Variant Management for Software-intensive System Families
Dr. Martin Becker

http://vm.iese.fraunhofer.de
Context

Engineering of variant-rich software / system families
Fraunhofer IESE – Facts & Figures

- Founded in 1996
- One of the leading institutes in software engineering in Europe and worldwide
- 200 employees
- Six application-oriented business fields
  - Product Sector
  - Service Sector
- Special departments for all phases of the lifecycle of software-based products
Variation Management Improvement

Systematic Variation Management Improvement

Characterize & Understand

Assessment of VM practices

Plan

Scoping Workshop

Domain Analysis & Modelling

Do

Support for VM Tool Adoption

VM in Artefacts

Evolve

Configuration Management

Change Management

Applied research since 1997
(= 3-4 innovation cycles)

Industry Partners (selection)
Discussion

- Please introduce yourself
  - What is your **background**?
  - **What systems** are you dealing with?
  - **Complexities** in your context:
    - System size
    - Organization (Distributed development, ...)
  - State your **expectations**
Challenge: Increasing Customization Demands

- Customers request specialized products adapted to their needs
  - Different system configurations
  - Different behavior
  - Different system interfaces
  - Different look and feel
Challenge: Complexity Increase in Software-intensive Systems

Overall complexity increases exponentially

Software can be changed easily / lately / fast

[source: adapted from Univ. of St. Gallen]
Industrial Trend: Front-Loading ➔ MB-SE

[source: CESAR Book, Springer]
Challenge: Speed Of Development

- Cycle time reduction outperforms efficiency improvements

“If you are not moving at the speed of the marketplace you’re already dead – you just haven’t stopped breathing yet.” Jack Welch
Challenge: Integrated Lifecycle and Variant Management

![Diagram showing Space (Variants) and Time (Versions) with variability and evolvability dimensions.](image)

- Variability
- Evolvability
Surviving the Variant Jungle

- >1,000 variable features
- >10,000 variation points
- 2,000 customized variants per year
- Runtime adaptation
- >10,000 variable parameters
- ISO 26262
- Variable binding time
- Shorter release cycles

Model-based/driven Development
Agenda

1. Motivation
2. Strategic Variant Management of Software-intensive Systems
3. Adoption Patterns and Experiences
4. Open Issues
5. Summary
Clone-and-Own Approach has Limitations

CLONING REASONS

- Unexpected request for a similar product
- Little available development resources
- Time, effort
- Missing focus on reuse
- Unknown future: reuse need not predictable
- Deliberate decision: independence from other projects

CONSEQUENCES

- Short-term:
  - + Effort savings
  - + Quick delivery of the new product
- Long-term:
  - - High maintenance effort
  - - Repetitive maintenance tasks

Based on our industry survey

"It gives freedom to change, [when cloning] there is no damage to existing products."

"At the beginning we did not know that we will have to support all the controllers that we support now – this emerged over time."

"It is easier to start with something. Cloning gives [us] an initial basis."

"It saves time. These components were already used, tested, closed. A kind of an off-the-shelf software."

"We need to perform many activities several times: for each variant, we have to check the code and implement the change or fix."

"(... ) code that we cloned looses connection with the product which it is cloned from, and then there is no sharing of new insights and innovations."
Challenge: Increasing Maintenance

![Graph showing the relationship between the number of delivered systems (to be maintained) and the percentage of resources allocated to developers and maintainers. The graph indicates a trend of decreasing resources for developers and increasing resources for maintainers as the number of delivered systems increases.]

Maintenance Paralysis
Challenge: Management Complexity of Cloned Variants
Followed reuse approaches **did not achieve** expected improvements. Focus was **small-grained, opportunistic, and technology-driven**.

Reuse repositories with more than 100 elements can cause substantial search and evaluation costs.
With Which Variants do you Earn the Most?

With the ones that can be avoided!

1. **Avoid unnecessary** variants
2. **Master necessary** variation
3. **Reduce opportunistic** variants/variation

Try to identify and avoid unnecessary variability.
Success Story: Cummins, Inc.

