

what

why

how

when

who
Internal Standardization Process == Highly Strategic!
Non technical aspects of standardization

- Legal, IP oriented:
  - Licenses
  - Patents
  - Copyright

- Political:
  - Decision power
  - Who is in control?
  - (Hidden) interests
  - Coalitions
  - Networks

- Business:
  - Value chains
  - Business models
  - Market development

- Social:
  - Privacy
  - Social value

Diagram:

- Standardization
  - Connects to legal, IP oriented
  - Connects to political
  - Connects to business
  - Connects to social
Architect and Standards: Love-Hate Relationship

**love**
- no worries: concerns are taken care of
- focus on core problems
- facilitates interoperability

**hate**
- limits innovation (harness)
- limits solution space
- simplistic management orders
Conclusions

**How to survive in innovative domains?**

- problem is understood
- domain structure is clear
- broadening set of stakeholders
- technology is ripe

<table>
<thead>
<tr>
<th>what</th>
<th>minimal, as little as possible requirements (not design or implementation) room for added value and innovation</th>
</tr>
</thead>
<tbody>
<tr>
<td>how</td>
<td>fast iteration make rationale explicit roadmapping</td>
</tr>
<tr>
<td>who</td>
<td>strategic insight technology know how market know how social and political insight ambitious but cautious</td>
</tr>
</tbody>
</table>

3. determine the right subjects and moments for **standardization**

4. apply a sensible **standardization** process

**standardization**
Integration
Decomposition is easy, integration is difficult
Nasty surprises show up during integration

Do you have any design issues for the design meeting?

The default answer is: No.

During integration numerous problems become visible

Architecting and Standardization
©2006, Embedded Systems Institute

version: 1.0
July 24, 2014
MSintegration
Architectural mismatch

Architectural mismatch:
wrappers, translators, conflicting controls

Additional code and complexity, no added value

Poor performance; additional resource usage

Problems

Architecture

Reuse
non problem

Duplication
Integrating concepts

1. functional decomposition
2. construction decomposition
3. allocation
4. infrastructure
5. choice of integrating concepts
Platform block diagram

Architecture guidelines

Test environment

Application Toolboxes
Hardware Abstraction
Hardware

Application Services
Infrastructure services
Base Product

Development support services

Product 1 specifics

Product 2 specifics

Product n specifics
Module 4

4 Process & People  (development lifecycle, product lifecycle, goods flow, supply chain, creation chain, ...)
exercise:
make map of processes & people involved; be specific (names) and quantify
Module Platform and Evolvability; Process and People

by Gerrit Muller    Buskerud University College

  e-mail: gaudisite@gmail.com
  www.gaudisite.nl

Abstract
This module provides processes and insights in people, processes and organization issues for evolvable platforms.
Abstract
Most products fit in a larger family of products. The members of such a product family share a lot of functionality and features. It is attractive to share implementations, designs et cetera between those members to increase the efficiency of the entire company.

In practice many difficulties pop up when product developments become coupled, due to the partial developments which are shared. This article discusses the advantages and disadvantages of a family approach based on shared developments and provides some methods to increase the chance on success.
Typical Examples of Generic Developments

Platform
Common components
Standard design
Framework
Family architecture
Generic aspects, functions, or features
Reuse
Products (in project environment)
### Claimed Advantages of Generic Developments

<table>
<thead>
<tr>
<th>Advantage</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced time to market</td>
<td>building on shared components</td>
</tr>
<tr>
<td>Reduced cost per function</td>
<td>build every function only once</td>
</tr>
<tr>
<td>Improved quality</td>
<td>maturing realization</td>
</tr>
<tr>
<td>Improved reliability</td>
<td></td>
</tr>
<tr>
<td>Improved predictability</td>
<td></td>
</tr>
<tr>
<td>Easier diversity management</td>
<td>modularity</td>
</tr>
<tr>
<td>Increases uniformity</td>
<td></td>
</tr>
<tr>
<td>Employees only have to understand one base system</td>
<td>less learning</td>
</tr>
<tr>
<td>Larger purchasing power</td>
<td>economy of scale</td>
</tr>
<tr>
<td>Means to consolidate knowledge</td>
<td></td>
</tr>
<tr>
<td>Increase added value</td>
<td>not reinventing existing functionality</td>
</tr>
<tr>
<td>Enables parallel developments of multiple products</td>
<td>product-to-product or project-to-project</td>
</tr>
</tbody>
</table>

