

## D2.4: Requirements with KPI for REVaMP<sup>2</sup> 2.0

---

## Executive summary

This document describes for each use case, the requirements and KPIs updated by the REVaMP<sup>2</sup> project partners, for version 2.0 of the tool chain.

The access of the information contained in this document is public.

The organization of each use case description and requirements is based on templates created or updated by members of the work package 2 to fit the needs of the REVaMP<sup>2</sup> project.

## Table of Contents

- EXECUTIVE SUMMARY .....2
- TABLE OF CONTENTS .....3
- ACRONYMS .....5
- 1. INTRODUCTION .....6
- 2. ABB .....7
  - 2.1. Use case Description..... 7
  - 2.2. Goals..... 7
  - 2.3. KPIs ..... 7
  - 2.4. Requirements ..... 8
- 3. ROBERT BOSCH..... 13
  - 3.1. Use Case Description ..... 13
  - 3.2. Goals..... 14
  - 3.3. KPIs ..... 15
  - 3.4. Requirements ..... 16
- 4. MACQ ..... 20
  - 4.1. Use case Description..... 20
  - 4.2. Goals..... 20
  - 4.3. KPIs ..... 21
  - 4.4. Requirements ..... 21
- 5. SAAB ..... 23
  - 5.1. Use case Description..... 23
  - 5.2. Goals..... 23
  - 5.3. KPIs ..... 23
  - 5.4. Requirements ..... 24
- 6. SCANIA..... 25
  - 6.1. Use case Description..... 25
  - 6.2. Baseline ..... 25
  - 6.3. KPIs ..... 25
  - 6.4. Requirements ..... 26
- 7. SIEMENS ..... 27
  - 7.1. Use case Description..... 27
  - 7.2. Goals..... 27
  - 7.3. KPIs ..... 28
  - 7.4. Requirements ..... 28
- 8. SOFTEAM ..... 29
  - 8.1. Use case Description..... 29
  - 8.2. Goals..... 29
  - 8.3. KPIs ..... 30
  - 8.4. Requirements ..... 32

- 9. THALES ..... 33
  - 9.1. Use case Description..... 33
  - 9.2. Goals..... 33
  - 9.3. KPIs ..... 34
  - 9.4. Requirements ..... 36
- 10. THE REUSE COMPANY ..... 38
  - 10.1. Use case Description ..... 38
  - 10.2. Goals..... 38
  - 10.3. KPIs..... 39
  - 10.4. Requirements ..... 39
- 11. OVERALL PROJECT KPIs ..... 43
  - 11.1. Karlstad University..... 43
  - 11.2. THE REUSE COMPANY (Knowledge Centric Solutions) ..... 43
  - 11.3. KTH (Royal Institute of Technology)..... 44
  - 11.4. Model Engineering Solutions GmbH..... 44
  - 11.5. Scania ..... 44
  - 11.6. University of Hildesheim..... 45
  - 11.7. Partner A ..... 45
  - 11.8. Partner B ..... 46
  - 11.9. Partner C ..... 46
  - 11.10. Partner D..... 46
  - 11.11. Partner E..... 46
  - 11.12. Partner F..... 47
  - 11.13. Partner G..... 47
- REFERENCES ..... 48

## Acronyms

BU	Business unit
IIoT	Industrial Internet of Things
LV	Low voltage
MV	Medium voltage
HV	High voltage
LoC	Lines of code

## 1. Introduction

The present deliverable marks the second stage in developing the project use cases. It follows the definition of the use cases presented in deliverable D2.1 - *Use case textual description & legacy and product line asset base, Requirements with KPI for REVaMP<sup>2</sup> 1.0*, and the findings presented in deliverable 2.3 - *REVaMP<sup>2</sup> 1.0 evaluation report*, updating the Requirements & KPI parts, for REVaMP<sup>2</sup> 2.0.

The original D2.1 deliverable content is complemented with details here, the (partner specific) KPIs are more elaborated and linked to goals of the partners or of the project.

### Structure of the Document

The subsequent sections describe the Use Cases, following this structure:

- §x.1: Use case Description
- §x.2: Goals
- §x.3: KPIs
- §x.4: Requirements

Finally, the last section (chapter 11) gives the overall project KPIs.

## 2. ABB

### 2.1. Use case Description

The ABB use case is a software platform for smart motor controller (so called drives) for electric motors (LV, MV and HV). These motor controllers are used in mining, ski lifts, big industry automation processes as well as in solar and wind turbines. Due to this wide range of applications there are a lot of variants, configurations and other variability aspects to handle with. So far, they are managed in a clone-and-own manner with a not very modular software architecture. The platform consists of about 1.2 MLoC and have more than 50 repositories representing different product families (LV, MV and HV). Each of these repositories was created in a clone-and-own manner. In order to follow the trend of digitalization and IIoT makes it necessary to move the current platform to a more flexible and variable architecture. Especially if you think on going to multi-core, digitalization or adding even more features to the platform.

### 2.2. Goals

Deliverable 2.1 presented a first version of the goals of ABB within ReVaMP<sup>2</sup>. Here, the goals are slightly extended and adjusted compared to Deliverable 2.1 due to the learning of SPL in praxis as well as discussions with the BU and presenting the first results of ReVaMP<sup>2</sup> to them.

Overall goals:

- Support development of methods to integrate legacy C/C++ code semi-automatically in a SPL
- Support development of methods for merging several clone-and-own repositories in a common SPL
- Identify features and components in legacy code
- Get a first beta version of a possible tool chain supporting feature identification, documentation and tracing
- Quantify the effort and costs for introducing an SPL in a present code base

Iteration 1:

- Identify and document features and components in legacy code
- Provide methods for (semi)automatically merging similar features
- Get a first draft of a process how to build up an SPL from a legacy code base, especially identify features and locate them in the code base
- Provide a lightweight method to document features in a code base
- Get first tools supporting this process

Iteration 2:

- Quantify the effort and costs for introducing an SPL in a present code base
- Get first feature trees or feature models from a present C/C++ code base
- Get methods for merging clone-and-own repositories
- Get methods for merging feature models
- Extend feature identification and documentation by adding feature dependencies and express their dependencies in constraints

### 2.3. KPIs

Due to the review of the KPIs and feedback from the BU receivers on the first intermediate results, we adjusted the KPIs. For all KPIs we estimate 0% as baseline reflecting the status before

ReVaMP<sup>2</sup>. The reason for that is just the fact that ABB started to investigate in the direction of SPLE with this specific use case at the project begin of ReVaMP<sup>2</sup>. Hence, there was no awareness for SPLE in that special use case.

**KPI 1 Feature Identification and Location in small code base**

In “M24” identifying and locate 8 features (50%) in the smart motor controller example provided by ABB. In “M36” identify and locate all features = 15 (100%).

**KPI 2 Feature Model generation**

“M24” run on a small code base (~ 20K LoC) and produce its feature model with 30% of feature dependencies.

“M36” run on a large code base (1.2M LoC) and produce a feature model with 70% of the feature dependencies.

**KPI 3 Feature Identification and Location in production code**

“M24” identify 25% of the features with a scattering degree (SD) > 65 in a large code base (1.2 MLoC). “M36” identify 50% features with a SD > 65 in a large code base (1.2 MLoC). The scattering degree of 65 was chosen due to Passos et. Al. [1], where the recommendation is given for optimal feature properties, like scattering degree or tangling degree. Due to that we chose 65° as a challenge for identifying “real” scattered features in the code.

**KPI 4 Feature Visualization**

“M24” visualize 50% of the features on large set (1.2M LoC) in an appropriate (understandable and easy) way so that the feature dependencies can be analysed “M36” visualize 100% of the features on large set (1.2M LoC)

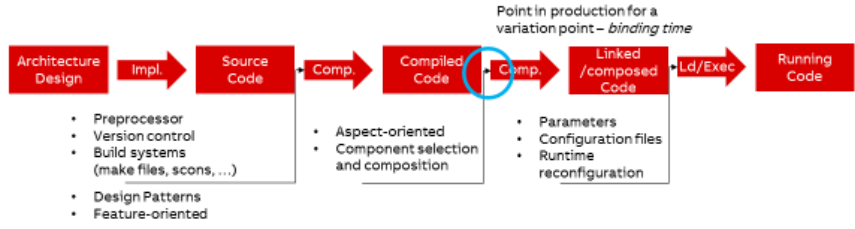
**2.4. Requirements**

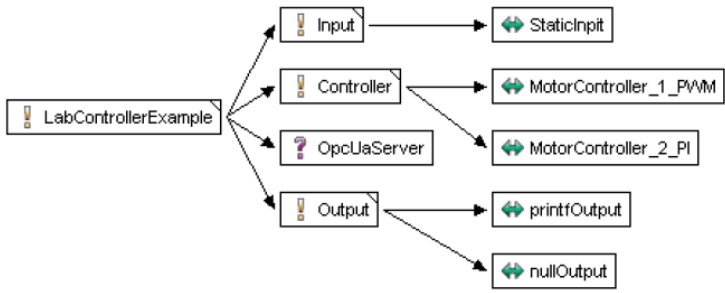
Req. ID	Object Text	Priority
1	The toolchain shall extract variability information from SCON files, see:	High



	<pre> Import('env')  mainSources = ['application.c']  <i>#include path for headers</i> env.Append(CPPPATH=['input', 'controller', 'output'])  if ('OPCUA' in env['CPPDEFINES']):     mainSources.append('applicationOpcua.c')     env.Append(CPPPATH=['opcua']) if ('USERLOCK' in env['CPPDEFINES']):     mainSources.append('cc//cc_crypt_api.h')     mainSources.append('cc//cc_crypt.c')     env.Append(CPPPATH=['userlock'])  mainObjects = env.CompileSources(mainSources); Return('mainObjects') </pre>	
2	The toolchain shall provide methods to localize and extract scattered features from different source code repositories. Example for scattered features are features with scattering degree of >65.	Medium
3	The toolchain shall provide methods or at least guideline how to merge localized features in different clone-and-own repositories.	High
4	The toolchain shall be designed in a way so that existing build systems can be used with the extracted variants. In particular it shall be possible to use the REVaMP <sup>2</sup> toolchain for extracting a SPL and managing a SPL but integrate seamlessly in a present build environment to avoid the "not yet another tool discussion".	High
5	The toolchain shall provide a visualization of the SPL. The challenge here is to provide a proper way to visualize hundreds/thousands of features and variants.	Medium
6	Beside software, the toolchain shall integrate also system variability aspects like having features implemented in SW or HW and how such a setup can be managed.	Low
7	The toolchain shall support Windows / Linux.	Medium
8	The toolchain shall provide a lightweight method to document features within source code, files and folders. Especially documentation and tracing of features over the code should be realized with a small overhead.	Low
9	The toolchain should provide late binding for parametrization	Medium
10	The toolchain shall provide possibilities for requirements analysis in order to decide if it is more beneficial to extend a current variant or create new one. Hence dependencies between features and requirements shall be supported.	High
11	The toolchain shall not increase the test effort. It shall reduce or keep the test effort stable in comparison to the test effort before introducing an SPL.	Low