Cost
- Management estimates product line **ROI of 10:1**

Time to Market
- Product cycle time: **a year to a few days**

Productivity
- 20 product groups → 1000 separate applications
- 75% of all software comes from core assets
- Productivity **improvement of 360%**

Enter new Markets
- Capability let Cummins enter and dominate industrial diesel engine market

Quality
- Software **quality** is at an **all-time high**
- **15 of 15 projects are on track** (was 3 of 10)
- **Customer satisfaction is high.**

[source: SEI]
Strategic Reuse / Variation Management is Needed for Business Benefits

Product Line
- Take economic advantage of commonality
- Bound variation

ISO/IEC 26550 – PLE Reference Model
Software and Systems Product Line Engineering (PLE)

[Diagram showing the structure of PLE with stages such as Application Requirements Engineering, Application Design, Application Realization, Application Verification & Validation, Domain Verification & Validation, Domain Requirements Engineering, Domain Design, Domain Realization, Organizational Management, and Technical Management.]
Reuse in the Large

In order to achieve higher levels of reuse one needs to

- Increase the granularity of the reused parts
- Provide building plan

Reuse-in-the-Small vs Reuse-in-the-Large
Keep Adaptation Cost Small

Study of reuse costs in NASA [1994, Barry Boehm]

[source: http://www.sigapp.org/acrl/issues/V5.2/cardino.html]
Core Asset:

```
“subset of assets that are **developed** in the domain engineering process or obtained by another way **for reuse** in the application engineering process.”
```
Variability

Variability \(\equiv\)

“a property that is different among the members of a product line”

“a capability to change or adapt a system”

Notes:

Counterpart commonality: ... same ... all

A kind of variable feature

Delayed (design) decision: FE -> AE

- Variability \(\rightarrow\) Decision
Variation Point

Variation Point :=

“represents one or several locations at which variation will occur within core assets”.

1. to highlight where variant elements occur (which makes variation easy to see and control)
2. to improve traceability of variability (requires that goal 1 has been fulfilled).
Orthogonal VM / FM

Bill of Material ➔ Bill of Features
Variability Management: Separation of Concerns

- **Problem Space/External Perspective**
  - **Variability Model** (Variabilities, Variants, Dependencies, Configurations)

- **Variant Specification Model** (Concrete Configuration)

- **Solution Space/Internal Perspective**
  - **Solution Assets**
  - **Core Assets, Family Model**

Domain Engineering: Develop for Reuse

Application Engineering: Develop with Reuse
Variability Specification Approaches

- Start with Feature Modelling
- Develop DSL if FM is not adequate
- [VM Tool support lacking] Consider usage of Decision Models

Adequate Specification Approach
A decision model is a model that captures **variability** in a product line in terms of **open decisions**, possible **values/resolutions** and the respective **effects**.

<table>
<thead>
<tr>
<th>ID</th>
<th>Question</th>
<th>Variation Point</th>
<th>Resolution</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Can cups be warmed up?</td>
<td>coffee machine</td>
<td>yes</td>
<td>A warming plate is attached</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>No warming plate is attached</td>
</tr>
<tr>
<td>2</td>
<td>Does the machine has a crushing mill for coffee, a slot for putting in already crushed coffee or both?</td>
<td>coffee machine</td>
<td>crushing mill</td>
<td>A crushing mill is installed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>coffee slot</td>
<td>A slot for coffee powder is installed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>both</td>
<td>Resolve decision 4: no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A crushing mill and a slot for coffee powder is installed</td>
</tr>
<tr>
<td>3</td>
<td>Does the machine has a milk frother or a cappuccinatore?</td>
<td>coffee machine</td>
<td>milk frother</td>
<td>A milk frother is attached</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cappuccinatore</td>
<td>A cappuccinatore is attached</td>
</tr>
<tr>
<td>4</td>
<td>Does the machine have a container for coffee beans?</td>
<td>coffee machine</td>
<td>yes</td>
<td>A container for coffee beans is installed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>No container for coffee beans is installed</td>
</tr>
</tbody>
</table>