"Free" feature propagation
Experiences with reuse, from counterproductive to effective

**bad**
- longer time to market
- high investments
- lots of maintenance
- poor quality
- poor reliability
- diversity is opposed
- lot of know how required
- predictable too late
- dependability
- knowledge dilution
- lack of market focus
- interference
- but integration required

**good**
- reduced time to market
- reduced investment
- reduced (shared) maintenance cost
- improved quality
- improved reliability
- easier diversity management
- understanding of one base system
- improved predictability
- larger purchasing power
- means to consolidate knowledge
- increase added value
- enables parallel developments
- free feature propagation
Successful examples of reuse

<table>
<thead>
<tr>
<th>Homogeneous Domain</th>
<th>Cath Lab</th>
<th>MRI</th>
<th>Television</th>
<th>Waferstepper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Dominated</td>
<td>Car</td>
<td>Airplane</td>
<td>Shaver</td>
<td>Television</td>
</tr>
<tr>
<td>Limited Scope</td>
<td>Audio Codec</td>
<td>Compression Library</td>
<td>Streaming Library</td>
<td></td>
</tr>
</tbody>
</table>

Product Families and Generic Aspects
©2006, Embedded Systems Institute
©2006, Gerrit Muller

version: 2.3
July 24, 2014
SWRsuccessful
Limits of successful reuse

struggle with integration/convergence with other domains

TV: digital networks and media
cath lab: US imaging, MRI

poor/slow response on paradigm shifts

TV: LCD screens
cath lab: image based acquisition control

software maintenance, configurations, integration, release

MRI: integration and test
wafersteppers: number of configurations

how to innovate?
Drivers for Generic Developments

Customer value
- application adaptability
- availability variations
- new features originating from different products
- timely availability
- reliability
- asset creation
  - increase economy of scale

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

Extrovert driver
Introvert driver

Product Families and Generic Aspects
©2006, Gerrit Muller
Granularity of generic developments shown in 2 dimensions

**Delegated integration**

**Shared integration**

actual integration level

---

component module subsystem platform system

---

EVM MIP CV Generator CCD flat detector

---

version: 2.3 July 24, 2014
Modified Operational Organization PCP

Product Families and Generic Aspects
©2006, Embedded Systems Institute
version: 2.3
July 24, 2014
GDoperationalOrganization
Sources of Failure in Generic Developments

Technical
- Too generic
- Innovation stops (stable interfaces)
- Vulnerability

Process/People/Organization
- Forced cooperation
- Time platform feature to market
- Unrealistic expectations
- Distance platform developer to customer
- No marketing ownership
- Bureaucratic process (no flexibility)
- New employees, knowledge dilution
- Underestimation of platform support
- Overstretching of product scope
- Nonmanagement, organizational scope increase
- Underestimation of integration
- Component/platform determines business policy
- Subcritical investment
Models for Generic Development

- **Customer**
  - Policy and planning
  - Customer oriented process (sales, service, production)
  - Product Creation Process
  - Create generic components
  - People and technology management process

- **Lead Customer**
  - Direct feedback
  - Too specific?

- **Carrier Product**
  - Product feedback
  - Product specific?

- **Platform**
  - Feedback problem
  - Too generic

- **Technology Push**
  - No feedback
Product Related Life Cycles

- Individual systems
- Service
- System creation
- Upgrades and options
- System production
- System sales
- Upgrades and options
- Production
- Sales
- Upgrades and options
- Disposal
Customer Oriented Process

Order Acquisition → Order Realization

Order Realization → Service Support

Information → Order

Order → Product

Service Support → $$

Material → Order

Order → $$

Customer-Oriented Process

version: 2.3
July 24, 2014
Impact of Procurement Duration

\[ \text{PD (Procurement Demand) ratio} = \frac{t_{\text{procurement}}}{t_{\text{demand}}} \]

If PD ratio < 1 then build on order
Else forecast based procurement

Less robust
Models for reuse

lead customer

carrier product

platform

technology push

advanced demanding

innovate for specific customer
refactor to extract generics

innovate for specific product
refactor to extract generics

innovate in generic platform
integrate in products

innovate in research laboratory
transfer to product development

good
direct feedback
too specific?

