Effective Reuse
From consistent requirements to variant management

Christof Ebert, Silke Riegraf, Vector
Martin Bobert, Stihl
Motivation
Intelligent Reuse
Change Project
STIHL – the world’s best-selling chain-saw brand since 1971

STIHL is a globally leading manufacturer of chain saws and other power tools for professional forestry and agriculture as well as for garden and landscape maintenance, the construction sector and the demanding private user.
STIHL – Products

- Saws
- Lawn mowers
- Brushcutters
- Hedge trimmers
- Garden shredders
- Blowers
- Cut-off machines
Vector Consulting Services

- supports clients worldwide
- improves product development and product strategy, IT and provides interim management
- is as Vector Group globally present with 1750 employees and well over 400 Mio. € sales
- offers with the Vector Group a portfolio of tools, software components and services
- is growing and thus continuously hiring

www.vector.com/consulting
Cost remains biggest short-term challenge across all industries. Connectivity and Digital Transformation evolved as a major challenge.
Motivation

Intelligent Reuse

Change Project
Intelligent Reuse implies a strategic focus across the life-cycle connecting product management, engineering and operations.
### Four Generic Strategies are Pursued

<table>
<thead>
<tr>
<th>Platform based products (45%)</th>
<th>Individualized products (15%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same platform with individual modules for customization</td>
<td>High degree of individualization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standardized products (15%)</th>
<th>Module based products (25%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardized products for different markets</td>
<td>One module set with different platforms for customization</td>
</tr>
</tbody>
</table>

One Platform(s) vs. Several Platforms

Note: Percentages provide desired usage from current surveys Vector with clients.
## Intelligent Reuse

### A Practical View on PLE and Intelligent Reuse

<table>
<thead>
<tr>
<th>PLE for software applications</th>
<th>Intelligent Reuse for systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software-driven foundations and principles</td>
<td>Need translation to product culture with components and modularization</td>
</tr>
<tr>
<td>Savings achieved at software applications with many variants</td>
<td>Savings achieved with focus on testing, documentation, certification</td>
</tr>
<tr>
<td>Many strictly defined principles; but often dogmatically applied</td>
<td>PLE methodology needs adjustment to specific context and markets</td>
</tr>
<tr>
<td>Handled inside the software teams, often combined with Scrum</td>
<td>Driven by product management, engineering, sales</td>
</tr>
<tr>
<td>UML as lingua franca; everything is modeled</td>
<td>Top-down modeling to manage dependencies and variation</td>
</tr>
<tr>
<td>Focus on mere software projects and limited scope</td>
<td>Products with long life-cycle and high criticality, e.g. safety, maintainability</td>
</tr>
<tr>
<td>Typically applied in rather small software teams</td>
<td>Distributed development with suppliers and collaborative design</td>
</tr>
<tr>
<td>Introduction easy within 1 year – for application software</td>
<td>Introduction needs thorough change management</td>
</tr>
</tbody>
</table>
Intelligent Reuse

Change to Reuse often Fails

- Over 50% of reuse programs fail
  - Initial cost savings not achieved
  - No black box reuse

- Reasons
  - Product management and sales not involved
  - Change is underestimated and handled ad-hoc
  - Insufficient know-how
  - Inadequate method and tools

Sources: Vector Consulting Services 2012
Motivation
Intelligent Reuse
Change Project
Complex Systems: Initial Alignment Steps

Change Project

**Strategy**

- Project 1: concept, bid
- Integration, Validation
- Development
- Project 2: concept, bid

**Components**

- **Standard Features**
  Core features for all markets which evolve in a single track. Intelligent variation management for requirements, automatic testing, documentation, etc.

- **Configurable Features**
  Features that do not apply in all markets. Variation management maintained on as-needed base. Test-driven requirements.

- **Evolution Options**
  Features, which are currently not available

**Roadmap, Architecture, Governance**

Business case: short-term, long-term

**Customized features**
Method and Tool Support

- **Model-based Development from Architecture Design to Series-Production Readiness**

- **Series-Production Use Cases**
  - Requirements Engineering
  - Function, System and Component Design
  - Software & System Design
  - Network Communication Design
  - Variant Management
  - Test Data Management
  - Change and Release Management
  - Design of Safety Relevant Systems

- **Usability and Performance**
  - Many Users
  - Sporadic Users

- **Integration**
  - Requested Import and Export Filters
Increasing functionality.
Increasing code size
Much higher effort for testing.
Software Specification and Test – Starting point

Stihl Project: XXXX  STIHL M-Tronic

Software Funktionsbeschreibung

Structure
- Exact description of each single function
- Discrete numbering of requirements
- Complete parameter list (800 Parameters)
- Versioned document with change log and history
- Size: ~180 pages

Advantage
- Completeness with no room for interpretation
- Basis for error free implementation
- Mapping between RQM and test (full traceability)
Managing Software Variants

It's difficult to manage multiple projects by Word and Excel!
Development process by PREEvision

- Project start
  - Project Specification (Reuse from other projects)
- Change request
  - Accepted Requirement Amendments
  - CHM
  - Found Specification Errors
- RQM
  - Testcase Creation
- TDM
  - Test Scripts according to Test Planning
- Supplier
  - Software Development Tools (model-based SW-development)
  - Check-in models
  - Found SW-Bugs
  - Testcase scripts
- Source Control, Software Bug-Tracking

(Automated) Test Stands (HIL)
Reusing existing requirements...