12	The feature extraction methods integrated in the toolchain shall be able to handle global variables and their dependencies.	High
13	The toolchain shall be able to extract and handle more than 2500 global variables and to classify them to different features.	High
14	The toolchain shall be able to analyse 1 MLoC-Code and to extract features from it in < 10 min.	Medium
15	The toolchain shall be able to identify dependencies between CPP defines which are not documented	Medium
16	The toolchain shall be lightweight and used with small training effort.	High
17	The toolchain shall be able to trace features over the code and generate documentation out of it (e.g. combination with Doxygen)	Medium
18	<p>The toolchain shall be able to identify defines in build files (scons files) and track their dependencies to avoid generating invalid variants or help to track the reason (or illegal cppdefine dependency) why a variant cannot be build</p> <pre> env.Append(CPPDEFINES = 'TARGET_NAME=' + env['TARGET_NAME']  env.Append(CPPDEFINES = 'BT_FL')  env.Append(SAFEFAT_CONFIG = "config") env.Append(SAFE_CONFIG = "config1")  env.Append(CPPDEFINES = 'PARTABLE_EXAMPLE_ENA') env.Append(CPPDEFINES = 'EVERYTHING_ALL') env.Append(CPPDEFINES = 'PAR_PRO_ALL') env.Append(CPPDEFINES = 'TOC_PRO_FLOG_ENA') env.Append(CPPDEFINES = 'TOC_PRO_TC_LC_PARAMETERS_ENA') env.Append(CPPDEFINES = 'TC_CONTROL') env.Append(CPPDEFINES = 'TOC_PRO_MALDEN_ENA') env.Append(CPPDEFINES = 'SLAVE') env.Append(CPPDEFINES = 'UNICODE_ENA') env.Append(CPPDEFINES = 'TOC_PRO_APPL_PRG_ENA') env.Append(CPPDEFINES = 'TOC_PRO_FDATA_ENA') env.Append(CPPDEFINES = 'TOC_PRO_PSL1_ENA') env.Append(CPPDEFINES = 'TOC_PRO_SO_ENA') #env.Append(CPPDEFINES = 'TOC_PRO_EN_ENA') #env.Append(CPPDEFINES = 'TOC_PRO_ETH_TOOLCHANNEL_ENA') #env.Append(CPPDEFINES = 'TOC_PRO_ETH_INTERACTIVE_ENA') env.Append(CPPDEFINES = 'TOC_PRO_FENA_TOOLCHANNEL_ENA') env.Append(CPPDEFINES = 'TOC_LANG_BASIC_ENA') env.Append(CPPDEFINES = 'TOC_PRO_DWARF_ENA') env.Append(CPPDEFINES = 'TOC_PRO_AP_ENA') env.Append(CPPDEFINES = 'TOC_PRO_DK_SUPPORT_ENA') env.Append(CPPDEFINES = 'TOC_PRO_VIRTUAL_DATA_FILE') #env.Append(CPPDEFINES = 'TOC_PRO_PD_SAVE_ENA') env.Append(CPPDEFINES = 'TOC_PRO_CHECKSUM_ENA') env.Append(CPPDEFINES = 'LOPT_SSH_IS_DEFPARSE_CRYPT_ENA') env.Append(CPPDEFINES = 'TOC_PG_4PWA_UPDATE_ENA') </pre>	High
19	Toolchain shall have eclipse cdt and visual studio integration	Medium
20	Toolchain shall be able to handle variability at different variation points, build time, configuration time, boot time (licenses), runtime	High

	<p><b>SPL</b> Variability</p> <ul style="list-style-type: none"> <li>• Ability to vary core assets depending on product context</li> <li>• so called <i>Variation points</i> <ul style="list-style-type: none"> <li>• <i>Early binding or late binding</i></li> </ul> </li> </ul>  <p>©ABB June 23, 2017   Slide 8 <span style="float: right;">ABB</span></p>	
21	<p>Toolchain shall consider and be able to handle a variable system which consists of two subsystems, a common platform and a product specific system. Each of these subsystems are variable but it has to be managed that some variants of the common platform are not valid with some SW variants at the product level. The managing between these systems should be loosely coupled because the common platform will also be used by other product systems. --&gt; In particular for both subsystem a SPL exist and should be handled but they are somehow related. Similar to git submodules. Such a mechanism should be reflected within REVaMP<sup>2</sup> toolchain.</p>	Medium
22	<p>The toolchain shall provide some feedback mechanism so that identified and localized features can be iterated by domain experts or developers</p>	Medium
23	<p>The toolchain shall be able to handle variability add different abstraction levels as well as the representation of it. That means not for all level of variant management a feature tree is a suitable solution for managing variants. There should be an option for also lightweight variant management.</p>	High
24	<p>The toolchain shall provide methods, guidelines and support for merging features from different repositories.</p>	High
25	<p>The toolchain shall be able to extract/identify features which are only referenced by jump tables, e.g. <code>const FNC_TYPE callback_array[2] = {callback_implementation3, callback_implementation4};</code> Often implemented in embedded or hard real time critical environments.</p>	High
26	<p>The toolchain shall be able to intercept the build process in order to identify also linked libraries to particular features, etc.</p>	Medium
27	<p>The toolchain shall be able to fully extract at least 90% of all features hidden in ABB use case.</p>	High

		
26	Support of SCON build system, in particular the toolchain shall analyse SCON (python files) and extract variability information (e.g. variation points, ...) from it and also generate SCON files for particular variants so that they can be built with the SCON build system	High
28	The toolchain shall be able to handle .c, .h, .cpp files. Also generates particular .c, .h, .cpp files for certain variants; remove variant information out of the files to reduce the memory footprint of specific product variants.	Medium
30	The toolchain shall also be able to extract / identify textual but none source code files as necessary for a feature (e.g. xml files) and add them to the identified feature.	Medium
31	The toolchain shall provide interfaces to support exiting test environments, like a build server, nightly test builds and test routines and Jenkins server for test documentation.	High

### 3. ROBERT BOSCH

#### 3.1. Use Case Description

The Bosch use cases relate to a software product line for Automotive Engine Control Units (ECU) developed by the business unit Powertrain Solutions - Electronic Controls (PS-EC). The application areas for the engine control are diverse: from two-wheelers (scooters, motorbikes), through passenger cars, delivery vehicles and trucks, up to off-road applications such as construction machinery and stationary industrial engines. This involves supporting a huge variety of product functionalities and an even bigger number of system and software variants to meet all the customer needs. The PS-EC engine control software is developed as a software product line.

**Use Case 1: Branching.** Identify groups of similar software component branches and extract the alternative history trees out of the configuration management system (SCM). Group the trees following root alternatives and by customer numbers. Based on this, identify differences in relevant pre-processor feature constants between branches, and identify features and their constraints that differ between branches.

**Use Case 2: Identification of variability dependencies in artefacts.** Parse the project artefacts, evaluate the system constant based presence condition information and extract the variability dependencies. These presence conditions are used for selection of SW components in SCM and for the conditional compilation in C files and configuration files.

**Use Case 3: Variability analysis.** Perform analyses and consistency checks on the variability information. For example, by relating artefact variability information, SCM information, and variability models, dead or superfluous artefact content can be identified. Past configuration data can be used to find unused components, component branches, feature constants, feature constant values, as well as to identify correlations between usages of components, branches, feature constants, and feature constant values. This can in turn be used to update variability models. Finally, a validation between the dependencies implemented in the artefacts and the variability model can be performed.

**Use Case 4: Variability visualization.** Optionally, visualizations of the variability information, supporting both modelling and configuration tasks, can be developed and evaluated.

**Use Case 5: Configuration support.** This use case concerns the improvement of project configuration and the reduction of configuration effort. This includes support for creation of new variability models and update of existing ones, based on the information provided by other use cases. In addition, analysis of the variability model and the statistical data can improve the configuration process by reducing the amount of needed configuration decisions through identifying the high-impact features, and by detecting untypical configurations and showing recommendations.

Ideally, all five use cases should be fulfilled. For use cases 1, 2, 3 and 5 Bosch knows of potentially suitable approaches, developed by the REVAMP2 partners as well as the general software product lines community, which can be applied and evaluated to satisfy these use cases with high probability. In the Use Case 4 (visualization) however, Bosch does not know of an approach that could bring benefit at the required scale (thousands of variability-related elements). It is therefore hard to predict whether the use case can be successfully fulfilled thanks to a new research during the project duration. This results in the classification of that use case as optional.

### 3.2. Goals

**Overall goal:** Support development and evaluation of technical solutions for improving efficiency and correctness of variability management activities (software implementation, feature modelling, and configuration).

**Detailed goals:**

- **Branching and merging (Bosch UC1).** Support development and evaluation of approaches to identify similar clone-and-own component branches, assess their merge potential, and merge them into reusable implementations. Especially, the approaches should be able to perform the following tasks:

Identify features which differ between branches

Identify differences in relevant pre-processor feature constants between branches

Identify differences in feature-based constraints between branches

- **Variability realization using pre-processor (Bosch UC2, UC3).** Support development and evaluation of approaches to analyse the pre-processor variability in the source code and other artefacts (at Bosch, non-code artefacts such as documents, data files, models and build scripts can also be annotated with feature constants). Especially, the approaches should be able to perform the following tasks:

Identify the pre-processor feature constants relevant to the analysed artefacts

Identify dependencies between the feature constants encoded in the artefacts

Perform validation between the dependencies implemented in the artefacts and the variability model

Identify dead feature artefact content (e.g., source code with non-satisfiable presence condition). The presence condition might be non-satisfiable because of variability model dependencies, or because of dependencies existing in other artefacts and evaluated earlier (e.g., build scripts, branch selection condition).

Ensure that the product instance-specific artefacts contain no content belonging to non-selected features (by-catch).

- **Statistical analysis of variability (Bosch UC3, UC4).** Support development and evaluation of approaches to statistically evaluate product line variability based on component usage and feature configuration data. Especially, the approaches should be able to perform the following tasks:

Identify unused variability: unused components, component branches, feature constants, feature constant values.