generic?
no feedback
bad
Use before reuse
Feedback

- stepsize: 3 months
- elapsed time: 25 months
Feedback (2)

steppsize: elapsed time

3 months 25 months

Target

2 months 12 months

Target

Start

Start
Small feedback cycles result in Faster Time to Market
Use = Validate before Reuse

Does it satisfy the needs?
- performance
- functionality
- user interface

Does it fit in the constraints?
- cost price
- effort

Does it fit in the design?
- architectural match
- no bloating

Is the quality sufficient?
- multiplication of problems
- or multiplication of benefits
5 Reference architecture
exercise:
make top 3 views
identify next 7 views
A Reference Architecture captures the essence of the architecture of a collection of systems. The purpose of a Reference Architecture is to provide guidance for the development of architectures for new versions of the system or extended systems and product families.

We provide guidelines for the content of a Reference Architecture and the process to create and maintain it. A Reference Architecture is created by capturing the essentials of existing architectures and by taking into account future needs and opportunities, ranging from specific technologies, to patterns to business models and market segments.

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.

July 24, 2014
status: preliminary
draft
version: 0.6
1. general introduction

2. level of abstraction

3. content

4. summary
Why Reference Architectures?

When to Use Reference Architectures?

What do Reference Architectures contain?

How to use Reference Architectures?

What are inputs of a Reference Architecture?

Criteria for a good Reference Architecture.
Increased complexity, scope, and size

Facilitate multi-site, multi-organization, multi-vendor, multi-* system creation and life-cycle support

Increased dynamics, integration

Effectively create new: products, product lines, product portfolio

Managing synergy

Providing guidance, e.g., architecture principles, best practices

Providing an architecture baseline and an architecture blueprint

Capturing and sharing (architectural) patterns

Providing a common lexicon and taxonomy

Providing a common (architectural) vision

Providing modularization and the complementary context

Articulation of domain and realization concepts

Explicit modeling of functions and qualities above systems level

Explicit decisions about compatibility, upgrade and interchangeability.
When to Use Reference Architectures

Reference Architectures facilitate the step towards product family architecting and evolvability; this often coincides with multi-* problems.

- evolvable product family architecting
- product family architecting
- mono-system architecting
- mono-system design
- mono-disciplinary engineering
- multi-site
- multi-supplier
- multi-vendor
- mono-disciplinary engineering
RA Elaborates Mission, Vision and Strategy

- **mission**
- **vision**
- **strategy**

Multiple organizations

Reference Architecture
guidance for future

elaborated in guidance for future
RA = Business Arch. + Technical Arch. + Customer Context

- Customer context
  - Customer enterprise
  - Users
  - Requirements
  - Black box view
  - Relations
  - Guidance

- Technical architecture
  - Design patterns
  - Technology

- Business architecture
  - Business model
  - Life cycle
Instantiation of a RA in few Transformations

reference architecture  → architect

family architecture  → constraints and opportunities

shared asset architecture  → extracting essentials

system architecture  → design and engineer

shared assets  → field feedback

system A  → build and test

system B  → actual systems

product family  → engineering documentation

architectures  → reference architecture

A Reference Architecture Primer
©2006, Gerrit Muller

version: 0.6
July 24, 2014
SAFRAInstantiation
Inputs of a Reference Architecture

- Existing architectures
- Essence
  - Architecture patterns
- Mining
- Proven concepts & known problems

- Customer needs
  - Business needs
- Exploration & analysis
- Product portfolio
  - Future requirements
- Vision

Reference Architecture

Guides evolution

Triggers new changes
Criteria for a good Reference Architecture

understandable for broad set of stakeholders

accessible and actually read/seen by majority of the organization

addresses the key issues of the specific domain

satisfactory quality

acceptable

up-to-date and maintainable

adds value to the business

customers

product managers

project managers

engineers
Challenge: Appropriate Level of Abstraction

Single System
Product Family in Context
Capturing the Essence
Size Considerations:
  What is the appropriate level of abstraction?
  How many details?
Decomposition of Large Documents
Level of Abstraction Single System

- Static system definition
- Multidisciplinary design
- System requirements

Number of details vs. level of abstraction:
- Monodisciplinary
- Multidisciplinary
Product Family in Context

- enterprise context
- enterprise
- stakeholders
- systems
- multidisciplinary design
- parts, connections, lines of code

