Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool

Muhammad Atif Javed and Barbara Gallina
School of Innovation, Design and Engineering,
Mälardalen University, Västerås, Sweden
{muhammad.atif.javed,barbara.gallina}@mdh.se

ABSTRACT

The integration between process engineering and variability management is required for tailoring of safety-oriented processes with variabilities to individual projects in a similar manner to the product lines. Previous studies have not adequately established the Safety-oriented Process Lines (SoPLs). This paper focuses on the seamless integration between Eclipse Process Framework (EPF) Composer and Base Variability Resolution (BVR) Tool. The former supports the major parts of the OMG's Software & Systems Process Engineering Metamodel (SPEM) Version 2.0, while the latter is a simplification and enhancement of the OMG's revised submission of Common Variability Language (CVL). The proposed integration is implemented as Eclipse plugin. It provides support for importing backend folders and files within the method library of EPF Composer, resolving problems with the files for variability management with the BVR Tool, and exporting back the resolved process models to the EPF Composer. The applicability of the implemented plugin is demonstrated by engineering an ECSS-E-ST-40C compliant SoPL for the space projects and applications.

CCS CONCEPTS

• **Software and its engineering → Software development process management**; **Software product lines**;

KEYWORDS

Seamless Integration, Process Engineering, EPF Composer, Variability Management, BVR Tool and Process Line Implementation.

ACM Reference Format:
Muhammad Atif Javed and Barbara Gallina. 2018. Safety-oriented Process Line Engineering via Seamless Integration between EPF Composer and BVR Tool . In *22nd International Systems and Software Product Line Conference - Volume B (SPLC '18), September 10–14, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3236405.3236406

1 INTRODUCTION

The safety-oriented processes tend to be reused, modified and extended to individual projects in a similar manner to the product lines [12]. However, to be able to establish the Safety-oriented Process Lines (SoPLs), the seamless integration between process engineering and variability management is required. SoPL engineering deals with the identification and systematization of commonalities and variabilities to concurrently engineer a set of processes; the achievement of single processes is based on the selection and composition of commonalities and variabilities [6].

The research in Oliveira Jr. et al. [3] incorporated the variability mechanism in OMG's Software & Systems Process Engineering Metamodel (SPEM). This proposal, however, needs to be implemented. Rouillé [13] suggested OMG's Common Variability Language (CVL), but the integration with a process engineering tool was not considered. Aleixo et al. [1] specified the feature variabilities annotations in the Extensible Metadata Interchange (XMI) files produced by Eclipse Process Framework (EPF) Composer[1]. Although the derivation of customized specifications are supported with a product line tool, the manual annotations bring serious difficulties. The implementation by Simmonds et al. [14] focused on models and transformations in the megamodel for establishing process lines. The megamodel includes models, their corresponding metamodels, and text-to-model, model-to-model, model-to-text transformations, as well as higher order transformations for which the source code is written in Java language [2]. The megamodel-based solution requires high effort for defining models, tailoring transformations and their evolution for the software processes modelled in EPF Composer. Furthermore, the graphical support for variability modelling and management is not provided.

EPF Composer and Base Variability Resolution (BVR) Tool[2] are well-known process engineering and variability management solutions, respectively. They are implemented as Eclipse plugins, which are licensed under the Eclipse Public License (EPL) Version 1.0. Besides that, the EPF Composer supports the major parts of the OMG's SPEM 2.0, while the BVR is built on the OMG's revised submission of CVL. In order to establish the SoPLs, the seamless integration between EPF Composer and BVR Tool has been achieved. More specifically, the implemented plugin provides support for importing backend folders and XMI files within the method library of EPF Composer, resolving problems with the opening and mapping of XMI files for variability management with the BVR Tool, and exporting back the configured process models to the EPF Composer. The applicability of the implemented plugin is demonstrated for the space projects and applications [4].