Identify correlations between usages of components, branches, feature constants, feature constant values. For example, a correlation can be that some feature constant values are only used for some branches, or that some feature values are always configured together.

- **Efficient configuration (Bosch UC5).** Support development and evaluation of approaches to increase the efficiency and correctness of the variability configuration tasks. Especially, the approaches should be able to perform the following tasks:

Support the creation of accurate and comprehensive variability models and identification of feature hierarchies (using the methods developed in the above goals)

Support configuration tasks based on the variability models, feature hierarchies and statistical data, for example by greying out unnecessary feature constants and values, providing default or typical values, indicating and applying dependencies, showing recommendations, detecting untypical configurations.

In addition to the above point, reduce the amount of needed configuration decisions by identifying the high-impact features and presenting them in the beginning of the configuration process. The

high-impact feature identification can be based on the variability models, feature hierarchies and the statistical data.

- **Visualization (Bosch UC4).** Optionally, visualizations of the variability information, supporting both modelling and configuration tasks, can be developed and evaluated.

### 3.3. KPIs

All the below KPIs relate to the ReVaMP<sup>2</sup> toolchain developed to support the use cases described in the previous section. Hence, for all KPIs the initial baseline value (before the start of the ReVaMP<sup>2</sup> project) is 0%.

**KPI 1 Scalability:** The developed toolchain should scale to the size of Bosch software. There are two scalability metrics depending on the analysis type: code size and number of alternative variants.

The analysis should scale to the code size of a complete product (>2 MLOC per product), to the number of branches in the most variant-rich components (>20), or both.

Measurement: calculate the average of target level achievement percentage for the two metrics according to the following table:

Goal achievement	0%	20%	40%	60%	80%	100%
Product size in LOC	Insufficient (<50 KLOC)	Component (50 KLOC)	Large component 100 KLOC	Component group 300 KLOC	Small product 1 MLOC	Any product > 2 MLOC
Supported branches	1	2	5	10	15	20

Target levels:

M24	50% (concrete metric values depend on the analysis type)
M38	100% (>2 MLOC, >20 variants)

**KPI 2 Integration:** All tools used for the Bosch use case should be compatible with the Bosch environment or at least Bosch input and output formats.

Measurement: amount of compatible tools / total amount of tools.

Target levels:

M24	50% tools used by Bosch UCs are compatible
M38	100% tools used by Bosch UCs are compatible

**KPI 3 Automation:** The preparation of an analysis using the ReVaMP<sup>2</sup> toolchain should require at most 10 minutes of manual effort.

Measurement according to the following table:

Goal achievement	0%	20%	40%	60%	80%	100%
Preparation	8 h	4 h	2 h	1 h	30 min	<10 min

Goal achievement	0%	20%	40%	60%	80%	100%
effort						

Target levels:

M24	20% < 4 h effort
M38	100% <10 min effort

**KPI 4 Result quality:** Identification of system constants used by an artefact, implemented dependencies between system constants, dead artefact fragments, unused artefacts, parameters and values, performed by the ReVaMP<sup>2</sup> toolchain should be correct and complete, with no false positives and no missed elements:

- 100% precision (proportion of correct reported elements within all reported elements)
- 100% recall (proportion of correct reported elements within all relevant elements in the artefact)

Measurement: for a given tool, average the precision and recall values. For the KPI, average the values of all tools. If the measurement for a given tool is not possible (e.g. due to no alternate method of determining the ground truth), the tool-specific value is an estimation.

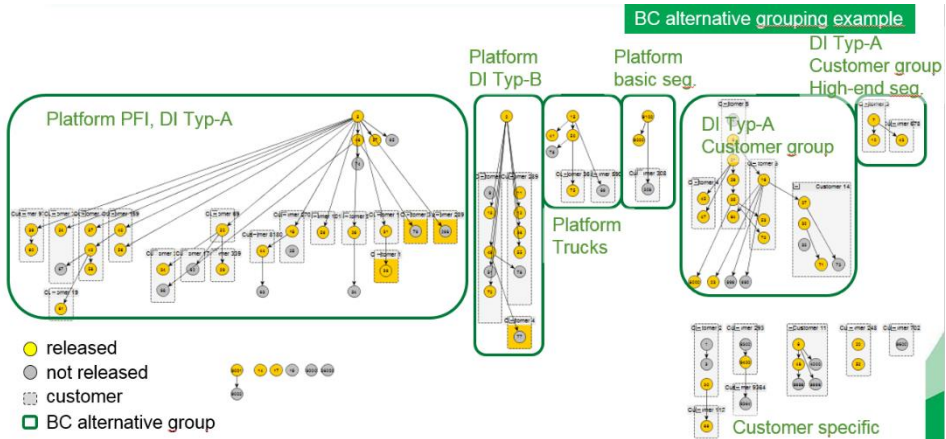
Target levels:

M24	80% result quality calculated as above
M38	100%

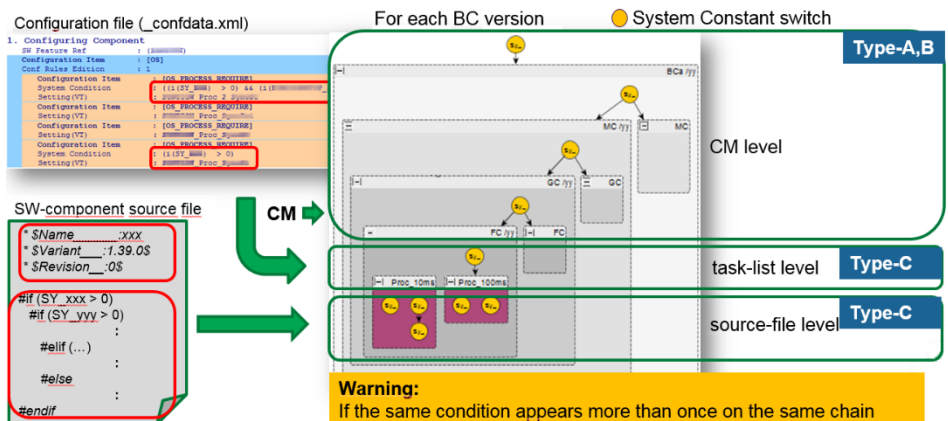
**Further comments**

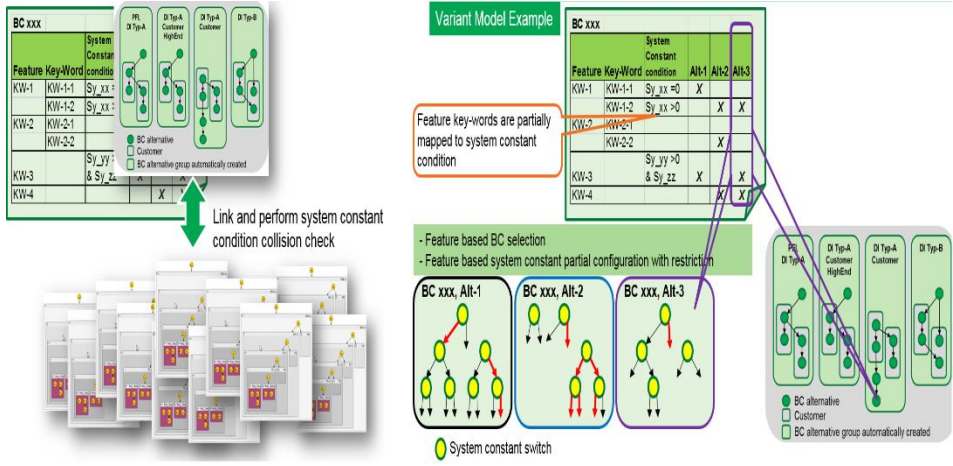
Initially, Bosch defined a further KPI 5 Productivity to assess the expected benefits of applying the ReVaMP<sup>2</sup> toolchain results in product development. However, the KPI is currently not assessable due to the difficulty of isolating the effect of ReVaMP<sup>2</sup> toolchain from the effects of other improvement measures performed in parallel.

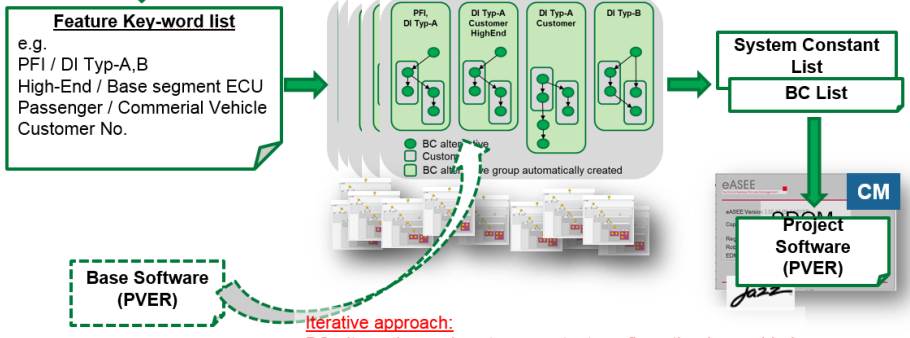
**3.4. Requirements**

Req. ID	Requirement Text	Priority
1	<p><b>Analysis of BC (software component) alternative branches</b></p>  <p>The diagram illustrates 'BC alternative grouping example' with several components: Platform PFI, DI Typ-A; Platform DI Typ-B; Platform basic seg.; Platform Trucks; DI Typ-A Customer group; High-end seg.; and Customer specific. A legend indicates: yellow circle for 'released', grey circle for 'not released', dashed box for 'customer', and green box for 'BC alternative group'.</p>	High



Req. ID	Requirement Text	Priority
1.1	The REVAMP approach should be able to identify similar clone-and-own component branches and asses their merge potential.	High
1.2	The approach should be able to identify features which differ between branches.	High
1.3	The approach should be able to identify differences in relevant pre-processor feature constants between branches.	High
1.4	The approach should be able to identify differences in feature-based constraints between branches.	High
1.5	The approach should be able to get the BC alternative information from Bosch SCM repositories SDOM and IBM ALM.	High
<b>2</b>	<p><b>Extraction and analysis of pre-processor variability</b></p> 	<b>High</b>
2.1	The approach should be able to analyze the pre-processor variability in the source code as well as other artefacts (at Bosch, non-code artefacts such as documents, data files, models and build scripts can also be annotated with feature constants).	High
2.2	The approach should be able to identify the pre-processor feature constants relevant to the analysed artefacts.	High
2.3	The approach should be able to identify dependencies between the feature constants encoded in the artefacts.	High
2.4	The approach should be able to perform validation between the dependencies implemented in the artefacts and in the variability model.	High
2.5	The approach should be able to ensure that the product instance-specific artefacts contain no content belonging to non-selected features (by-catch).	High
2.6	The approach shall provide a generic interface so an end user can add own specific parsers for other file formats.	High