Decision models can be used **regardless** the type of the artifact (documents, models, code) => **orthogonal variability modeling**

- Requires no tool support
- Supports manual resolution of VPs

© Fraunhofer IESE
Feature Models

Describes the features (mandatory, optional, alternative) of a product line member

Available / Used Variability Mechanisms

General
- Parameterization
- Configuration
  - Selection
  - Module-Replacement
- Composition

Extension
- Domain-Specific-Language
  - Generation
  - Interpretation
- Conditional Execution (at runtime)
- Transformation

SW-Specific
Orthogonal VM / FM

Bill of Material ➔ Bill of Features
Success Stories

As simple as possible, no matter what the cost.

Ludwig Mies van der Rohe

2nd Generation PLE Approach: 150%

Feature-based Abstraction

[Diagram showing the process of feature-based abstraction with stakeholders like Requirements Engineers, Architects, Developers, and Quality Assurance.]

Key
- Variation Points

[source: Provided courtesy of BigLever Software]
Feature-based Variability Management

- Feature Selection
- Processing
- Core Assets

Application Engineering

Family Engineering

Feature Model

© Fraunhofer IESE
Feature-based Variability Management in Practice: Pure::Variants
Feature-based Variability Management in Practice: GEARS
Challenge: Variability Erosion of Core Assets

```cpp
#include <errors.h>
#include <list.h>
#include <portable.h>
#include <projdefs.h>
#include <queue.h>
#include <semphr.h>
#include <task.h>

#include <list.c>
#include <portable.c>
#include <queue.c>
#include <tasks.c>
```

```cpp
//main.cpp
#ifdef COLOR
    paint();
#endif

#if SIZE>20
    large();
#else
    small();
#endif
...
Agenda

1. Motivation
2. Strategic Variant Management of Software-intensive Systems
3. Adoption Patterns and Experiences
4. Open Issues
5. Summary
## Common Variant Management Approaches

<table>
<thead>
<tr>
<th>Independent</th>
<th>Platform-based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strategic</strong></td>
<td><strong>Ad-hoc</strong></td>
</tr>
<tr>
<td>Managed Cloning</td>
<td>Clone&amp;Own</td>
</tr>
<tr>
<td>Production Line (80%)</td>
<td>Reuse Repository</td>
</tr>
<tr>
<td>Platform (50%)</td>
<td>Configurable Product (150%)</td>
</tr>
<tr>
<td>Production Line (150%)</td>
<td></td>
</tr>
</tbody>
</table>

- **Independent** approaches are more strategic and focus on creating a product line that can be managed and cloned.
- **Platform-based** approaches are more ad-hoc and focus on using a platform to create a configurable product.

- **Managed Cloning** is a strategic approach where a product line is created and managed.
- **Clone&Own** is an ad-hoc approach where cloning is used to create a product line.
- **Reuse Repository** is an ad-hoc approach where reuse is implemented in a repository.

- **Product Line (80%)**: Represents 80% of the product line.
- **Platform (50%)**: Represents 50% of the platform.
- **Configurable Product (150%)**: Represents 150% of the configurable product.
Observations of Variant Management Approaches

<table>
<thead>
<tr>
<th>Independent</th>
<th>Platform-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed Cloning</td>
<td>Product Line (90%)</td>
</tr>
<tr>
<td>Clone &amp; Own</td>
<td>Platform (50%)</td>
</tr>
<tr>
<td>Reuse Repository</td>
<td>Production Line (150%)</td>
</tr>
<tr>
<td></td>
<td>Configurable Product (150%)</td>
</tr>
</tbody>
</table>

Diversification of VM approach
Evolution of VM approach
Lean & agile VM approach
A Lightweight Improvement Approach

Initiation
- Potential Analysis
- Variant Analysis

Planning
- Scoping
- Specify Variability
- Design PL Architecture
- Pilot Development
- Reuse Instrastr. Setup

Improvement
- Customize Reuse approach
- Improve Reuse approach
- QA Improvements
- Guide Core Assets Dev.
- PL-Configuration Management
- Reuse Instrastr. Improvements

Train VM Stakeholders

Funding
Scoping

Goals:
- Define business goals and constraints
- Plan the product line

Activities: PuLSE™-ECO Scoping