The rest of this paper is organized as follows: Section 2 provides background information on EPF Composer and BVR Tool. Section 3 discusses the seamless integration between EPF Composer and BVR Tool for establishing SoPLs. Section 4 concludes the paper and

SPLC '18, September 10–14, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5945-0/18/09.
https://doi.org/10.1145/3236405.3236406

[1] See https://www.eclipse.org/epf/
[2] See https://github.com/SINTEF-9012/bvr

sketches future research directions. The appendix demonstrates usability of the implemented solution by engineering an ECSS-E-ST-40C compliant SoPL.

2 BACKGROUND

2.1 EPF Composer

EPF Composer provides support for authoring, tailoring and deploying (software) systems development processes. This means that process structures containing all necessary process elements (e.g., activities, tasks, roles, workproducts, etc.) can be specified. EPF Composer is based on the Unified Method Architecture (UMA) metamodel, which is an evolution of the OMG's SPEM 1.1 [10]. The major parts of UMA are incorporated in SPEM 2.0 [11]. UMA defines library as a root container for method plugins and configurations.

Method plugins are divided into two primary categories: method content and processes. The method content describes the required steps and skills to achieve the specific development goals, which comprise of content packages, standard categories and custom categories. The tasks, roles, work products and guidance are specified in the method content packages, whereas the disciplines, domains, work product kinds, role sets and tools are regarded as standard categories. EPF Composer stores all method library content in a repository of XMI files, which is an OMG's specification for storing and interchanging metadata in Extensible Markup Language (XML) format. The method content elements are organized into semi-ordered sequences, which provide the means to create a process lifecycle. The capability patterns are building blocks that hold process knowledge for a key area of interest, but the complete lifecycles are modelled as delivery processes.

Method configurations point out the working subset within the library. Therefore, the corresponding method content and process elements have to be selected. EPF Composer supports the generation of website based on the method configuration that can be deployed to web servers for distributed collaboration between multiple teams. Two web application formats are supported: HTML and Java EE. Note that the standard and custom categories appear in the published website as navigation views.

2.2 BVR Language and BVR Tool

BVR [15] is a language built on top of CVL [7] for enabling variability modeling in the context of safety-critical systems engineering. The language is implemented as a series of Eclipse plugins, which supports feature modelling, resolution, realization and derivation of specific family members, as well as their testing and analysis. Because the language defines variability orthogonally for any Meta-Object Facility (MOF)-compliant model (representing the Base model), communication with other tools is needed to map the elements of a target configuration and variability abstractions in BVR. The generation of target configurations is performed with three editors: VSpec, Resolution, and Realization.

The VSpec editor permits variability engineers to create VSpec models, which are an evolution of the Feature-Oriented Domain Analysis (FODA) [9], usually called Feature Models. More specifically, VSpec extends FODA by including additional concepts such as variables, references and multiplicities. The mandatory features are connected to the parent feature via solid lines, whereas the dashed lines represent optionality. The constraint-based resolution is also incorporated. This means that logical operators such as implication, alternative, negation might be used.

The Resolution editor permits variability engineers to perform the resolution and obtain resolved models. To perform the resolution, engineers have to specify the desired inclusion/exclusion choices for the specific configuration. The validation process is executed to conform whether the resolution corresponds to the VSpec model. It is possible to define multiple resolutions for the processes with variabilities. Software Product Line Covering Array (SPLCA) tool is also integrated in the BVR bundle [8]. It supports the generation of covering arrays from large feature models.

Realization is based on the placements and replacements within the fragment substitutions. Fragment substitution removes elements within the placement and substitutes them with the elements in replacement. Therefore, the links between VSpec features and fragment substitutions need to be established. For making specification intuitive and visual, the placements and replacements are highlighted in red and blue colours, respectively. The substitutions are executed based on variability definitions in the abstract and realization layers in order to derive new processes. The realization engine identifies and resolves the conflicts between fragments, and reports unresolved failures. In summary, the BVR Tool provides advanced support for managing families for the choices of the base models for example process lines and product lines.