Req. ID	Requirement Text	Priority
3	<p><b>Analysis of variability over multiple sources and artefact types</b></p>  <p>The diagram illustrates the process of analyzing variability. It starts with a table for 'BC xxx' containing feature key-words (KW-1 to KW-4), system constants (Sy_xx, Sy_yy, Sy_zz), and conditions. This is linked to a 'Variant Model Example' table and further to feature-based BC selection and system constant partial configuration. A legend defines symbols for BC alternative, Customer, and BC alternative group automatically created.</p>	High
3.1	The approach should be able to aggregate and relate the extracted constraints over multiple sources and artefacts, for example branch selection conditions, build scripts, source code, and runtime parameters.	High
3.2	The approach should be able to identify dead feature artefact content (e.g., source code with non-satisfiable presence condition). The presence condition might be non-satisfiable because of variability model dependencies, or because of dependencies existing in other artefacts and evaluated earlier (e.g., build scripts, branch selection condition).	High
3.3	The approach should be able, based on the artefacts and past configuration data, to statistically evaluate product line variability and identify unused variability: unused components, component branches, feature constants, feature constant values.	High
3.4	The approach should be able, based on the artefacts and past configuration data, to statistically evaluate product line variability and identify correlations between usages of components, branches, feature constants, feature constant values. For example, a correlation can be that some feature constant values are only used for some branches, or that some feature values are always configured together.	High
4	<b>Variability visualization</b>	High
4.1	The system shall be able to graphically represent the entire variability model, both for modelling and configuration purposes, in an easy to use manner.	High
4.2	The system shall support the assisted manual configuration of variant switches (BC alternatives, system constant configuration) using the visualization.	High
4.3	Further visualizations of the variability information, supporting both modelling and configuration tasks, can be developed and evaluated.	Medium

Req. ID	Requirement Text	Priority
5	<p><b>Variability modelling and configuration</b></p> <p><b>Pro-active approach</b> According to the selected feature, the tool provides a list of compatible BC alternatives. Once the BC alternative is selected, the corresponding list of system constant is provided for manual configuration. The system constants which are already configured through automation (Use-Case 3) are restricted from manual configuration.</p> <p><b>UseCase 1</b></p>  <p><b>Feature Key-word list</b> e.g. PFI / DI Typ-A,B High-End / Base segment ECU Passenger / Commercial Vehicle Customer No.</p> <p><b>System Constant List</b> BC List</p> <p><b>Project Software (PVER)</b></p> <p><b>Base Software (PVER)</b></p> <p><b>Iterative approach:</b> BC alternative and system constant configuration is provided</p>	High
5.1	The approach should increase the efficiency and correctness of the variability modeling and configuration tasks.	High
5.2	The approach should support the creation of accurate and comprehensive variability models and identification of feature hierarchies (using the methods developed in the above requirements).	High
5.3	The approach should support configuration tasks based on the variability models, feature hierarchies and statistical data, for example by greying out unnecessary feature constants and values, providing default or typical values, indicating and applying dependencies, showing recommendations, detecting untypical configurations.	High
5.4	The approach should reduce the amount of needed configuration decisions by identifying the high-impact features and presenting them in the beginning of the configuration process. The high-impact feature identification can be based on the variability models, feature hierarchies and the statistical data.	High

## 4. MACQ

### 4.1. Use case Description

Macq provides a use case in the road traffic control domain with four existing generations of traffic cameras (hardware and software for license plate recognition and vehicle classification) and the corresponding back-end systems.

Software matches the available and changing parallelism in the available hardware and will be “revamped” into a new product line.



### 4.2. Goals

#### High-Level Project Approach:

- **Research methodologies** that allow **feature** and **asset determination** and extraction
- Test the **tools to extract** assets and features from legacy
- Test the **tools** that allow the **creation of new revamped generations** from legacy assets
- Provide a **use case** with different generations of embedded hard and software that have different **parallel architectures**.

**Goal:** With the ReVAMP<sup>2</sup> tools we will revive a lot of software for our next generation of cameras and recombine separated backend systems into the new Smart Mobility product line.

Our contribution to WP2 (use case) will build upon WP3 (Methodologies), WP4 (Asset extraction and visualization) and WP7 (Tool-chain integration).

New generation of camera product line and Smart Mobility backend system that reuses assets from previous generations.

Tools that support asset visualization and variability decisions for both hardware and software. Starting from the VARIES concepts Variability drivers, Variability decisions, Variability stakeholders, Portfolio shaping. We will develop methodologies that allow to describe the rationale behind the decisions and their impact and methodologies to evaluate if the constraint that lead to those decisions are still true in an evolving context or could be lifted. There will be a focus on the interaction of the decisions and constraints.

### 4.3. KPIs

**KPI1.1:** 100% of the concerned legacy should be analyzed and documented if we can harvest assets from them.

Since Macq in the last 20 years changed several times its versioning system we need to start with a list of all repositories. Then for each repository list the legacy projects/modules. For each of the modules ask the domain experts about their relevance.

**KPI1.2:** 70% of the legacy should be analyzed by the tools.

Try to analyze the assets that are identified relevant in KPI1.1

M24: First evaluation analysis %

M30: Intermediate evaluation analysis %

M36: Final evaluation analysis %

**KPI2:** At the end of the project we aim to have a value of x€ or the equivalent of y person years of extracted assets.

Evaluate extracted assets as PMs that can be transformed into a corresponding € value

**KPI3:** Use the methodology tools in at least 12 meetings.

There are three kind of meetings that can be used:

- Monthly sales meeting,
- Two weekly sprint review meetings for two teams
- Two weekly stakeholder meeting

**KPI4:** Description of the variability rationale of the new product line with the ReVAMP<sup>2</sup> methodologies and tools

**KPI5:** 30% increased efficiency in communication between stakeholders and developers.

M24: Baseline interview stakeholders and developers to score the communication efficiency

M30: Intermediate interview

M36: Final interview

### 4.4. Requirements

Req. ID	Object Text	Priority
1	Be able to capture/identify features and assets	Undefined
2	Be able to (semi-)automatic extract assets.	Undefined
3	Be able to combine assets into a new generation of a product line.	Undefined

4	Be able to identify the rationale behind an asset and variants using variability drivers and decisions.	Undefined
5	Be able to quantify how the interaction between hardware and software design decisions influence how legacy assets can be combined into new product generations with a lot of variants.	Undefined

## 5. SAAB

### 5.1. Use case Description

In the development of Gripen E/F simulators are used extensively. These range from completely software based to simulators with hardware in the loop simulators. The scope of the SAAB use case is to demonstrate the feasibility of automating the selection of a configuration of the Gripen E/F aircraft and the corresponding correct configuration for instantiation in a simulator.

### 5.2. Goals

The objective of the use case is to automate the creation of a particular Gripen E/F aircraft configuration tailored for a particular simulator.

Multiple simulators with different capabilities are in use in the development of Gripen E/F. Each simulator will be part of a larger simulator product family with its own set of defined features. Likewise, the Gripen E/F is a product family which will have its own set of features.

The aim is to build a configurator system where a user – with defined credentials - can select a desired Gripen E/F product configuration and select the target simulator. Based on the selection build scripts will be created automatically. Moreover, this means that models and components of the aircraft as well as the simulators will be selected based on the credentials of the user (The user shall not be able to build a configuration containing features that the user is not allowed to see/explore).

### 5.3. KPIs

The following KPIs are defined for the SAAB use case. All KPIs are rated at 0% at the start of the project. Assessment of progress for the KPIs is subjective as they relate to the adoption of product family concepts within the Gripen E/F project. 0% implies no adoption and 100% does of course mean complete adoption but setting the values in between such that they have an objective meaning is hard. For some KPIs intermediate measures have been identified. Actual measures that does not fully conform to the defined thresholds shall be interpreted as there is significant progress towards a measure or that a threshold has been attained and there is progress towards the next.

For KPI 1.1 and 1.2 there is an overarching goal to establish a unified methodology framework for Product Family Management throughout the Aeronautics Business Unit.

- **KPI1.1 Methodology Definition:** Existence of a methodology definition within the SAAB Aeronautics quality management system. Measure M18. This is measured as follows
  - 50% means that there are basic methodology definitions in Gripen E/F working instructions
  - 80% means that methodology instructions are in existence in the Gripen E/F SEMP (Systems Engineering Management Plan).
  - 100% means that the methodology fully introduced in SAAB Aeronautics Quality Management System.
- **KPI1.2 Methodology adoption:** Adoption of the methodology for Gripen E/F and Simulator development. Measure M24. This measure is divided into two parts each weighted at 50% of the total measure. This is measured as follows:
  - 0% No adoption of the methodology within the Gripen E/F project.

- 50% Complete adoption within Gripen E/F or simulator development respectively (All aspects of the methodology is in active use).
- Values between 0-50% represents gradual adoption of the methodology within each organisation

**KPI2 Artefact identification:** Demonstrate capability to identify options within a product family based on development artefacts. Options for development artefacts will be identified for both Simulators and the Gripen E/F products. This is measured as follows:

- 0-50% for the identification of simulator artefacts and
- 0-50% for Gripen E/F artefacts

50% for the respective group is assumed to be reached when there is a released feature model for the group. 20-30% is awarded for mature work in progress.

**KPI3 Configuration definition:** Capability to define a product configuration based on selection of options within the product family definition. This is measured as follows:

- 50% for the ability to define a configuration in a tool that is able to interact with build/PDM systems
- 100% for the ability to define configurations when the credentials of the user defining the configuration is used to identify the set of features that should be visible to the user.