Number of details:
- $10^9$
- $10^6$
- $10^3$
- $10^0$
- $10^{-3}$
- $10^{-6}$
- $10^{-9}$
RA: Capturing the Essence

A Reference Architecture Primer
©2006, Gerrit Muller

version: 0.6
July 24, 2014
RAPdiaboloRA
RA: level of abstraction, number of details

- compact reference architecture
  - few diagrams only

- extensive reference architecture
  - many documents

level of abstraction

number of details

$10^3$

$10^6$
Size Considerations

10^3 level of abstraction

10^6 number of details

- **compact**
  - reference architecture
  - few diagrams only

- **extensive**
  - reference architecture
  - many documents

**low effort to**
- create
- maintain
- read
- easy to share

**limited**
- guidance
- anchor value

**significant effort to**
- create
- maintain
- read
- difficult to share

**great**
- guidance
- anchor value
Decomposition of Large Documents

- **Document structure**:
  - Overview
  - Compound document
  - Recursion

- **Frontpage**:
  - Title
  - Identification
  - Author
  - Distribution
  - Status
  - Review

- **History changes**

- **Meta information**: max 2 pages

- **Contents**: 2..18 pages

- **Atomic document**: 1. aap
  2. noot
  3. mies
  lists
  and ca 50%
  text
What should be in Reference Architectures?

- Guidance from Best Practices
- Visualizations
- Structure

What content should be in Reference Architectures?
1.1 One of several prerequisites for architecture creative synthesis is the definition of 5-7 specific key drivers that are critical for success, along with the rationale behind the selection of these items.

2.1. The essence of a system can be captured in about 10 models/views.

2.2. A diversity of architecture descriptions and models is needed: languages, schemata and the degree of formalism.

2.3. The level of formality increases as we move closer to the implementation level.

from http://www.architectingforum.org/bestpractices.shtml
Possible useful visualizations

actual figures and references to their use at http://www.gaudisite.nl/figures/<name>.html
Ideal Structure does not exist

A Reference Architecture Primer
©2006, Embedded Systems Institute
138  ©2006, Gerrit Muller
Synthesis, Integration, Relation Oriented

1. Functional Decomposition

2. Construction Decomposition

3. Allocation

4. Infrastructure

5. Choice of integrating concepts
Checklist for RA content

customer context

business
financials
stakeholders
benefits, concerns

concept of operations

relations
guidance

technical architecture

key performance parameters
product features, functions

core technologies
critical resources
design issues
dominant patterns

business model
life cycle
stakeholders
benefits, concerns
Summary of the role of Reference Architectures

- **mission**
- **vision**
- **strategy**

Reference Architecture

- **technology**
- **elaboration**
- **needs**
- **opportunities**
- **knowledge**
- **guidance**

- **customers**
- **market**

- **existing architectures**

- **multiple organizations**

- **new or evolved architectures**
6 Assessment & Evolution
exercise:
  define 3 change cases
  determine impact of 1 change case
Evolvability
### High Level Problem Statement

<table>
<thead>
<tr>
<th>Category</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installed Base Business</td>
<td>costly</td>
</tr>
<tr>
<td>Life Cycle Management</td>
<td>high effort</td>
</tr>
<tr>
<td>diversity and # of configurations</td>
<td></td>
</tr>
<tr>
<td>Development efficiency</td>
<td>costly</td>
</tr>
<tr>
<td></td>
<td>high effort</td>
</tr>
<tr>
<td></td>
<td>too late</td>
</tr>
<tr>
<td>Innovation rate</td>
<td>too low</td>
</tr>
<tr>
<td></td>
<td>too late</td>
</tr>
</tbody>
</table>

see next slides
**Evolvability Problem Statement**

- **exploration is difficult**
  - too much
time, effort, cost
from idea to tryout

- **reliable realization is difficult**
  - too much
and unpredictable
development
time, effort, cost
from tryout to realization

- **engineering is difficult**
  - some new features
late relative to competition
too much
material and labor cost

---

**innovation life cycle**

- tryout
exploration of innovative
features

- scale up
for clinical use

- scale up
for volume sales

---

1
10
100

---

A Reference Architecture Primer
©2006, Gerrit Muller

version: 0.6
July 24, 2014
DERAproblemStatement
Sources of Change

- Customer context
  - Humans
  - Other systems
  - Legislation
  - Reimbursement

- Technical architecture
  - Clinical applications
  - Workflow applications

- Business architecture
  - Domain specific technology
  - Generic technology

- Competition
- Organization
- Business model
Sources of Change

**Customer Context**
- Humans
- Other systems
- Legislation
- Reimbursement

**Technical Architecture**
- Clinical applications
- Workflow applications
- RF coils
- Gradient amplifier

**Business Architecture**
- Competition
- Organization
- Business model

**Generic Technology**
- PMW
- PII

**Domain Specific Technology**
- USA
- Windows Vista
- PCI-X
- Database
Reuse in CAFCR perspective

What does Customer need in Product and Why?