3 SEAMLESS INTEGRATION BETWEEN EPF COMPOSER AND BVR TOOL

EPF Composer has attracted considerable attention from researchers and practitioners worldwide. However, to be able to establish the SoPLs, the integration with variability modelling and management is required [1][3][14]. This might be done in two possible ways: either the support for variability modelling and management is incorporated and implemented, or otherwise the integration with variability management solution needs to be achieved. In the context of the AMASS[3] project, it is decided to achieve the seamless integration between EPF Composer and BVR Tool that provides advanced support for establishing SoPLs.

As mentioned in Section 2.2, BVR defines variability orthogonally for any MOF-compliant model, but the integration with EPF Composer brings additional challenges. In this paper, the seamless integration problems between EPF Composer and BVR Tool are investigated and resolved. At first, we perform the migration of EPF Composer from Eclipse Galileo to Eclipse Neon (Section 3.1). After that, the support is implemented for importing the method library of EPF Composer and resolving the problems with XMI files for variability management with the BVR Tool (Section 3.2). For a software process modelled in EPF Composer, the feature diagram associated to the SoPL is modelled in the VSpec editor, the process configurations are performed in the resolution editor, and the placement and replacement fragments are defined in the realization editor. Finally, the configured process models are exported back to the EPF Composer (Section 3.3). The seamless integration is implemented as Eclipse plugin. An overview of the seamless integration between EPF Composer and BVR Tool is shown in Figure 1.

[3] See https://www.amass-ecsel.eu/

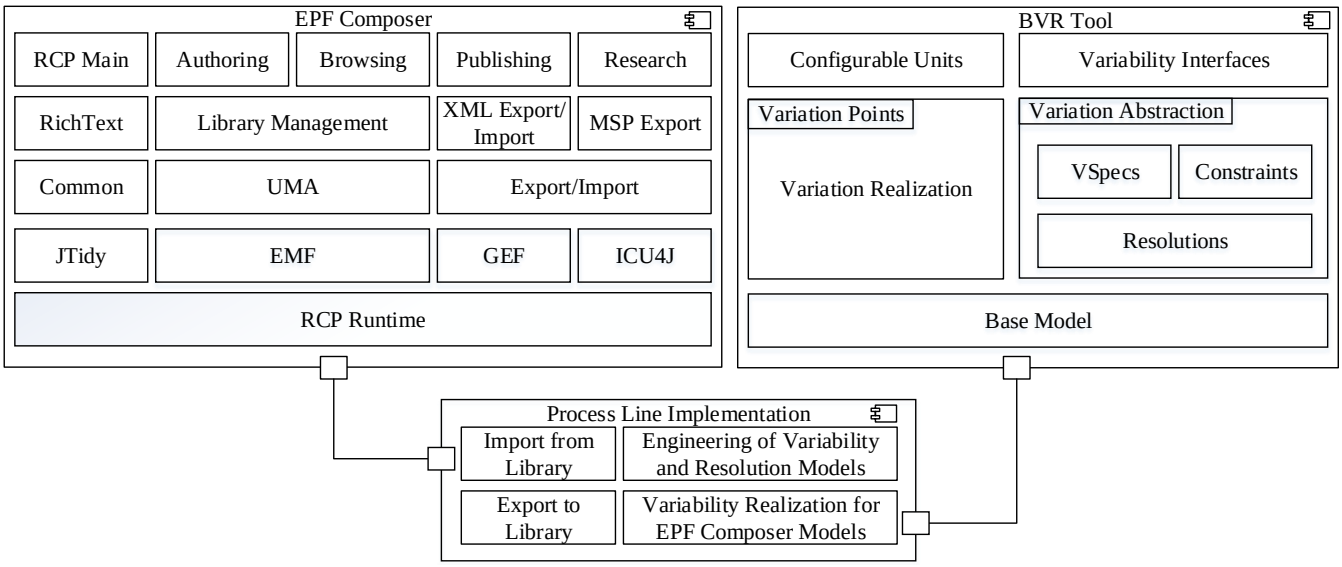

Figure 1: Overview of the seamless integration

3.1 Migration of EPF Composer from Eclipse Galileo to Eclipse Neon

EPF Composer is the only available implementation of OMG's SPEM 2.0, but the migration of EPF Composer to newer versions of technologies was never performed. Accordingly, we evolved the EPF Composer[4] from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 after 11 years. This is done for performing the integration with BVR Tool in the AMASS platform. This contribution is acknowledged by the IBM[5], and therefore committer status is assigned for the project. The migration is performed in four steps:

- **Step 1**: Required softwares are installed from the software repository for the Eclipse Neon release and then deprecations in the source code are analysed and fixed.
- **Step 2**: Problems with the persistence of method elements (i.e., method configurations, method plugins, method content descriptions and processes) are resolved. In particular, they are stored in their own folders and XMI files.
- **Step 3**: Selection of currently used method configuration is performed through a combo box for which the appearance and height problems are resolved, the blank views are removed from both authoring and browsing perspectives, and the rich text editor problems are resolved for enabling users to format and style text.
- **Step 4**: Incompatible bundles in feature plugins are deleted, replacement bundles are added and other missing dependencies for the bundles are resolved for the generation of application. As per recommendation, the EPF Composer might be launched as a standalone application, but also in the Eclipse Integrated Development Environment (IDE).

3.2 Importing the EPF Composer Library and Resolving the Problems in XMI Files

EPF Composer is based on the UMA metamodel. It persists the method library contents in their own folders and XMI files, in particular, method plugins, processes, content descriptions and configurations. In case of a new plugin, a plugin.xmi file is created in the new plugin directory and the reference of plugin is added to the library.xmi file. When a new capability pattern or delivery process is created, the model.xmi and content.xmi files are created in a new directory, and the reference of new process is added to the plugin.xmi file. Similarly, moving a content element to another plugin changes plugin.xmi in both plugins. The configuration file is used to specify the working set. Therefore, it records the references to included content packages and processes.

The XMI files produced by EPF Composer are neither directly opened nor mapped at the realization editor. Therefore, the problems in XMI files have to be resolved for variability management with the BVR Tool. It is decided to copy the method library before resolving the problems in XMI files for two reasons. Firstly, the library might be keep running in EPF Composer. Secondly, reverting back is just required for configured processes. The packages and resource factories are registered for the UMA metamodel. Otherwise, the package and class not found exceptions would be raised. We have identified that the hypertext references (hrefs) in XMI files are based on the globally unique identifiers (GUIDs) for example uma://_ErexoKA4EeaPp8nsuu2eew. This is the case with multiple UMA metamodel elements, such as tasks, roles, work products, tool mentors and method packages. As a result, the malformed URL exceptions are produced. The platform specific paths or otherwise Uniform Resource Identifiers (URIs) should be used instead. The support for the identification and resolution of problems with hrefs has been implemented.

[4]See https://bugs.eclipse.org/bugs/show_bug.cgi?id=516608
[5]See https://www.ibm.com/

As specified in Section 2.2, to create the placement and replacement fragments, the model elements are dragged and dropped to the realization editor, and then create placement or create replacement option is selected from the context menu. This, however, produces the illegal operation exception for UMA compliant models. The analysis reveals that multiple metamodel elements have associated description implementations, such as deliverables and break down elements. Their naming structure `"parent name","parent GUID"` is not allowed. Accordingly, we performed temporary adaptations for supporting placements and replacements in the realization editor. The problems are resolved for all XMI files; the method library, configurations, plugins, processes and content descriptions might be considered for variability management with the BVR Tool. The visual support for highlighting objects placements in red while replacements in blue colours, as well as retrieving selections are supported for UMA compliant models.

3.3 Exporting the Generated Process Models to EPF Composer Application

The generated process models are automatically exported back to the EPF Composer. At the opening of generated process models, the dialogue window pops up to inform that "the files have been changed on the file system. Do you want to load the changes?" Pressing the "Yes" button loads the derived process models in EPF Composer. The support for saving the copy of previous models is also incorporated. It might be noted that the changes for resolving problems in XMI files and supporting the communication with realization editor had been reverted back in exported models.

