Development of AUTOSAR standard documents at Carmeq GmbH

Regina Hebig, Holger Giese, Kimon Batoulis,
Philipp Langer, Armin Zamani Farahani, Gary Yao,
Mychajlo Wolowyk

Technische Berichte Nr. 92

des Hasso-Plattner-Instituts für
Softwaresystemtechnik
an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 92

Regina Hebig | Holger Giese | Kimon Batoulis | Philipp Langer |
Armin Zamani Farahani | Gary Yao | Mychajlo Wolowyk

Development of AUTOSAR Standard Documents at Carmeq GmbH

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2015
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URN urn:nbn:de:kobv:517-opus4-71535
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-71535

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-317-6

This report documents the captured MDE history of Carmeq GmbH, in context of the project Evolution of MDE Settings in Practice. The goal of the project is the elicitation of MDE approaches and their evolution.

Contents

1. Introduction

This report is part of the project *Evolution of MDE Settings in Practice*[1]. The goal of the project is the elicitation of MDE approaches and their evolution. In this context, we examine the evolution history of different MDE settings in practice. This report documents the captured MDE history of Carmeq GmbH[2] and its setting.

We use the following definition for *MDE setting*: an MDE setting is the set of manual and automated activities (e. g. generations or transformations) as well as tools and (modeling) languages that are used during development.

We focus on the activities that we captured in the form of Software Manufacture Models (*SWMaMos*). We defined *SWMaMos* in [1] as a special process model, where the characterization of MDE activities captures how artifact relations change. In that context a *SWMaMo* defines artifact roles that occur during development.

In the next section we provide a summary about the examined project. In Section 3 we give an overview on the models created to describe the captured MDE settings and their histories. In Section 4 we summarize some observations about the evolution captured for the MDE settings. We analyzed the captured models with focus on the question how the MDE settings support changeability of the software under construction (Section 5) and with focus on the question how flexible the MDE settings are (Section 6).

[1] `http://www.hpi.uni-potsdam.de/giese/projekte/mde_in_der_praxis.html` (last access 3rd March 2015).

[2] `http://www.carmeq.de/` (last access 3rd March 2015).

2. Overview on Project

In the following we introduce the project under consideration with the captured MDE setting. We also captured the historic evolution of the MDE setting. Since we focused on structural evolution, we mostly did not model two different historical versions of an MDE setting when only a language or implementation of an automated activity evolved.

Development of the standard documents for AUTOSAR (AUTomotive Open System ARchitecture) started in 2004. AUTOSAR is an open standard in the automotive industry, which is acknowledged, developed, and supported by most large car manufacturers. The standard provides a basic infrastructure and interfaces for developing vehicular software and user interfaces. Carmeq supports the release management with the help of modeling tools.

The MDE setting that is captured in this report was specifically created to deal with creation and extension of AUTOSAR standard documents.

The setting that was in use at the time of the interviews included several manual and automated activities. Together, these activities start with the creation of a concept and end with its insertion in the AUTOSAR standard documents.

There are three different cases for the application of the setting. In case small changes to existing standard documents are required a Request for Changes (RfC) is created within a Bugzilla bug tracking system. On the basis of this RfC *implementation tasks* are derived. In case of larger changes a concept document is created, first. From this concept document RfCs are derived, which are a basis for later derivation of implementation tasks. Also in the case that completely new concepts are required, a concept document is created, which is then directly used to create corresponding tickets within the Bugzilla Bug tracking system.

The creation of a concept document happens manually on the basis of a Word template, by the concept owner. In case changes or extension affect already existing parts of the standard, the affected AUTOSAR[1] standard documents as well as corresponding Enterprise Architect UML models of the Basic Software (BSW) model might be used during the creation of the concept document, too. Besides

[1] `http://www.autosar.org/` (last access 3rd March 2015).

textual description of the concept document can include tables as well as a list of all AUTOSAR standard documents that are affected by the new concept. Further, the concept document might be accompanied with graphics which are sketches of the required models.