**KPI4 Configuration instantiation:** Capability to instantiate of product configuration based on a defined selected set of features. This is measured as follows

- 50% for the ability to create a build recipe for interaction with a build/PDM system
- 100% for the actual ability to create a correct build recipe

#### 5.4. Requirements

Req . ID	Object Text	Priority
1	The system for enabling configuration shall only display those variation point values that are valid for a specific user (credentials based). I.e., as a user it shall only be possible to view those variation points, and variation point values that the particular user should be aware of.	High
2	It shall be possible to capture confidentiality and export control constraints and use such parameters for defining configurations for use in specific configurations for partner/country usage.	High
3	The administrator of the Toolchain shall not automatically be granted access to all information managed by the toolchain	High
4	It shall be possible to manage multiple product lines within the tool chain	High
5	It shall be possible to capture relationships and attributes on relationships between elements in two product lines	Medium
6	The toolchain shall be built on existing standards, e.g., OSLC (Open Services for LifeCycle collaboration)	Medium
7	It shall be possible to obtain and synchronise user identities from a centralised system, i.e., LDAP	Medium



## 6. SCANIA

### 6.1. Use case Description

A modern Scania truck is controlled by a distributed control system consisting of 20-80 interconnected ECUs (Electronic Control System). Each control system is built in many variants (many configurations). As an example, even a simple fuel level indication system in a Scania truck has 24000 variants! This means that only a very small portion of all possible variants are possible to verify by testing in real vehicles. The goal of the whole verification process is however to cover as many as possible of the variants. As a partial solution to this challenge, overall aims of Scania in the ReVaMP<sup>2</sup> project is to:

- Transform current complex variability model into a feature diagram
- Check the consistency of presence conditions against the established variability model
- Perform variability-aware C-code verification against natural language / semi-formal/formal functional/safety requirements

### 6.2. Baseline

Before the beginning of ReVaMP<sup>2</sup> WP4 and WP6, the extraction of variability models from the implementation and analysis of PL assets with respect to variability models was manual. The input for the work in WP4 and WP6 was a list of sources that contain the necessary information to automate the mentioned analyses. The implementation of extractors and analysis operations was initiated in WP4 and WP6.

### 6.3. KPIs

*In WP4, Scania will develop tools/scripts for extracting variability models from implementation.*

**KPI #1:** Number of different use cases for which target automation accuracy was attained. The goal is to provide automation for the following use cases: 1) extraction of variability model from product line management tool 2) extraction of variability model from end-of-line (EOL) tools 3) extraction of variability model from software implementation 4) unification of the extracted data into an overall variability model. The KPI is to be evaluated with respect to this goal, i.e. if 1/4 of the use cases has been automated, then the goal has been reached to 25%.

*In WP6, Scania will attempt to analyze and verify variability models with respect to specifications.*

**KPI #2:** Number of different use cases for which target automation accuracy was attained. The goal is to: 1) analyze completeness and consistency of requirements 2) analyze correctness of automotive safety integrity levels (ASIL) w.r.t. requirements break-down structure 3) analyze consistency and completeness of variability conditions for requirements 4) given C-code, requirements and their variability conditions, automatically annotate the given C-code for verification 5) perform variability-aware formal verification of the C-code w.r.t. annotations. The KPI is to be evaluated with respect to this goal, i.e. if 1/5 of the use cases has been automated, then the goal has been reached to 20%.

**KPI #3:** Number of different asset classes (e.g., requirements, system models, software models, hardware models, code, tests, documentation) for which target automation accuracy was attained. The goal is to automate analysis of: 1) requirements 2) ASILs of requirements 3) variability conditions of requirements 4) overall variability model 5) C-code implementation. The KPI is to be

evaluated with respect to this goal, i.e. if 1/5 of the use cases has been automated, then the goal has been reached to 20%.

**KPI #4:** Number of different asset representation language (e.g., Simulink, SysML, C) for which target automation accuracy was attained. The goal is to provide automatic analysis capabilities for the following languages: 1) C code 2) Domain-specific product line management language 3) Domain-specific language for specifying validity (variability) conditions of requirements. The KPI is to be evaluated with respect to this goal, i.e. if 1/3 of the use cases has been automated, then the goal has been reached to 33%.

#### 6.4. Requirements

Req . ID	Object Text	Priority
1	The toolchain shall have the capability to manage the following information elements <ul style="list-style-type: none"> <li>- Requirements at several levels of hierarchy</li> <li>- System models and interfaces, including variability parameters</li> <li>- C code</li> </ul>	High
2	The toolchain shall support extraction of models + variability from implementation assets	High
3	The toolchain shall support analysis of models in the form of queries, e.g. "what components are part of this configuration"	High
4	The toolchain shall support specification of informal, semi-formal, and formal requirements and their relation to variability (e.g. parameters)	Medium
5	The toolchain shall support semi-formal and formal variability-aware verification of requirements	Medium
6	Where possible the toolchain shall support publishing and consuming data in accordance with Linked Data, e.g., using OSLC (Open Services for LifeCycle collaboration)	Medium

## 7. SIEMENS

### 7.1. Use case Description

Siemens Industry Software NV (SISW) is an engineering innovation company and leading provider of test and mechatronic simulation software and engineering services in the automotive, aerospace and other advanced manufacturing industries. With multi-domain and mechatronic simulation solutions, SISW addresses the complex engineering challenges associated with intelligent system design and MBSE. In the automotive industry in particular, product lines are common; but the design of them is not trivial and more and more challenges, due to increasing environment regulations, are arising.

For example, the optimal design of a fully-electrical or hybrid drive train highly depends on its load scenarios. However, in an automotive product line this is the key differentiator between the individual products in one line. An optimal product line has therefore the right balance between economic and environmental needs, which only can be achieved by combining multi-physic behaviour simulation together with the configuration challenge in product lines. Additionally, an automotive suspension is a vital system which is employed to sustain the vehicle weight, to isolate the chassis from the apparent vibrations of the road, hence improving the comfort for the passengers, to enhance road holding and maintain the wheels in a correct position on the road.

One challenge faced by SISW clients is to design and compare a significant number of product suspension design variants. The goal of our use case is to develop a structured and automated approach to extract variability models based on different automotive suspension design variants and to use the extracted variability related models to ease the configuration process of new products in the product family.

### 7.2. Goals

The aim of this use-case is to contribute towards machine-aided generative mechatronic design and product line engineering of new products and variants.

The assets of this use case are possible logical architecture models of hybrid vehicle powertrains, and suspension systems. There are many different architectures for such models, and the main goal is to identify functional features which are common to the variants, and the common patterns for variants which contain that feature. The automatically extracted features should then be hierarchically ordered in a CVM-like diagram, and the common elements which correspond to a feature showcased. The automatically generated CVM must support PLE for new assets by (i) using relevant features, (ii) having a small number of features (max 50), and (iii) providing a good hierarchical view of the features.

These are the overall goals:

- Showcase the common element in variants containing an automatically extracted feature.
- Automatically identify relevant functional features from architecture models.
- Map each identified feature to a common pattern.
- Identify dependencies among identified features and/or patterns.
- Automatically synthesise a Central Variability Model based on the identified features and dependencies.

### 7.3. KPIs

SISW's KPIs are focused on the ease-of-use of the ReVAMP<sup>2</sup> toolchain, and in the added value it brings to our PLE approach, as perceived by our (expert) users. Because the ReVAMP<sup>2</sup> toolchain represents a new feature in our tools, the value of the KPIs before the project is not well-defined. However, for the purpose of evaluation, it can be assumed to be 0, since there was no possibility to use this feature before the project. The KPIs we are assessing are as follows:

- Feature to Pattern Mapping
- Relevance of Features
- Identify Dependencies
- Usability of the Model

Because all KPIs for Siemens are based on the usability and added value that the ReVAMP<sup>2</sup> toolchain (BUT4Reuse in particular) brings to our tools, they are challenging to evaluate quantitatively since they depend on the feedback of the users. At this point we are using a grade in a scale from 1 to 10 (1 being not useful and 10 being indispensable), given by the experts who are experimenting with the toolchain. This means that each KPI is calculated by averaging out N user-given values as follows:

$$KPI = \frac{1}{N} \sum_{i=1}^N user\_value_i$$

At a later stage, large usability tests can be conducted so as to ensure results for the KPI value are indeed representative.

### 7.4. Requirements

	<b>Object Text</b>	<b>Priority</b>
1	The toolchain shall extract variability information from XML files describing topological architectures of automotive suspension systems. The particular XML format is initially set as GraphML.	High
2	The toolchain shall locate features over the topological architectures diagrams.	High
3	The toolchain shall discover constraints between features.	High
4	The toolchain shall provide a visualization of the product line previewing the existence of hundreds of features and variants.	High
5	The toolchain shall support Windows.	High
6	The toolchain shall support Linux.	Low
7	The toolchain shall be lightweight and used with small training effort.	High
8	The toolchain shall synthesize feature models.	High
9	The toolchain shall suggest alternative ways to synthesize a feature model.	Medium
10	The toolchain shall be able to be integrated to a build server.	Low

## 8. SOFTEAM

### 8.1. Use case Description

Modelio is the MDE workbench, graphical model editor, with lots of functionalities for different users and markets. For Business Architects we provide solutions based on TOGAF and ArchiMate. For System Architects we propose SysML, MARTE and UML. For developers, we dedicate code generation and reverse engineering tools for Java, C++, C#. All these solutions come with templates for documentation authoring, model transformations, code generation and GUI perspectives. The product line has more than 12 packaged solutions that are ported to 3 major operating systems (Windows, Linux and iOS) for x32 and x64 platforms.

Major challenges:

- Maintaining complex product line.
- Automating product packaging on the fly for ever-growing demand for custom solutions.

Expected benefits:

- Automated extraction of the product line model.
- Custom solution generation tool with composability constraints verification capabilities.

### 8.2. Goals

During the period we detailed and expressed our goal through a set of requirements needed to successfully achieve our use case targeted results:

- "The toolchain shall extract variability model and the PL constraints from the Modelio PL artefacts such as:
  - sets of Eclipse plugins for Modelio commercial and open source solutions
  - build and packaging scripts for Modelio commercial solutions and open source version"

*Through this requirement our goal is to achieve the complete automation of the use case, since everything could be done with only a set of pre-existing versions*

- The toolchain shall provide capabilities to generate / package new variants of Modelio solutions corresponding to constraints given by users.

*Through this requirement our goal is to have a way to specify constraints.*

- The toolchain shall automatically verify the validity of certain composition of Modelio solutions.

*Through this requirement our goal is to check the UML model against the Feature model*

- The toolchain shall provide a methodology for maintaining the PL generation pipeline.

*Through this requirement our goal is to be assured that the PL generation pipeline is maintained*

- The toolchain shall provide capabilities to propagate updates and fixes to the whole Modelio PL seamlessly.

*Through this requirement our goal is that every change needs to be only made once in one tool.*

- The toolchain shall indicate the specific tests for new variants.

*Through this requirement our goal is that the provided toolchain could indicate the specific tests for ne variants*

- The toolchain shall help to document the Modelio PL portfolio.