4 CONCLUSIONS AND FUTURE WORK

In this paper, the seamless integration between EPF Composer and BVR Tool has been achieved. EPF Composer and BVR Tool are open source and well-known process engineering and variability management solutions based on OMG's SPEM and CVL standards, respectively. At first, we evolved the EPF Composer from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 after 11 years. After that, the seamless integration is implemented as Eclipse plugin, which provides advanced support for establishing SoPLs. More specifically, the backend folders and files from the EPF Composer method library are imported, the problems with the opening and mapping of XMI files are resolved, and the processed models are exported back to the EPF Composer. The implemented plugin is available under the PolarSys OpenCert[6] project. The usability of the implemented plugin is demonstrated in the appendix; ECSS-E-ST-40C compliant SoPL is established for the space projects and applications. It has also been videotaped[7]. As future work, we plan to extend the integration to other tools (CHESS[8] toolset for system variability and OpenCert for assurance case variability) in order to implement ideas related to Anti-Sisyphus [5], the method to support variability management along three dimensions: process, product and assurance case.

[6]See https://www.polarsys.org/proposals/opencert
[7]See https://www.amass-ecsel.eu/content/training
[8]See https://www.polarsys.org/projects/polarsys.chess

ACKNOWLEDGMENTS

This work is supported by EU and VINNOVA via the ECSEL Joint Undertaking under grant agreement No. 692474, AMASS project. The authors would like to thank Ø. Haugen, A. Vasilevskiy, I. Ayala and A. Carlsson for their relevant suggestions.

REFERENCES

[1] Fellipe Araújo Aleixo, Marília Aranha Freire, Wanderson Câmara dos Santos, and Uirá Kulesza. 2010. Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach. In *Enterprise Information Systems - 12th International Conference, ICEIS 2010, Funchal, Madeira, Portugal, June 8-12, 2010, Revised Selected Papers*. 372–387. https://doi.org/10.1007/978-3-642-19802-1_26

[2] María Cecilia Bastarrica, Jocelyn Simmonds, and Luis Silvestre. 2014. Using megamodeling to improve industrial adoption of complex MDE solutions. In *6th International Workshop on Modeling in Software Engineering, MiSE 2014, Hyderabad, India, June 2-3, 2014*. 31–36. https://doi.org/10.1145/2593770.2593773

[3] Edson Alves de Oliveira Junior, Maicon G. Pazin, Itana Maria de Souza Gimenes, Uirá Kulesza, and Fellipe Araújo Aleixo. 2013. SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines. In *Product-Focused Software Process Improvement - 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013. Proceedings*. 169–183. https://doi.org/10.1007/978-3-642-39259-7_15

[4] Deliverable D1.1 (AMASS). 2017. Case studies description and business impact–final version. http://www.amassecsel.eu/sites/amass.drupal.pulsartecnalia.com/files/documents/D1.1_Case-studies-description-and-business-impact_AMASS_Final.pdf. (2017). (Last accessed: July 3, 2018).

[5] Barbara Gallina. 2015. Towards Enabling Reuse in the Context of Safety-Critical Product Lines. In *5th IEEE/ACM International Workshop on Product Line Approaches in Software Engineering, PLEASE 2015, Florence, Italy, May 19, 2015*. 15–18. https://doi.org/10.1109/PLEASE.2015.12

[6] Barbara Gallina, Shaghayegh Kashiyarandi, Helmut Martin, and Robert Bramberger. 2014. Modeling a Safety- and Automotive-Oriented Process Line to Enable Reuse and Flexible Process Derivation. In *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC Workshops 2014, Vasteras, Sweden, July 21-25, 2014*. 504–509. https://doi.org/10.1109/COMPSACW.2014.84