These documents are referenced in the Requests for Changes that are derived by the concept owner. To inform modelers and other colleagues about the new tasks, concept owners further define implementation tasks. These implementation tasks in Bugzilla are used for change tracking to fulfill an RfC.

The AUTOSAR standard is accompanied by three models. First, there is the AUTOSAR metamodel which defines a language for specifying software within the AUTOSAR standard. The ECUConfiguration model (EcuC) is used to define boundaries for concrete parameters within AUTOSAR models.

Finally, the Basic Software Model (BSW model) describes standardized parts of the basic software within the AUTOSAR standard. Extracts of this BSW model are used within the AUTOSAR standard documents. The BSW model is technically stored and treated as Enterprise Architect UML models.

The extension of the BSW model in conformance to a new RFC or concept document is the next step in the MDE setting for the creation and extension of AUTOSAR standard documents. The resulting Enterprise Architect (EA) models are converted automatically to Eclipse Ecore[2] to perform certain checks like a check whether preconditions for the generation are met or whether the BSW model conforms to the AUTOSAR metamodel.

For the generated Ecore models it is checked whether they are conform to the AUTOSAR metamodel. Therefore, an EMF representation of the AUTOSAR metamodel is used. As a result, email reports are generated. When problems are reported, a new modeling iteration begins.

Afterwards, an openArchitectureware (OAW[3]) workflow is used together with Apache[4] String templates to generate figures and tables on the basis of the Ecore representation of the BSW model. Tables are generated in HTML format (for AUTOSAR standard documents that are word documents) and as tex-files (for AUTOSAR standard documents that are latex documents). Figures are generated as PNG files (for AUTOSAR standard documents that are word documents), as EMF-Files(as basis for the review of the figures), and as PDF files (for AUTOSAR standard documents that are latex documents). In addition, an AUTOSAR XML file (.arxml) is generated out of specific parts of the BSW. This file is used as basis

[2] `http://www.eclipse.org/modeling/emf/?project=emf` (last access 3rd March 2015).
[3] `http://openarchitectureware.org/`(last access January 2014).
[4] `http://www.apache.org/` (last access 3rd March 2015).

for a conformance check of the BSW against an AUTOSAR XSD schema that is a derivation of the AUTOSAR metamodel.

The generation is followed by a review. It is for example checked, whether the layout changed in comparison to former versions of the figures and tables. To support these manual reviews, diffs with the former versions are created (using the SVN diff tool for tables and the tool Beyond-Compare) for figures. When problems are encountered during the reviews, a new modeling iteration begins.

In parallel to modeling and generation of figures and tables, the concept is included into existing or new AUTOSAR standard documents. When the concept is novel, a new standard document is created manually as a Word document based on the concept document and Word templates for software specifications in AUTOSAR. The tables and figures are integrated semi-automatically in the existing AUTOSAR standard documents in Word using macros. Modules in the standard documents affected by the concept are refined manually and references are adapted. The references are validated automatically and after a final review the updated AUTOSAR standard document is released.

3. Model Guide

In this section, an overview of the models created during the study is given.

We captured for each setting the history together with reasons and triggers for changes. Such triggers could be events in the project, external events, or changes in interacting settings. For each version of a setting we documented the change in contrast to the former version and some further information, where known. First, we captured where the demand for the new form of the MDE setting stems from (*Demand*). Second, we captured the organizational context in which the changes on the MDE setting were applied (*Creation*). Finally, we summarized the motivation for the change (*Motivation*).

The history is shown in:

- Appendix: models\history_overview

The first model illustrates the process of the setting (i. e., how the activities are usually applied). A process view can be used to show where activities are alternatives for each other. Each of the activities is represented in a Software Manufacture Model. Further, the process can define a restrictive order of activities.

The process is shown in:

- Appendix: models\process_versions

The second model shows all activities in Software Manufacture Model (SWMaMo) [1] notation. A simplified notation is used (compared to [1]), which is summarized in Figure 3.1 and explained in the following.