*Through this requirement our goal is that within our use case the tool chain could help to document the Modelio PL portfolio*

### 8.3. KPIs

The KPIs that we will apply to Modelio use case are:

Unique ReVAMP <sup>2</sup> selling point measured	Indicator	Month		
		M20	M32	M38
Level of variability model <b>extraction</b> automation	Proportion of variability model elements that can be accurately and automatically extracted from legacy input assets on available use case benchmarks	33%	50%	66%
Level of variability model <b>verification</b> automation	Proportion of variability model defects that can be accurately and automatically identified, diagnosed and corrected from variability model, either manually constructed or automatically extracted from legacy assets on available use case benchmarks	33%	50%	66%
Level of domain model and product specific <b>synchronization</b> automation	Proportion of co-evolved domain model elements and product specific model elements that can be accurately and automatically synchronized in available use case benchmarks	33%	50%	66%
Engineering tool integration	Number of base commercial products integrated into interoperable ReVAMP <sup>2</sup> tool-chain	3	5	7
Variability management automation tool integration	Number of advanced variability management PoC integrated into interoperable ReVAMP <sup>2</sup> tool-chain	0	2	3

The ultimate goal of Softeam is to be capable to integrate and bundle a new custom version within 1 day. As the **baseline**, currently, such a process takes 15 days with work done by several engineers.

The first input comes from a management’s decision, it requires time to decide whether we need a new version to suit some client’s needs or if an older build would be enough with the appropriate modules. Then, if a new version is required, we need to define exactly what component will be included in it.

When all the elements are ready, some time is required to fetch every needed component, integrate them manually then make sure that it will work together and does what is needed. Then the new version can be published with its documentation and installation instructions.

Action	Estimated time to market	Effort needed
Decision - Management decision that a new version is needed involving various meetings	2 weeks	4 person/day
Definition - Selection of features done by Project Manager	2 weeks	4 person/day
Integration - Integration and Bundling	1 week	5 person/day

Publication - the support team to put the new package on-line	1 week	2 person/day
<b>Total</b>	<b>6 weeks</b>	<b>15 person/day</b>

With the first version of the ReVaMP<sup>2</sup> toolchain, the steps will still be the same with some improvement in terms of time.

The product line generated with But4Reuse's help provides an overview of all the different components of Modelio, it will simplify the decision-making process and the definition of the needs for a new build.

Most requirements and dependencies will be modelled In the Feature model / 150% model and the tool provide a way to build automatically a new version, so the integration work will be simplified. The effort to be done here will mostly be to check that everything works as expected.

Action	Estimated time to market	Effort needed
Decision	1 week	1 person/day
Definition	2 days	2 person/day
Integration	3 days	3 person/day
Publication	1 week	2 person/day
<b>Total</b>	<b>3 weeks</b>	<b>8 person/day</b>

With this first version of the toolchain, we intend to divide per two the time and work needed to generate a new version of Modelio.

At the end of the project, we aim at changing the way we produce new versions, the decision and definition of the needs will be deported directly to the user via a web interface, where they can choose the features they want in their build with a total effort of less than one day.

The integration will be automated by the product lines tools by checking the modelled constraints, the new version is then built and can be instantly downloaded by the user.

With this toolchain, Softeam will only need to update the feature model, the components included in the system and add new ones when they are ready.

The new PL-enabled process should drastically improve the performance and quality through the following drivers:

- Features are directly selected by the customer
- The composition and compatibility of features is automatically verified by PL tools
- The integration and bundling are automatically done by PL tools
- The verification and testing may be automated to a certain extent
- The distribution may also be automated

That way we intend to reduce the total time to market for the new custom bundles. Thus, the ultimate KPI is:

KPI	Baseline	Target
Time to market	72 hours (tbc)	24 hours or less

The PL process requires several steps including:

- Variability model extraction including the constraints
- Defining mapping of the Variability model and the PL assets
- Selecting the configuration needed for a new custom product
- Integration and bundling.
- Co-evolution of PL models with development / evolution of Modelio

Not all tools are readily available at the current stage. Some will require customization and development of specific adapters. Thus, the implementation of the validation experiments is staged in the following way.

- December 2018
  - Measuring and clarification of the baseline
  - Experiments with extraction tools and manual adaptation of PL models
  - Experiments with PL tools for generation of new products
- December 2019
  - Development of adaptors for automated extraction
  - Development of web tools for features selection by the customer
  - Development of on-line engines for the packaging and verification
  - Deployment of co-evolution solution
  - Measuring the performance of the fully automated suite

Thus, the KPIs targets may differ at the above-mentioned stages:

Milestone	Targeted KPI	Expectations
December 2018	72 hours	The process is largely manual. Learning curve will impact the performance. In the meantime, the solution will perform as good as the baseline
December 2019	Time to market for a new Modelio bundle. 24 hours or less	The full automated suite will help to drop the time to market by 300% or most probably more.

#### 8.4. Requirements

Req. ID	Object Text	Priority
1	The toolchain shall extract variability model and the PL constraints from the Modelio PL artefacts such as: - sets of Eclipse plugins for Modelio commercial and open source solutions - build and packaging scripts for Modelio commercial solutions and open source version	High
2	The toolchain shall provide capabilities to generate / package new variants of Modelio solutions corresponding to constraints given by users.	High
3	The toolchain shall automatically verify the validity of certain composition of Modelio solutions.	High
4	The toolchain shall provide a methodology for maintaining the PL generation pipeline.	High
5	The toolchain shall provide capabilities to propagate updates and fixes to the whole Modelio PL seamlessly.	High
6	The toolchain shall indicate the specific tests for new variants.	Medium
7	The toolchain shall help to document the Modelio PL portfolio.	High
8	The toolchain should support Window and Linux.	High



## 9. THALES

### 9.1. Use case Description

#### Generic need description

An initial software product PA was developed for a client A. Then it was evolved (in a configuration management branch) to a product PB for a client B. Almost in parallel, it was evolved (in another branch) to a product PC for a client C. Each branch introduced some new functionalities, corrected some bugs discovered by each client, and the product was refactored and broadly rearchitected at least in one branch. Due to industrial constraints (time, budget, separate responsibilities, etc.), the company never had time to merge these branches into a single product exhibiting variability, that is the goal of this use case, using ReVaMP<sup>2</sup> tools. We now want to have a single configurable product (using configuration files) or variable product line (using generation with pure::variants) for some variabilities, potentially addressing all the functionalities that we have developed for clients A, B, and C, and all bug corrections, but with single common software architecture.

#### Specific case description

The specific case “MazeGenerator” is stored in the shared space of the ReVaMP<sup>2</sup> project, in:

<https://svn.fzi.de/svn/ReVaMP2/WP2%20Industrial%20Use%20Cases/Artefacts/Thales%20artefacts/MazeGenerator/>

Please look at the document named “[mazegeneratorVersionControlHistory\\_README.docx](#)” for:

- a reminder of the goals of this example;
- an explanation of the representativity of this example;
- a description of the structure of the configuration management base;
- the expected results.

### 9.2. Goals

We remind here the reader of the Thales requirements, that:

- i) list the characteristics needed for a tool (or tool chain) to fulfil the goals of the Thales use case;
- ii) provide a suggested sequence of activities that should be supported by the tool (chain);
- iii) will be used as a reference for the KPIs in the subsequent section.

Here are these requirements:

- (A) Extraction and identification of the **structure** of the configuration management base (branches, versions) from the Git **repository**.
- (B) Identification of **evolutions** (known **refactoring** types, and other **additions**, **suppressions**, **modifications**) done between two successive versions in a same branch.
  - Weighted number of **refactoring** types correctly identified, among:
    - Move Method (weight 3.5)
    - Extract Method (weight 3) / Inline Method (weight 1)
    - Rename Method (weight 3)
    - Move Class (weight 2.5)
    - Rename Class (weight 1)
    - Change Method Signature (weight 1)
    - Move Attribute (weight 0.3)

- (TBC)
        - Zero **false positive** (evolutions badly identified as refactorings)
        - Small proportion of **false negative** (refactoring not identified)
    - (C) Correlation of these evolution identifications with the **comments** stored at commit time in the repository.
      - Ability to detect empty comments (not pertinent).
      - Possibility to parse comments to detect patterns, to identify:
        - Bug corrections
        - New functionality implementations
        - Reports from one branch to another
    - (D) Synthesize clearly these unitary evolutions of (B), at the highest level of abstraction possible, the set of **differences** between any 2 versions anywhere in the repository:
      - using the **comments** retrieved by (C) when possible,
        - grouping unitary evolutions identified by (B);
      - using standard **refactoring** names when possible;
      - expressing **additions** / **suppressions** / **modifications** in terms of the highest possible level of code artefact (class, method, ...).
    - (E) Represent concise architectural differences in UML diagrams.
    - (F) Proposition of a **variability model** and **possible architectures** for a hypothetical version that would merge all 2 or more input versions:
      - these architectures will be ones of existing versions, or possibly merges of those;
      - giving an estimation of the proportion of existing code that could be automatically merged to each new architecture possibility.
- KPIs:
- (G) Generate a merged code base version and variability model, given:
    - 2 or more input versions;
    - a target architecture;
    - and the Variability Implementation Technique decided for each Variation Point.
  - (H) Launch and compare results of automatic tests.

### 9.3. KPIs

KPI indicators are normalized to have a 0..1 range, and to have 0 meaning no support and 1 meaning full support.

Binary KPI indicators (Abilities) are valued 0 when not supported, 1 when supported, and possibly using intermediate value when an estimation of the support can be given.