[7] Øystein Haugen, Birger Møller-Pedersen, Jon Oldevik, Gøran K. Olsen, and Andreas Svendsen. 2008. Adding Standardized Variability to Domain Specific Languages. In *Proceedings of the 12th International Conference on Software Product Lines (SPLC '08), Limerick, Ireland, September 8-12, 2008*. https://doi.org/10.1109/SPLC.2008.25

[8] Martin Fagereng Johansen, Øystein Haugen, Franck Fleurey, Anne Grete Eldegard, and Torbjørn Syversen. 2012. Generating Better Partial Covering Arrays by Modeling Weights on Sub-product Lines. In *Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*. 269–284. https://doi.org/10.1007/978-3-642-33666-9_18

[9] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report. Carnegie-Mellon University Software Engineering Institute.

[10] Object Management Group (OMG). 2004. Software Process Engineering Metamodel Specification (SPEM), Version 1.1. ftp://ftp.omg.org/pub/spem-rtf/SPEM-CD-20040308.pdf. (2004). (Last accessed: July 3, 2018).

[11] Object Management Group (OMG). 2008. Software & Systems Process Engineering Metamodel Specification (SPEM), Version 2.0. http://www.omg.org/spec/SPEM/2.0/. (2008). (Last accessed: July 3, 2018).

[12] H. Dieter Rombach. 2005. Integrated Software Process and Product Lines. In *Unifying the Software Process Spectrum, International Software Process Workshop, SPW 2005, Beijing, China, May 25-27, 2005, Revised Selected Papers*. 83–90. https://doi.org/10.1007/11608035_9

[13] Emmanuelle Rouillé, Benoît Combemale, Olivier Barais, David Touzet, and Jean-Marc Jézéquel. 2012. Leveraging CVL to Manage Variability in Software Process Lines. In *19th Asia-Pacific Software Engineering Conference, APSEC 2012, Hong Kong, China, December 4-7, 2012*. 148–157. https://doi.org/10.1109/APSEC.2012.82

[14] Jocelyn Simmonds, Daniel Perovich, María Cecilia Bastarrica, and Luis Silvestre. 2015. A megamodel for Software Process Line modeling and evolution. In *18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS 2015, Ottawa, ON, Canada, September 30 - October 2, 2015*. 406–415. https://doi.org/10.1109/MODELS.2015.7338272

[15] Anatoly Vasilevskiy, Øystein Haugen, Franck Chauvel, Martin Fagereng Johansen, and Daisuke Shimbara. July 20-24, 2015, Nashville, TN, USA. The BVR tool bundle to support product line engineering. In *Proceedings of the 19th International Conference on Software Product Line (SPLC '15)*. https://doi.org/10.1145/2791060.2791094

APPENDIX

A TAILORING OF ECSS-E-ST-40C FOR SPACE SOFTWARE ENGINEERING

ECSS-E-ST-40C targets software development. It is one of the series of ECSS standards intended to be applied together for the management, engineering and product assurance in space projects and applications. Similar to other standards, it represents the effect *"standards for making standards"*, the idea being that this permits suppliers to use their own standards, provided that they comply with the requirements of ECSS-E-40 or some tailoring of it defined by the customer. The tailoring rules are provided in a specific annex, *Annex R (normative)*. Specifically, the tailoring is conducted based on the software criticality, which ranges from A to D.

Due to space reasons and similar technical details for engineering an overall ECSS-E-ST-40C compliant process line, we have limited the discussion to Section 5. More specifically, the software design and implementation engineering process is constituted of design of software items, coding and testing, and integration activities, each of which contains various tasks, which in turn contains various steps. The *integration*, for instance, is composed of two tasks: *Software integration test plan* and *Software integration test report*. According to Annex R, the former is applicable (Y) for levels A-B, and is also applicable (Y) for level C except SUITP K.9 and K10; but it is not applicable for level D. The latter is applicable (Y) for levels A-C; but inapplicable (N) for level D. In this context, the ability to manage process variability is becoming strategic: key to reduction of unnecessary and repetitive process management activities and thus key to potential time saving and cost reduction.