This notation depicts activities as rounded rectangles, artifacts as angular rectangles. The degree of automation of an activity is illustrated with a small box at the upper border within the activity. When the box contains a gear wheel the activity is fully automated, an activity containing a stick figure is performed manually and when the box contains a diagonal line the activity is performed semi-automatically.

Activities have pins illustrated as white squares attached to the outer bound. The pins represent the connection of an artifact to the activity either as an input or as an output for the activity. Two special kinds of pins are the *definition pin* and the *manual enrichment pin*. A *definition pi*n indicates that an input artifact that

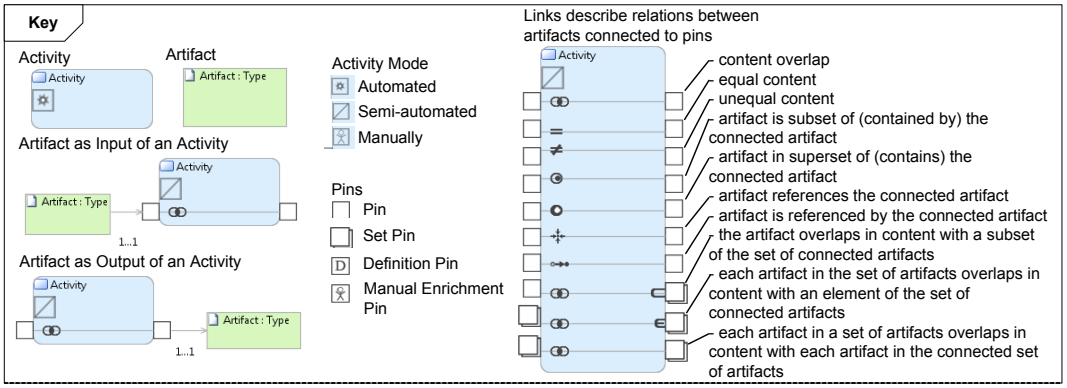

Figure 3.1.: Key for used notation of Software Manufacture Models

is connected to this pin is a specification for how the activity is executed (e. g., a transformation specification). A *manual enrichment pin* indicates that an output artifact that is connected to this pin does not only contain content that was taken from other artifacts, but is also enriched with further content manually.

Further, all pins can be *set pins* as well. A *set pin* indicates that not only a single artifact, but a set of artifacts is input or output. Constraints on the cardinality of artifacts can be notated at the connection between artifact and pin.

What happens during an activity is described by links, which are represented by edges between two pins. Links between two input artifacts indicate what relations between artifacts are preconditions for the execution of the activity. Links between input and output artifacts or between two output artifacts indicate what relations between artifacts are established during the execution of the activity. Links are directed, i. e., one of the pins is the start pin and the other is the end pin. This does not depend on the question whether a pin is an input or output pin of the activity. An icon on the one side of the link indicates what relation is established between two artifacts. The relation is directed from the artifact, where the icon is located, to the other artifact. There are basically two kinds of relations. First, there are content relations, which indicate how content of two artifacts is interrelated. This can be overlap relations (to indicate that both artifacts overlap in content), equals relations (to indicate that content of both artifacts is equal), or unequal relations (to indicate that content of both artifacts is unequal). Second, there are artifact relations. This can be containment relations, that describe that one artifact is contained by another artifact (*subset of*) or contains another artifact (*superset of*). Further, artifact relations can describe how artifacts reference each other (*references* or *referenced by* relations).

Links are not only formulated between normal pins, but also between set pins. To indicate the scope of affected artifacts, each relation has a prefix. By default, each relation has a for-all-for-all prefix, which is not annotated visually in the diagrams. The for-all-for-all prefix indicates that for each artifact connected to the start pin of the link the relation holds to each artifact connected to the end pin of the link. Alternatively, there are the for-all-subset prefix and the for-all-element prefix. These prefixes indicate that for each artifact connected to the start pin of the link the relation holds to a subset of the set of of artifacts (or only one artifact in the set of artifacts, respectively) connected to the end pin of the link.

Figures 3.2, 3.3, and 3.4 show three examples for activities specified in Software Manufacture Model notation.