KPI ID	Unique ReVaMP <sup>2</sup> selling point measured	Indicator
(A0)		<b>Ability</b> of the tool to <b>extract</b> the <b>structure</b> of the configuration management base (branches, versions) from a Git repository
(A1)		<b>Ability</b> of the tool to <b>extract</b> the <b>contents</b> of the configuration management base (files) from a Git repository
(A2)		<b>Ability</b> of the tool to <b>extract</b> the <b>metadata</b> of

		the configuration management base (commit comments, dates, authors) from a Git repository
(B0)		<b>Ability</b> of the tool to <b>identify evolutions</b> (see requirement (B)) between two successive versions in a same branch
(B1)		<b>Cumulated weight</b> of the types of <b>evolutions</b> for which the identification is implemented, following the reference given in requirement (B), divided by the total weight of expected types of evolutions to be identified
(B2)		<b>1 - Weighted mean rate of false positives</b> for each type of identified evolution, on the MazeGenerator case
(B3)		<b>1 - Weighted mean rate of false negatives</b> for each type of identified evolution, on the MazeGenerator case
(C0)		<b>Proportion</b> of commit comments correctly analysed
(D0)		<b>Proportion</b> of synthesis elements correctly presented by the tool
(E0)		<p><b>Correctness</b> of the generated <b>UML model</b>, measured as :</p> $\frac{n - d - m}{n}$ <p>where:</p> <ul style="list-style-type: none"> <li>• n is the total number of model elements in the expected model,</li> <li>• d is the number of model elements generated but different from the expected ones;</li> <li>• m is the number of missing model elements in the generated model.</li> </ul>
(F0)		<b>Correctness</b> of the generated <b>Feature Model</b> , measured using the same principles as (E0).
(G0)		<b>Correctness</b> of the generated <b>merged code base</b> , measured using the same principles as (E0), but applied to the number of source code lines
(H0)		<b>Ability to identify</b> corresponding automatic tests
(H1)		<b>Ability to launch</b> corresponding automatic tests

## 9.4. Requirements

Req. ID or Title	Object Text	Priority
Purpose	<p>This document describes needs of the THALES company for a tool or toolchain in the REVaMP<sup>2</sup> project, that is named <b>the Reevolution toolchain</b> in the rest of this document.</p> <p>It is to be understood as a subpart of the global REVaMP<sup>2</sup> toolchain, and its components and contributors are yet to be identified and defined.</p> <p><b>The Reevolution toolchain</b> is not meant to be the name of a specific tool, but rather a specification for any toolchain that would be compliant with it.</p> <p>It may be implemented in several ways, and the design of its various possible implementations is left to the appreciation of potential contributors.</p>	N/A
Scope	<p>The needed tool or toolchain will help THALES to reengineer several legacy source code bases that are main assets for several THALES product families.</p>	N/A
Definitions	<p><b>Reevolution</b> (standing for "<b>R</b>everse &amp; <b>R</b>e-evaluate <b>E</b>volutions"): the class of needs that is described in this document, and that is the generic goal of the THALES use case in REVaMP<sup>2</sup>.</p> <p><b>Configuration Management Repository</b>: a database storing all Versions and Branches of Code Bases for a given Product Family, managed by a Configuration management tool such as SVN (Subversion) or Git, typically.</p> <p><b>Code Base</b>: a set of source files forming the necessary asset to produce a Product.</p> <p><b>Product</b>: an artefact intended to be sold to a specific Client/Customer (or a set of Client/Customers), being a solution for a given Client/Customer problem. A Product is produced from a given Code Base.</p> <p><b>Product Family</b>: a set of Products, belonging to the same domain, and sharing some common characteristics (shared or similar requirements, functionalities, architecture, etc.), but intended to different Clients/Customers.</p> <p><b>Product Line</b>: a Product Family, where:</p> <ul style="list-style-type: none"> <li>- Variability is explicitly managed using a Variability Model (typically Feature Model);</li> <li>- Software Architecture is shared between all Products of the Product Line;</li> <li>- ideally, a single Code Base serves to produce all Products of the Product Line.</li> </ul> <p><b>Product Family Ancestor</b>: the original Product from which all other Products of a Product Family derived, by branching in the Configuration Management Repository.</p> <p><b>Version</b>: a frozen, well-identified (generally by a Version Number) snapshot of a Product Code Base.</p> <p><b>Evolution</b>: a difference between two successive Versions of a Product, interpreted by the Reevolution toolchain as belonging to one of these categories:</p> <ul style="list-style-type: none"> <li>- <b>Architectural Evolution</b> (Refactoring);</li> <li>- <b>Bug Correction</b>;</li> <li>- <b>Functional Evolution</b>.</li> </ul>	N/A
References	<p><b>Refactoring: Improving the Design of Existing Code.</b> Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts. 1999.</p>	N/A
A	<p>(A) The Reevolution toolchain shall extract and identify the structure of a Configuration Management Repository (Branches, Versions) from a</p>	Low

	Git repository.	
B	(B) The Reevolution toolchain shall categorize evolutions done between two successive versions in a same branch, between these 3 categories: - <b>Architectural Evolution</b> : evolution being an instance of a known Refactoring Type, as described in the book "Refactoring" by Martin Fowler & al., 1999; - <b>Functional Evolution</b> : the evolutions that cannot be identified by the test above.	High
C	(C) The Reevolution toolchain shall correlate these evolutions with the comments stored at commit time in the repository, to further categorize evolutions between the 2 categories of (B) and: - <b>Bug Correction</b> .	Medium
D	(D) The Reevolution toolchain shall synthesize these evolutions of (B), at the highest level of abstraction possible, the set of differences between any 2 versions anywhere in the repository: - using the comments retrieved by (C) when possible, grouping unitary evolutions identified by (B); - using standard refactoring names when possible; - expressing additions / suppressions / modifications in terms of the highest possible level of code artefact (class, method, ...).	High
E	(E) The Reevolution toolchain shall represent concise architectural differences in UML diagrams.	Medium
F	(F) The Reevolution toolchain shall propose a variability model and possible architectures for a hypothetical version that would merge all 2 or more input versions: - these architectures will be ones of existing versions, or possibly merges of those; - giving an estimation of the proportion of existing code that could be automatically merged to each new architecture possibility.	High
G	(G) The Reevolution toolchain shall generate a merged code base version and variability model, given: - 2 or more input versions; - a target architecture; - and the Variability Implementation Technique decided for each Variation Point.	High
H	(H) The Reevolution toolchain shall launch and compare results of automatic tests, between the original set of Products from the Product Family and the Product Line generated by (G) above.	Low

## 10. THE REUSE COMPANY

### 10.1. Use case Description

One challenge faced by the different organizations is to take advantage of all the knowledge developed among different projects, produced in many times with no management in a disorganized way, and to reuse the existing requirements in new variations of the same systems. With this Use Case, it is aimed to *discover and manage commonality and variability inside already existing and not organized requirements*, to identify and organize the variability in product lines and the variability models, and to create reusable requirements structures to be instantiated in future projects. The way to proceed will be to develop a System Knowledge Repository (SKR) and design a reusable Knowledge Base to represent commonalities among different domains which enhance assets reusability.

### 10.2. Goals

The main goals of the use Case as described in Deliverable 2.1, will represent the baseline for this first iteration within the project. The set of goals as they were described has been redefined due to the inputs discovered during the execution of the project, which make them more accurate and measurable.

Baseline set of goals:

1. Define a common understanding within variability management and product lines processes according to different standards for software and systems product line engineering
2. Define the quality priorities to ensure requirements reuse within product lines.
3. Develop a complete ontology, that enhance the reusability of systems knowledge by following a predefined structure of a Variability Management Ontology.

First iteration set of goals:

- 1.1 Defining a common understanding within variability management and product lines processes according to different standards for software and systems product line engineering
- 1.2 Enabling capabilities of extracting requirements from textual documents.
- 1.3 Providing a common framework that enables interoperability among the different sources of information, allowing the extraction of information from different sources.
- 2.1 Defining the quality priorities to ensure requirements reuse within product lines.
- 2.2 Defining the basic set of rules that will identify the level of quality for the information extracted from the SPL
- 3.1 Developing a complete ontology, that enhance the reusability of systems knowledge by following a predefined structure of a Variability Management Ontology.
- 3.2 Extracting requirements from textual documents based on textual patterns and the identification of properties and relationships also based on applying pattern matching over the requirements specifications.

### 10.3.KPIs

KPI ID	Unique ReVaMP <sup>2</sup> selling point measured	Indicator
EXT_01	Level of variability model <b>extraction</b> automation	Proportion of variability model elements that can be accurately and automatically extracted from legacy input assets on available use case benchmarks
VER_01	Level of variability model <b>verification</b> automation	Proportion of variability model defects that can be accurately and automatically identified, diagnosed and corrected from variability model, either manually constructed or automatically extracted from legacy assets on available use case benchmarks
UC_01	Genericity of the <b>automation</b> services	Proportion of different asset classes (e.g., requirements, system models, software models, hardware models, code, tests, documentation) for which target automation accuracy was attained within the tool chain
UC_02		Proportion of different asset representation language (e.g., Simulink, SysML, C) for which target automation accuracy was attained within the tool chain
UC_03		Proportion of different use cases for which target automation accuracy was attained within the tool chain
INT_01		Proportion of applications that can be interconnected within the tool chain

### 10.4.Requirements

Req. ID or Title	Object Text	Priority
System purpose	The system purpose is to discover and manage commonality and variability inside already existing and not organized requirements, to identify and organize the variability in product lines and the variability models, and to create reusable requirements structures to be instantiated in future projects.	N/A
1	The system shall discover commonality within not organized requirements.	High
2	The system shall discover variability within not organized requirements.	High
3	The system shall manage commonality within not organized requirements.	High
4	The system shall manage variability within not organized requirements.	High
5	The system shall identify variability from variability models in product lines.	High
6	The system shall identify commonality from variability models in	High

	product lines.	
System scope	<p>Knowledge-Based Requirements Engineering is being applied among different industries in the market, such as aerospace, automotive, energy or defence and space, which take advantage of the Knowledge-Based solutions to improve the ALM and PLM processes in the organization.</p> <p>RQS has wide application in all the described industries, and provides a way to cope with key trends in the market for system knowledge management and analysis, with reuse and traceability as a key opportunity to be applied into variability management and product lines framework.</p>	N/A
7	The system shall manage traceability from variability models in product lines.	High
8	The system shall manage knowledge from variability models in product lines.	High
Definitions, acronyms, and abbreviations	<p>ALM: Application Lifecycle Management.          BS: Breakdown Structure.          KM: Knowledge Manager.          OSLC: Open Services for Lifecycle Collaboration.          PDCA: Plan-Do-Check-Act.          PLM: Product Line Management.          RAT: Requirements Authoring Tool.          RQA: Requirements Quality Analyser.          RQS: Requirements Quality Suite, developed by TRC.          SCM: System Conceptual Model.          SKR: System Knowledge Repository.          TRC: The REUSE Company.</p>	N/A
9	The system shall support different user rights.	Medium
10	The system shall support PDCA process.	Medium
11	The system shall manage knowledge of requirements.	High
12	The system shall manage quality of requirements.	High
13	The system shall be compatible with correctness quality assessment.	High
14	The system shall be compatible with completeness quality assessment.	High
15	The system shall be compatible with consistency quality assessment.	High
Major system capabilities		
16	The system shall extract variability from a set of requirements.	High
17	The system shall identify commonalities from a set of requirements.	High
18	The system shall create reusable requirements structures.	High
19	The user shall gather the system information into the SKR.	Medium
20	The user shall restrict requirements reusability.	High
21	The user shall perform incompleteness quality checking of the requirements documents.	High
22	The user shall perform inconsistency quality checking of the requirements documents.	High
23	The user shall perform correctness quality checking of the	Medium