The baseline contains approved requirements of all variants

New Projects will initially reuse requirements from the baseline

Requirements are structured in packages by functions

Custom metric „Create Reuse Of Structure“

- Copy complete structure (elements of meta class requirement package) into the projects product line
- Create „reuse“ of each requirement in the selected structure
Project specific changes require a branch from the baseline

Reused requirements will be detached from baseline → this process is implemented via a custom metric

Custom metric „Detach Reuse“ = Branch-Metric

- Copies requirement which needs to be branched
- Removes the reuse of the baseline requirement
- Creates a requirement link between the baseline and the branched requirement (the copy from step 1) and saves the version (# of check-ins) of the baseline in the link description

```
public Object calculateResult() {
    MRequirement requirement = (MRequirement) getInput(input);
    if (reusedCheck(requirement)) {
        if (requirement.mdfGetVersionID() != null) {
            String versionID = requirement.mdfGetVersionID();
            MRequirement copiedReq = createACopy(requirement);
            String branchVersion = getBranchVersion(requirement);
            deleteRequirement(requirement);
            searchRegAndCreateReqLink(copiedReq, branchVersion, versionID);
            return versionID;
        }
    }
}
```

The reused project after execution of „Detach Reuse“ metric

excerpt of the metric’s Java code
Standardization of made requirement changes (in branches) in certain time intervals → merge into new version of the baseline

Metric „Merge Branch into original requirement“

- Copy branched requirement with all attributes into the according baseline requirement
- Delete branch requirement link
- Create a reuse of the new version of the baseline requirement in the project
- Delete branched requirement
Implemented report generator for the project data sheets (uses calculation and diagram metrics)

**Project data sheet report generator**

**Metric** "Generate Diagrams"

**Metric** "Calculate Values"

*Schematic structure of project data sheet report generator and screenshots of .pdf export*
Test Data Management

Create and manage

- Test case specifications (Test Case Description)
- Test case implementation (scripts, manual execution steps)
- Test run evaluation (test coverage, planning, statistics)

Mapping between test-specifications, requirements and parameters

Connection between RQM and TDM
Test Data Management

Customizing for TDM

Generate config files for test cases to parameterize scripts of automated test cases for HIL-testing

Implement test case analysis on basis of test cases and whole SW-modules

Calculate degree of maturity based on planned test phases and test case run results

Overview over test case analysis

Degree of maturity over time
## The Need for Change

<table>
<thead>
<tr>
<th>Categories</th>
<th>Challenges</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>▶ Ownership of assets unclear</td>
<td>▶ Strengthen coherent product strategy and product-driven organization</td>
</tr>
<tr>
<td></td>
<td>▶ Structure does not match product needs</td>
<td>▶ Optimize product management, engineering and integration organizations</td>
</tr>
<tr>
<td>Culture</td>
<td>▶ Reluctance to invest for future</td>
<td>▶ Commit clear reuse targets, related to market, revenues, cost, technical debt</td>
</tr>
<tr>
<td></td>
<td>▶ Feedback from market to engineering fails</td>
<td>▶ Create conscious product culture</td>
</tr>
<tr>
<td></td>
<td>▶ Too much project focus</td>
<td>▶ Drive projects from products, not vice versa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Perform business case for new features: short-term vs. long-term cost/benefits</td>
</tr>
<tr>
<td>Engineering</td>
<td>▶ Complexity will grow</td>
<td>▶ Install proactive market-driven reuse connected to development, test, documentation, certification</td>
</tr>
<tr>
<td></td>
<td>▶ Products needs new features</td>
<td>▶ Establish strong change board to judge tradeoffs based on overall business case</td>
</tr>
<tr>
<td></td>
<td>▶ Too many project-driven requests will erode architecture and reusability</td>
<td>▶ Maintain product and components roadmap as master for all changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Rigorous two-way traceability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Establish consistent tool support</td>
</tr>
<tr>
<td>Contractor and supplier</td>
<td>▶ Low incentive for maintaining reuse</td>
<td>▶ Motivate and reward conformance to product-line specifications</td>
</tr>
<tr>
<td></td>
<td>▶ Less control</td>
<td>▶ Include source code in supplier agreements</td>
</tr>
</tbody>
</table>
Strategic Level
- Align engineering, product management and sales on targets and implementation approach
- Perform rigorous reviews on roadmap, change requests, feature prioritization, market communication, dependency management, etc.
- Provide a “change map” with timeline, dependencies (people, projects, markets, platforms) and strategic relevance
- Establish organization to align stakeholders in product management, engineering, sales
- Apply strict governance criteria, namely short-term (cash flow from projects) vs. long-term (reuse of product)

Deployment Level
- Start with few components which deliver results
- Consider activities with high savings potential, e.g., validation, automatic regression testing, certification, component-based documentation
- Manage the change process with specific targets, ownership, communication, periodic reviews, share best practices
- Update processes and tools along the way
Conclusions

- **Intelligent reuse** helps reducing cost and improving efficiency
  - Modularization with tailored reuse concepts
  - Cost savings levers: automatic regression test, documentation, certification, feature evolution, service-oriented business models

- **Success factors**
  - Stepwise approach starting with requirements engineering
  - Thorough management buy-in and leadership
  - Strong product management with transparent product strategy
  - Architecture competence and governance

- **Major risks**
  - Unclear responsibilities and lack of ownership
  - Insufficient change management, and thus fallback to the past
Further Reading

Requirements Engineering

Christof Ebert
dpunkt.verlag
450 pages
5. fully reworked edition 2014

„The classic for systematically handling requirements. Written by a practitioner for practice – easy to understand and to apply! During a joint project I experienced first hand that the author knows what he talks about."

Hans Leibbrand, COO, Thales
Thank you for your attention.
Contact us – We are glad to support you.


Vector Consulting Services

Phone +49 711 80670-0  www.vector.com/consulting
Fax +49 711 80670-444  consulting-info@vector.com