Customer What
- Customer objectives

Customer How

Product What
- Functional

Product How
- Conceptual
- Realization

rate of change

"easy" reuse

costly reuse

Understanding spec design implementation
Myth: Platforms are Stable

Dynamic Market

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

Architecture

Components

Platform

Fast changing Technology

1991

1992

Change

Growth

1994

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

1996

Last changed in:

1991

1992

1994

Growth
Lifecycle Differences

- **problem response**
  - 3 months

- **clinical prototype**
  - 1 year

- **procedural change**
  - 1 year

- **legislation change**
  - 1 year

- **workstation useful life**
  - 10 years

- **MR scanner useful life**

- **new generation of magnets\n  gradients\n  detectors**

- **minor SW release**

- **major SW release**

- **commodity hardware and software**
Reference Model for Healthcare Automation

Information handling
- entirely distributed
- wide variation due to "socio-geographics": psycho-social, political, cultural factors

Imaging and treatment
- localised
- patient focus
- safety critical
- limited variation due to "nature": human anatomy, pathologies, imaging physics

Image handling
- distributed
- limited variation due to "nature": human anatomy, pathologies, imaging physics

Base technology
- not health care specific
- short life-cycles
- rapid innovation

Archiving
- service business
- not health care specific
- extreme robust
- fire, earthquake, flood proof
- life time
- 100 yrs (human life)
Exponential Pyramid, from requirement to bolts and nuts

- $10^0$: number of details
- $10^1$, $10^2$, $10^3$, $10^4$, $10^5$, $10^6$, $10^7$: research focus
- system
- multi-disciplinary system
- mono-disciplinary

- $10^0$: system
- $10^1$: requirements
- $10^2$: design decisions
- $10^3$: parts
- $10^4$: connections
- $10^5$: lines of code
Waferstepper Example

- Source
- Reticule
- Lens
- Wafer

10 Mloc

Overlay: 45nm
CD Control: 10nm
Productivity: 100Wph
From Components to System Qualities

components

subsystems

functions

system qualities

overlay
CD control
productivity

prepare

align and calibrate

scan and expose

transport wafer

laser

lens

stages

handlers

electronics infra

metro frame

cabinets

ICs

PCBs

drivers

monitor

disks

computer

TCP/IP

comms package

database

user interface

robot

bolts

nuts

transport wafer

transport reticle

air mounts

lamps

air showers

flaps

sensors

robot

motors

mirrors

fiducials

lens elements

frames

sources

准备好

对准和校准

扫描和暴露

运输晶圆

激光器

镜头

阶段

处理程序

电子基础设施

地脚

控制CD

产品

系统属性

图层

底板

计算机

磁盘

显示器

驱动程序

数据库

用户界面

通信包

螺栓

螺母

光源

显微镜

基准

透镜元件

叶片

空中喷射器

框架
Role of Software

- **Components**: Basic building blocks of a system.
- **Subsystems**: Groups of related components.
- **Functions**: The behaviors or actions that a system performs.
- **System Qualities**: Attributes that describe the system's behavior or characteristics.

Software (SW) implements functionality and determines emerging qualities.
When SW engineers demand "requirements",
then they expect *frozen* inputs
to be used for
the design, implementation and validation
of the software
System vs Software Requirements

10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7

number of details

system requirements

multi-disciplinary

software requirements

mono-disciplinary

A Reference Architecture Primer
©2006, Embedded Systems Institute
158 ©2006, Gerrit Muller

version: 0.6
July 24, 2014
VREQpyramid
Why is the Software Requirement Specification so Large?

- software subsystem
- user interface
- system behavior
- operational choices
  - synergy, tools, ...
- limited computing resources
- control of physical subsystems:
  - sensors, actuators
And why is it never up-to-date?