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

- Reduced time to market
- Reduced cost per function
- Improved quality
- Improved reliability
- Improved predictability
- Easier diversity management
- Increases uniformity
- Employees only have to understand one base system
- Larger purchasing power
- Means to consolidate knowledge
- Increase added value
- Enables parallel developments of multiple products
- “Free” feature propagation

building on shared components
build every function only once
maturing realization
modularity
less learning
not reinventing existing functionality
product-to-product or project-to-project

economy of scale
## Experiences with reuse, from counterproductive to effective

<table>
<thead>
<tr>
<th>bad</th>
<th>good</th>
</tr>
</thead>
<tbody>
<tr>
<td>longer time to market</td>
<td>reduced time to market</td>
</tr>
<tr>
<td>high investments</td>
<td>reduced investment</td>
</tr>
<tr>
<td>lots of maintenance</td>
<td>reduced (shared) maintenance cost</td>
</tr>
<tr>
<td>poor quality</td>
<td>improved quality</td>
</tr>
<tr>
<td>poor reliability</td>
<td>improved reliability</td>
</tr>
<tr>
<td>diversity is opposed</td>
<td>easier diversity management</td>
</tr>
<tr>
<td>lot of know how required</td>
<td>understanding of one base system</td>
</tr>
<tr>
<td>predictable too late</td>
<td>improved predictability</td>
</tr>
<tr>
<td>dependability</td>
<td>larger purchasing power</td>
</tr>
<tr>
<td>knowledge dilution</td>
<td>means to consolidate knowledge</td>
</tr>
<tr>
<td>lack of market focus</td>
<td>increase added value</td>
</tr>
<tr>
<td>interference</td>
<td>enables parallel developments</td>
</tr>
<tr>
<td>but integration required</td>
<td>free feature propagation</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Product Families and Generic Aspects**

©2006, Gerrit Muller
### Successful examples of reuse

<table>
<thead>
<tr>
<th>Homogeneous Domain</th>
<th>Hardware Dominated</th>
<th>Limited Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>cath lab</td>
<td>car</td>
<td>audio codec</td>
</tr>
<tr>
<td>MRI</td>
<td>airplane</td>
<td>compression library</td>
</tr>
<tr>
<td>television</td>
<td>shaver</td>
<td>streaming library</td>
</tr>
<tr>
<td>waferstepper</td>
<td>television</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

Internal benefits
- asset creation
- increase economy of scale

Extrovert driver

Introvert driver

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale

Internal benefits
- asset creation
- increase economy of scale
Granularity of generic developments shown in 2 dimensions
Financial Viewpoint on Process Decomposition

- **Customer**
  - Sales
  - Logistics
  - Production
  - Service
  - Presales

- **People**
- **Technology**
- **Process**

- **Management**

- **Cashflow generation**
  - Sales
  - Logistics
  - Production
  - Service
  - Support
  - Order
  - Information
  - Material

- **Tomorrow's Cashflow**

- **Strategic Asset Generation**
  - Product Needs and Feedback
  - Product related processes
  - Technical Product Documentation

- **Assets**
  - Business Drivers
  - Customer Roadmap
  - Technology, Process and People roadmaps
  - Budgets

- **Product Families and Generic Aspects**

©2006, Embedded Systems Institute
©2006, Gerrit Muller

version: 2.3
July 24, 2014
GDprocessDecompositionFamilyByValue
Modified Operational Organization PCP

- **operational**
  - portfolio operational manager
  - family operational manager
  - project leader
  - single product project leader
  - component project leader
  - module component developers

- **technical**
  - portfolio architect
  - family architect
  - platform architect
  - product architect
  - component architect
  - subsystem architect
  - subsystem developers

- **commercial**
  - portfolio marketing manager
  - family marketing manager
  - platform manager
  - product manager
  - component manager

---

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

Suppining Business

- Policy and Planning
- Customer Oriented Process (sales, service, production)
- 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
- No Feedback

Technology Push

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

version: 2.3
July 24, 2014
GDmodels
Product Related Life Cycles

- Individual systems
- Service
- System creation
- System production
- System sales
- Upgrades and options production
- Upgrades and options sales
- Disposal
System Life Cycle

- System
- Order
- Using
- Local changes, e.g.
  - Accounts procedures
- Using
- Upgrade
- Sales
  - Shipping
  - Refurbishing
  - Shipping
  - Installation
  - Secondary use
- Maintenance
- Dispose
Creation Chain

*shared*
strategy, processes,
tools, repository

*aligned*
strategy, processes,
tools, repository

*independent*
strategy, processes,
tools, repository

Component Creator

Product Creator & Integrator

In House Customer

Partner Product

Privileged Customer

Off the Shelf Component Supplier

Complementor

Direct Customer

Competitor

Customer via Reseller

---

Product Families and Generic Aspects

©2006, Embedded Systems Institute

©2006, Gerrit Muller

version: 2.3

July 24, 2014

PEVOCreationChain
Customer Oriented Process
Impact of Procurement Duration

Customer order \[t_{\text{demand}}\] desired delivery

\[t_{\text{procurement}}\]

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