B IMPORTING FROM LIBRARY AND FIXING THE PROBLEMS

We have implemented a dialogue wizard to support the mapping of target configurations at the realization editor, as shown in Figure 2, the recent/default path choice is automatically filled in the path text box otherwise the path containing a specific method library might be browsed. The dialogue wizard performs two tasks: (i) imports the contents of the method library in the target directory; and (ii) resolves problems with the XMI files. The error free models are made available in the project folder. All the model files can be opened, for example, method configurations, method plugins, method content descriptions and processes.

C PROCESS VARIABILITY MANAGEMENT WITH THE BVR TOOL

The generation of target configurations for a software process modelled in EPF Composer is performed with VSpec, Resolution, and Realization editors, as illustrated in Figure 3. The tailoring rules provided in Annex R of ECSS-E-ST-40C are modeled within the VSpec editor. The resulting VSpec model shows the tree structure representing logical constraints to be considered during the resolution. As mentioned in the background, the solid line indicates that the particular feature applies to all criticality levels, whereas the dashed line represents a variation point. The whole tree cannot be visualized due to space limitations; therefore the

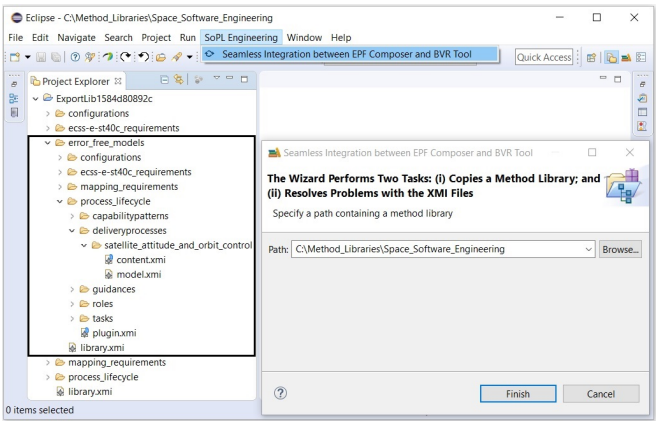

Figure 2: The achievement of error free models

minimize option (+) is used for hiding the features. The tasks associated with `software integration test plan development`, and `software units and software component integration and testing` are marked as optional. The multiplicity xor(1..1) is assigned to the criticality; therefore exactly one out of A, B, C and D must be selected for the software product. As the choices are associated with multiple criticality levels, the constraints have been applied; valid tailoring is guaranteed if the constraints are properly specified. For instance, the constraint `(A or B) implies (not Software_integration_test_plan)` indicates that the Software_integration_test_plan must be excluded for processes with criticality A or B. Likewise, the constraint *C implies (Software_integration_test_plan and ((not SUITP_K9) and (not SUITP_K10)))* enforces inclusion of Software_integration_test_plan, but also exclusion of SUITP K.9 and K10 for criticality C.

The resolution models are automatically generated from the VSpec model, but the choices needed to be included or excluded for individual processes. In this regard, multiple resolutions might be defined for the process with variability. The implemented editor supports error checking and validation of resolutions. The derivation process involves the substitutions in which elements of a placement fragment are removed and elements of a replacement are injected. To create the placements and replacements, the elements from the models are dragged and dropped on the realization editor. It is therefore possible to define the placements and replacements between multiple models. The placements are visualized in red, while the replacements in blue colours. The variation points have the associated bindings, i.e., the mapping of boundary elements between placements and replacements. As the requirements are either included or exuded based on the criticality of the system, the bindings specifies the replacements with NULL elements. Otherwise the specific replacement elements needed to be selected in the bindings tab. The variation points also refer to the VSpecs to define what abstract notion of variability the variation point actually realizes. The inclusion/exclusion of particular choices for the configuration/resolution is therefore considered.

In order to derive the configuration, the execute option in particular resolution is selected. This generates the desired models that are automatically exported back to the EPF Composer.