	requirements.	
24	The quality checking shall determine the reusability of the requirements.	Medium
25	The representative breakdown structures shall be reused.	High
26	The user shall identify representative requirements structure.	Medium
27	The user shall identify the main concepts within product lines development.	
28	The user shall identify the main architectures within product lines development.	Medium
29	The system shall support controlled vocabulary management.	Medium
30	The user shall define the common vocabulary of the system knowledge.	Medium
31	The common vocabulary of the systems knowledge shall follow the predefined structure of the Variability Management Ontology.	Medium
32	The user shall organize the common vocabulary based on the semantic meaning.	Medium
33	The system shall support SCM breakdown structures in different views of the xBS.	Medium
34	The variability model shall be one of the xBS views.	Low
35	The user shall define the breakdown structures from the product line without limitation.	Low
36	The system shall support requirements patterns management.	High
37	The system shall support requirements grammars management.	High
38	The system shall identify requirements typology.	High
39	The patterns shall detect commonalities from requirements specifications.	High
40	The grammars shall detect commonalities from requirements specifications.	High
41	The patterns shall detect variabilities from requirements specifications.	High
42	The grammars shall detect variabilities from requirements specifications.	High
43	The user shall design the set of patterns for variability discovering in requirement statements.	Medium
44	The user shall design the set of patterns for variability discovering in requirement statements.	Medium
45	The knowledge framework for variability management shall define Variability Management Ontologies.	High
46	The knowledge framework for variability management shall manage the assets from the Variability Management Ontologies.	High
47	The set of patterns shall be configurable.	High
48	The knowledge framework shall support patterns configuration.	High
49	The knowledge framework for variability management shall generate Product Requirements Specifications.	High
50	The variability model shall be the source of information for the Product Requirements Specification.	Medium
51	The feature model shall be the source of information for the Product Requirements Specification.	Medium

52	The knowledge framework for variability management shall generate the Variability Management Ontology for the product.	High
53	The system shall be compatible with different data sources.	High
54	The System shall be compatible with OSLC interoperability.	High
55	The System shall interoperate with product lines data sources.	High

## 11. Overall project KPIs

Following KPIs are given as overall project KPIs, i.e. high-level non-technical KPIs. Each partner can have one or several roles among:

- Use case provider;
- Academic partner;
- Tool provider.

When asked by partners, their name have been obfuscated by randomly assigning them a letter (A, B, C, ...).

### 11.1. Karlstad University

#### 11.1.1. Academic Partner KPI

**Task:** PhD dissertation

**Baseline:** New PhD student hired

**Month 24:** Progress at 40% as documented in the individual study plan

**Month 36:** Licentiate completed

**Expectation:**

**Task:** Publish research articles

**Baseline:** 0 (#Articles)

**Month 24:** 1

**Month 36:** 5

**Expectation:**

### 11.2. THE REUSE COMPANY (Knowledge Centric Solutions)

#### 11.2.1. Use Case Provider KPI

**Task:** Process Automation. Number of manual steps to extract commonality of req. docs. Manually vs (/) Number of automatic steps.

**Baseline:** 15/4

**Month 24:** 15/8

**Month 36:**

**Expectation:** 15/13

#### 11.2.2. Tool Provider KPI

**Task:** Process Automation. Number of implemented features to extract commonality of req. docs. To satisfy the Number of automatic steps.

**Baseline:** 2

**Month 24:** 10

**Month 36:**

**Expectation:** 13

### 11.3.KTH (Royal Institute of Technology)

#### 11.3.1. Academic Partner KPI

**Task:** Continue research related to activities in REVAMP

**Baseline:** Prepare an application for a project on automated software verification

**Month 24:** Funding for the project AVeRT obtained from Swedish funding agency Vinnova

**Month 36:** Deliver a formal framework for requirement breakdown

**Expectation:** Deliver a toolset for specification and automated verification of software systems

**Task:** Academic theses related to activities in REVAMP

**Baseline:** 0

**Month 24:** 4 MSc theses, new PhD hired

**Month 36:**

**Expectation:** 6 Msc Theses

**Task:** Publications

**Baseline:** 0

**Month 24:** 5

**Month 36:**

**Expectation:** >= 12

**Task:** Created a course on program verification which is strongly inspired by the work in REVAMP

**Baseline:** Old course existed

**Month 24:** Completely revamped the old course and given the first installment

**Month 36:** Give the second installment

**Expectation:** Continue giving the course on a yearly basis

### 11.4.Model Engineering Solutions GmbH

#### 11.4.1. Tool Provider KPI

**Task:** Detection and Visualization of Simulink Model Partitions. Number of partition types automatically detectable and visualizable by tool.

**Baseline:** 2

**Month 24:** 6

**Month 36:**

**Expectation:** 10

### 11.5.Scania

#### 11.5.1. Use Case Provider KPI

**Task:** Provide tool support for writing variability-aware, ISO 26262 compliant specifications for subsystems and components in the Scania PL.

**Baseline:** No tool support.

**Month 24:** Prototype of the tool Asset Editor.

**Month 36:** Principles from Asset Editor used to configure a well-known requirements management tool procured by Scania.

**Expectation:** Use the tool to automatically produce ISO 26262 compliance argumentation.

**Task:** Formally verify properties of C code.

**Baseline:** Formal verification of production software not performed.

**Month 24:** 5 properties verified against 2 application level C language modules.

**Month 36:**

**Expectation:** 12 properties verified against 3 application level C language modules.

## 11.6. University of Hildesheim

### 11.6.1. Academic Partner KPI

**Task:** Paper Publication

**Baseline:** 0

**Month 24:** 9

**Month 36:**  $\geq 12$  (3 papers already accepted in/for 2019)

**Expectation:**  $\geq 14$

**Task:** Completed Theses (PhD, MSc, BSc)

**Baseline:** 2 PhD theses with a tight connection to REVAMP sub-topics already started before the project; corresponding students are working in REVAMP now

**Month 24:** 2 (1 MSc, 1 BSc)

**Month 36:**  $\geq 4$  (+2 submitted PhD theses)

**Expectation:**  $\geq 4$  (more project-related theses in progress/in discussion)

**Task:** Software Product Line Development Training

**Baseline:** 0

**Month 24:** 2 tutorials at SPLC 2018

**Month 36:**  $\geq 2$  (tutorials at SPLC 2019, if accepted, or other trainings)

**Expectation:**  $\geq 2$

## 11.7. Partner A

### 11.7.1. Use Case Provider KPI

**Task:** Feature Model Creation for a Legacy Component

**Baseline:** manual effort 160h for modeling 250 features and 250 relations

**Month 24:** manual effort of 80h for reviewing and extending model content proposed by the REVAMP<sup>2</sup> toolchain (50% effort reduction)

**Month 36:** as in month 24

**Expectation:** 50% effort reduction (h)

**Task:** Verification of Source Code against a Variability Model

**Baseline:** manual effort of 20h for reviewing a component (20 KLOC of source code)

**Month 24:** manual effort of 8h for reviewing a report provided by the REVAMP<sup>2</sup> toolchain (60% effort reduction)

**Month 36:** manual effort of 5h (75% effort reduction)

**Expectation:** 75% effort reduction

## 11.8.Partner B

### 11.8.1. Use Case Provider KPI

**Task:** Extraction of features from architectures of mechatronic systems: hybrid drivetrain classification

**Baseline:** 250 h for manually classifying 100,000 architectures, by an expert user

**Month 24:** 20 h for setting up classification problem and running automatic classification

**Month 36:** same as M24: performance increase of (automatic) classification but negligible time reduction compared to (manual) problem setup

**Expectation:** >90% speedup of classification for given use case

## 11.9.Partner C

### 11.9.1. Use Case Provider KPI

**Task:** Feature Identification and Documentation

**Baseline:** manual effort 1000h for 86 features

**Month 24:** using REVaMP<sup>2</sup> toolchain effort is 310 h (reduction in effort of 69%)

**Month 36:**

**Expectation:** 40% reduction in effort (h)

**Task:** Migration of legacy code to SPL

**Baseline:** 36 months for 80% of the source code (Danfoss experience)

**Month 24:** 30.34 months using REVaMP<sup>2</sup> (speedup of 15,72 %)

**Month 36:**

**Expectation:** 30% (speedup in moving to a SPL by using REVaMP<sup>2</sup> in months)

## 11.10.Partner D

### 11.10.1. Tool Provider KPI

**Task:** Business domains of application. Number of "interested in SPL" market domains.

**Baseline:** 2

**Month 24:** 2

**Month 36:**

**Expectation:** 5

## 11.11.Partner E

### 11.11.1. Use Case Provider KPI

**Task:** Feature model creation and variant generation automation for Modelio

**Baseline:** Manual generation of new variants of Modelio installation (10h)

**Month 24:** Creation of new variants using the feature model (3h)

**Month 36:** Complete automation of the variant generation process (1h)

**Expectation:** Reduction of time and effort needed to produce new versions of Modelio

## 11.12. Partner F

### 11.12.1. Use Case Provider KPI

**Task:** Within partner F a large number of assets are used to build simulators and aircrafts for simulation. The success of PLE can be measured in the number of times an asset is used within one or more product lines.

We assign the value of an asset (asset value) used once:  $1 (=2^{(1-1)})$ , if a PLE approach is used and the result of the derivation of the asset is used  $n$  times then the asset value is set to  $2^{(n-1)}$ .

The PLE approach is deemed successful if the summarised asset value is increasing and the mean value (asset value)/(number of assets) is increasing

**Baseline:** To be determined

**Month 24:** To be determined

**Month 36:** To be determined

**Expectation:** Summarised asset value and mean asset value is increasing.

Due to the comparatively slow pace within aerospace development and the large number of assets the improvements are expected to be small to moderate - but still significant

## 11.13. Partner G

### 11.13.1. Use Case Provider KPI

**Task:** T3.3

**Baseline:**

**Month 24:** Questionnaire with 21 questions and overall score

**Month 36:**

**Expectation:** 30% increased efficiency in communication between stakeholders and developers.

## References

- [1] L. Passos, et. Al. 2015. Feature scattering in the large: a longitudinal study of Linux kernel device drivers. In Proceedings of the 14th International Conference on Modularity (MODULARITY 2015). ACM, New York, NY, USA, 81-92. DOI=<http://dx.doi.org/10.1145/2724525.2724575>.