Abstract

This module addresses product families and generic developments.
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>Description</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></td>
</tr>
<tr>
<td>Improved reliability</td>
<td>maturing realization</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></td>
</tr>
<tr>
<td>“Free” feature propagation</td>
<td>product-to-product or project-to-project</td>
</tr>
</tbody>
</table>

**Product Families and Generic Aspects**

4 Gerrit Muller

version: 2.3

October 19, 2014

GDclaims
<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>
</tbody>
</table>
Successful examples of reuse

<table>
<thead>
<tr>
<th>homogeneous domain</th>
<th>cath lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRI</td>
<td>television</td>
</tr>
<tr>
<td>television</td>
<td>waferstepper</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hardware dominated</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane</td>
<td>shaver</td>
</tr>
<tr>
<td>television</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>limited scope</th>
<th>audio codec</th>
</tr>
</thead>
<tbody>
<tr>
<td>compression library</td>
<td>streaming library</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 of accumulated feature set
- design for configurability
- shared architectural framework
- quality increase
- maturity

Internal benefits
- asset creation
- predictability
- availability integrated base product
- increase economy of scale
- predictability

Extrovert driver
- new features originating from different products
- timely availability
- reliability

Introvert driver
- availability variations
Granularity of generic developments shown in 2 dimensions

Delegated integration

Shared integration

actual integration level

system

platform

subsystem

module

component

CV

MIP

EVM

Generator

flat detector

CCD

Delegated integration

Shared integration

actual integration level

system

platform

module

subsystem

component

CV

MIP

EVM

Generator

flat detector

CCD

Product Families and Generic Aspects

version: 2.3
October 19, 2014
GDgranularity

Gerrit Muller
Modified Process Decomposition

Customer-Oriented Process
Product Creation Process
Policy and Planning Process
Shared Assets Creation Process
People, Process, and Technology Management Process

Customer Roadmap
Business Drivers
Product needs and feedback
Product roadmap, Budget, plan
Generic assets roadmap, Budget, plan
Customer Roadmap
Product needs and feedback
Product roadmap, Budget, plan
Generic assets roadmap, Budget, plan

Needs & Feedback
Technical Product Doc.
Product-related processes
Product needs and feedback
Product roadmap, Budget, plan
Generic assets roadmap, Budget, plan

Requirements & Feedback
Needs & Feedback
Product needs and feedback
Product roadmap, Budget, plan
Generic assets roadmap, Budget, plan

Product needs
and feedback
material

Support

Customer

Product Families and Generic Aspects

version: 2.3
October 19, 2014
GDprocessDecompositionFamily
Value and Feedback Flow

customer

Policy and Planning Process

Product Needs and feedback

Product Creation Process

Feedback

Value

Shared Assets Creation Process

Needs & feedback

People, Process, and Technology Management Process

Customer-Oriented Process

Presales sales logistics production service

Support

Material

Information

Technology, product, roadmaps

Budgets

Policy and Planning Process

Customer roadmap

Business drivers

Product Needs and feedback

Product Creation Process

Shared Assets Creation Process

Product Needs and feedback

People, Process, and Technology Management Process

needs

Product-related processes

Technology, Process, and People roadmaps

Product Families and Generic Aspects

version: 2.3

October 19, 2014

GDprocessDecompositionFamilyPlusFlow

12 Gerrit Muller
Modified Operational Organization PCP

Product Families and Generic Aspects

Gerrit Muller

version: 2.3
October 19, 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
  - Supplying business
    - 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 Families and Generic Aspects
  - Version: 2.3
  - October 19, 2014
  - Gerrit Muller
  - GDmodels
What are the top 3 benefits for your product family or generic development?
What are the top 3 disadvantages?
Roadmap Creation

Contradicting Experiences

**good**
- reduced time to market
- reduced investment
- reduced (shared) maintenance cost
- improved quality
- improved reliability
- easier diversity management
- understanding of one base system
- means to consolidate knowledge
- increased added value
- enables parallel developments
- free feature propagation

**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

Drivers

- application adaptability
- availability variations
- design for configurability
- shared architectural framework
- timely availability
- reliability
- quality increase
- predictability
- maturity
- asset creation
- increase economy of scale

Shared Asset Creation Process

Longer Chains

Exercise Product Families and Generic Developments

version: 2.3
October 19, 2014
Some Architecting Means

Organizational Complexity

Delay to Market

Pitfalls

Successful and Failing Models

Exercise Product Families and Generic Developments

version: 2.3
October 19, 2014

Gerrit Muller