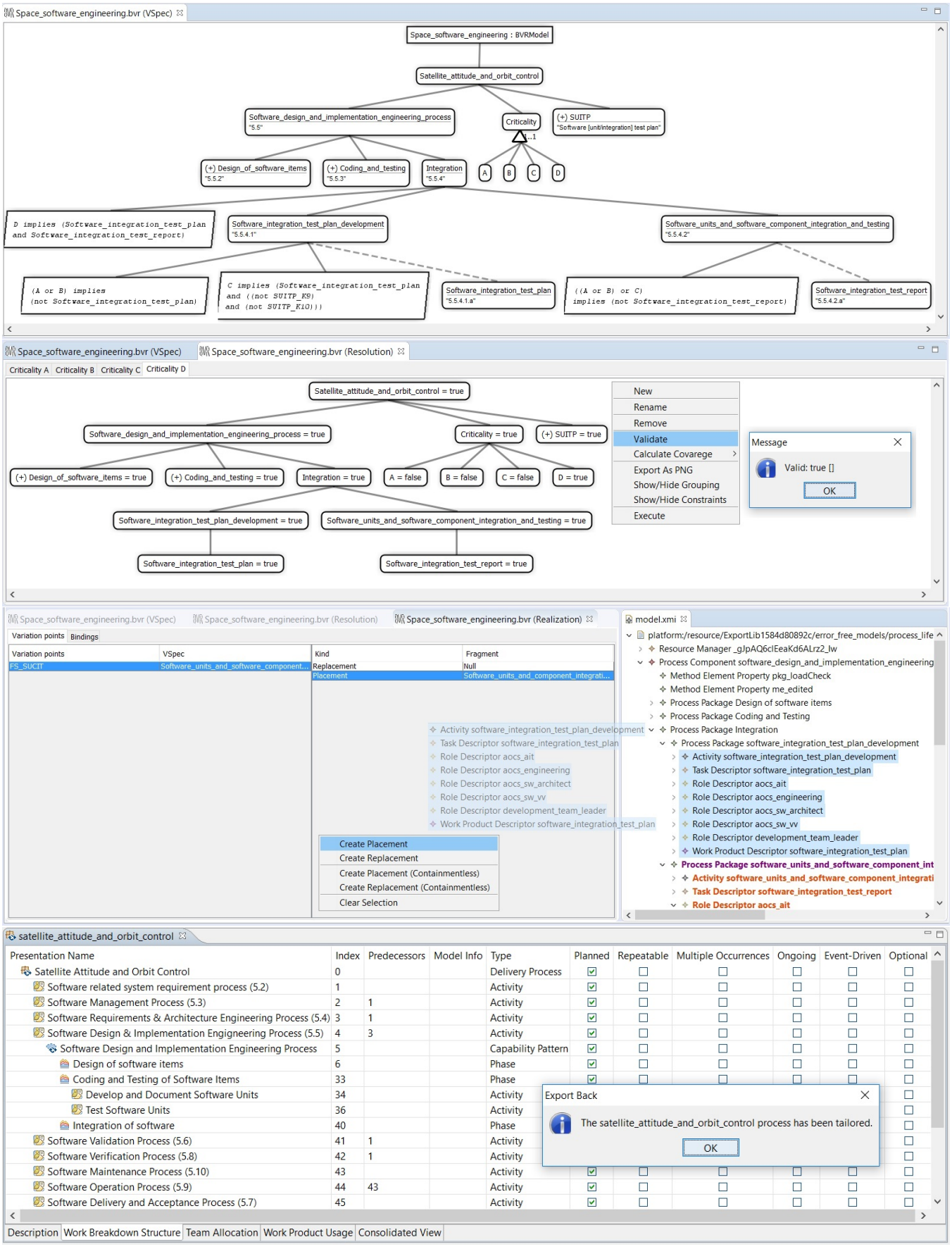

Figure 3: ECSS-E-ST-40C compliant SoPL