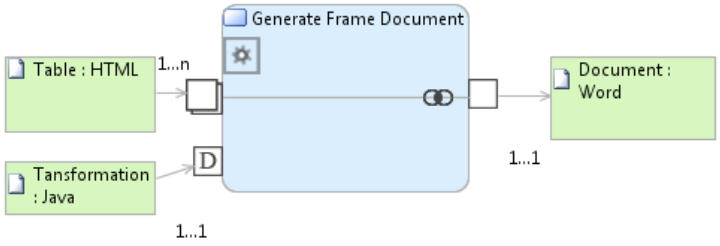

Figure 3.2.: Example for notation of an activity, where multiple HTML tables are consumed to produce a Word document (in consequence the resulting Word document overlaps with each HTML table from the input). The applied transformation is implemented in Java (indicated by the definition pin)

The Software Manufacture Model is shown in:

- Appendix: models\swmamo_activities (parts 1 to 7)

The third model is the megamodel that describes how the different artifacts used within the MDE setting are interrelated. The megamodel is generated from the SWMaMo, which already includes the information what artifacts and relations are created.

The same artifact (identified by the combination of name and type) might be modeled multiple times within the software manufacture model (to reach a clear structure within the SWMaMo). However, the same artifact is represented only once within the megamodel. In Figure 3.5 a notation key for relations in megamodels

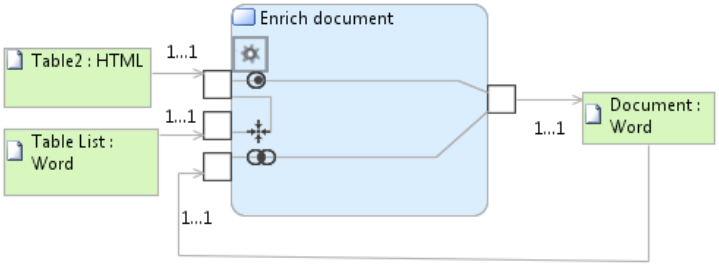

Figure 3.3.: Example for the notation of an activity, where the relation between two consumed artifacts is precondition for the execution of the activity. A table that is used for this activity needs to be referenced by a table list. In consequence of the execution of the activity, the HTML table is integrated into the word document, which is manipulated during the activity (i. e., the produced version of the document overlaps in content with its consumed version)

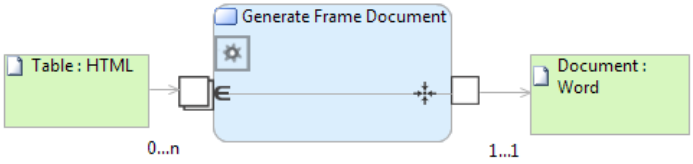

Figure 3.4.: Example for the notation of an activity, where a Word document is generated that references one of the tables within the set of tables that are consumed by the activity

together with an example is shown. Possible relations are *contains* (i. e., one artifact contains the other, for example the Word document contains the Table 2), *equals* (i. e., the content of both artifacts is equal, but might be represented using different languages), *overlaps* (i. e., the content of both artifacts is overlapping, but might be represented differently; for example the word document overlaps in content with the HTML table), *unequals* (i. e., the content of both artifacts is unequal), *references* (i. e., one artifact references the other; for example the TableList references Table 2), and *unspecified* (which can occur during modeling, when the relation is not yet fully specified).

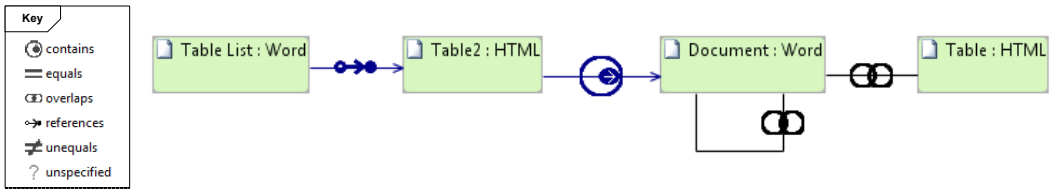

Figure 3.5.: Example for a megamodel, generated from the SWMaMo with the two activities shown in Figures 3.2 and 3.3. Table list references Table 2 which is contained by a *Word document*. The Word document further overlaps in content with itself (since it can be changed during development) and the artifact *Table*