- Prepare Scoping
- Conduct Scoping Workshops

Results:
- Product-Release-Plan
- Product-Feature-Matrix
- Domain- and Asset-Assessment
Variant Analysis

- Variant Analysis can be used for:
  - Analyzing the reuse potential
    - Both high-level and detailed analysis
    - Determining suitable reuse approach (product lines, reusable libraries,...)
  - Analyzing existing variants
    - For example preprocessor-based code
  - Planning migration towards the selected reuse approach
    - Determining migration scope
    - Determining suitable starting point
  - Supporting the migration
    - Detailed analysis results very helpful for restructuring the code
    - Fast model updates possible after any change
Specify Variability

Goals:
- Provide variability model(s) for reuse programme

Activities:
- Refine variability-related information from scoping
- Provide overview on variability modeling approaches & tools
- Model variability (including interdependencies) in a dedicated variability model

Results:
- Variability Model
- Tool selection
PL Architecture Design

Goals:
- Design PL architecture (addressing VM aspects)
- Evaluate PL architecture for its suitability
- Document and communicate PL architecture to relevant stakeholders

Activities:
- Design PL architecture in an incremental and iterative way
- Evaluate PL architecture with relevant stakeholders
- Plan communication of the architecture
- Document architecture in adequate manner (e.g. UML / SysML)
- Continuously communicate the architecture

Results:
- Documented and communicated PL architecture
Fraunhofer Variability Improvement Analysis (VITAL)

Analysis Method & Tool to:

- Analyze reuse infrastructure, generic solutions
- Understand variability erosion trends
- Create a comprehensive model of variabilities and their usage
- Calculate and manage variability-related metrics
Fraunhofer Variability Improvement Analysis (VITAL)

How are the variation points nested?
Are there clear hierarchical dependencies?

1334 nodes (variabilities)
1570 edges (dependencies)

Identify dependencies of parameter names to ease configuration.
INLIVE - Integrated Lifecycle and Variant Management

- Guidelines
- Examples
- Tool Integration
CRYSTAL Project

http://www.crystal-artemis.eu/

INTEROPERABILITY

Interoperability Specification and Reference Technology Platform

IESE Contribution:

Tool and method integration in:

- Variation Management
- Model-based Requirements Engineering
- Virtual Engineering (Simulation)
- Safety Engineering
Variant Management of Safety-critical Software-Intensive Systems

- Different Cost-of-Change profile
- Cost-of-Change is often not known
- What are the areas where variant management is most useful / applied?
  - How to establish reuse-able building blocks on technical solution level?
  - How big are typical reusable building blocks?
  - Shall reuse-able solutions be fixed or configurable?
- How to approach variant management, if the standard is not (that) reuse aware?
- Where to apply variant management: functional architecture / logical architecture?
- Proactive variant management seems to be more suitable
Agenda

- Motivation
- Strategic Variant Management of Software-intensive Systems
- Adoption Patterns and Experiences
- Open Issues
- Summary
Summary

Thank you for your attention!

As with the Non-software Disciplines

- Managing large-scale system families requires strategic variant management approaches
- Business-driven, Architecture-Centric, Coordinated
- Sound management of interdependencies is key

SW-specific Issues

- **Speed of development** is competitive factor
- Broad range of variant management approaches exists
- **No one-size-fits** all – domain specific tailoring of VM approach
- Support transition between VM approaches (refactoring)

Contact:

**Dr. Martin Becker**  
Dipl.-Inform.  
Department Head  
ES Development  
Fraunhofer-Platz 1  
67663 Kaiserslautern  
Telefon +49(631) 6800-2246  
Fax +49(631) 6800-92246  
martin.becker@iese.fraunhofer.de  
www.iese.fraunhofer.de