The megamodel is shown in:

- Appendix: models\megamodel (overview and parts 1 to 8)

4. Summary on Evolution

In the last 10 years, the process of creating new AUTOSAR standard documents has changed several times. During the interviews we captured 5 structural evolution steps, which we summarize and categorize in Table 4.1. We differentiate between structural and non-structural changes in the MDE setting. Non-structural changes are substitutions of used modeling languages or implementations of automated activities. Structural changes can affect the number of tools, automated or manual activities, artifacts, and languages as well as the order of manual and automated activities.

The results are 6 historical settings: in the first setting, no modeling was used (**pre mod**), the second setting is the result of the introduction of modeling with enterprise architect (EA) and Ecore (**models**), the third setting is the result of the introduction of macros that support the integration of figures and tables into standard documents (**macro**), the fourth setting is the result of the introduction of automated diffs on tables and figures to support the manual review process (**diff**), the fifth setting is the result of the introduction of the CI server (**ci**), and finally the current setting is the result of the implementation of an additional importer to move EA models into Eclipse (**current**). What is not captured here is an additional change in the setting that led to the introduction of latex as an alternative technology to formulate the standard documents.

Figure 4.1 visualizes how the process changed within the evolution steps. For readability, a high-level view on the process (e. g., without responsibility issues) is shown. The activities and control flow that have been added during the evolution of the process are highlighted by different line styles and bold-facing of text.

Besides the 5 explicitly captured evolution steps there are two kinds of changes that happen regularly to the setting. First, the AUTOSAR metamodel changes regularly, and consequently the modeling language, as well as conformance checks are updated. Second, the generator decides where figures and tables are stored in the repository. The macros that are used for integration of the figures and tables into the documents implement the access to this repository. Each time this structure of the repository is changed, the implementations of the macros need to be updated.

Table 4.1.: Captured structural evolution steps

		S1 (pre mod → models)	S2 (models → macro)	S3 (macro → diffs)	S4 (diffs → ci)	S5 (ci → current)
Non-Structural Changes						
[C1]	exchange automated activity				✓	✓
[C2]	exchange language					
Structural Changes						
[C3]	change number of artifacts	✓		✓		
[C4]	change number of languages	✓		✓		
[C5]	change number of manual activities		✓			
[C6]	change number of tools	✓	✓	✓	✓	✓
[C7]	change number of automated activities	✓	✓	✓		
[C8]	change order of manual / automated activities	✓	✓	✓		

Different motivations for the changes on the MDE settings can be observed.

Motivation for introducing Modeling Modeling was introduced as an integral part of the standard document creation process in 2004. The UML modeling tool Enterprise Architect was chosen, because of its synchronization mechanism that allows modelers to work in parallel on the same model. The motivation for bringing modeling into the company was to cope with the complexity of the documented concepts. The additional usage of Ecore was motivated by some shortcomings of the Enterprise Architect modeling tool. In particular, the performance of the Enterprise Architect API was not sufficient to implement checks on the models.

Motivation for introducing Macros Word Macros were already in use for layouting purposes before modeling became an issue. The concept was reused to automate the integration of figures and tables into the standard documents.

Motivation for introducing Diffs There is no guarantee that models, which are extracted from the enterprise architect, are always similar in layouting. To ensure that different generated versions of the same model do not unnecessarily differ from each other, manual reviews of the models are performed. To support this task, automated diff-tools on figures and tables were introduced. The resulting diffs support the manual identification of the deviations.

Motivation for usage of CI Server To minimize platform specific differences between generation results, a CI server for the generation was introduced.

Motivation for the introduction of an additional EA-to-Eclipse importer Finally, an additional importer for Enterprise Architect models was implemented. This new importer is used for the generation of BSW service interfaces, and translates other aspects of the EA model than the old importer. The old importer is still used for generation of other artifacts.

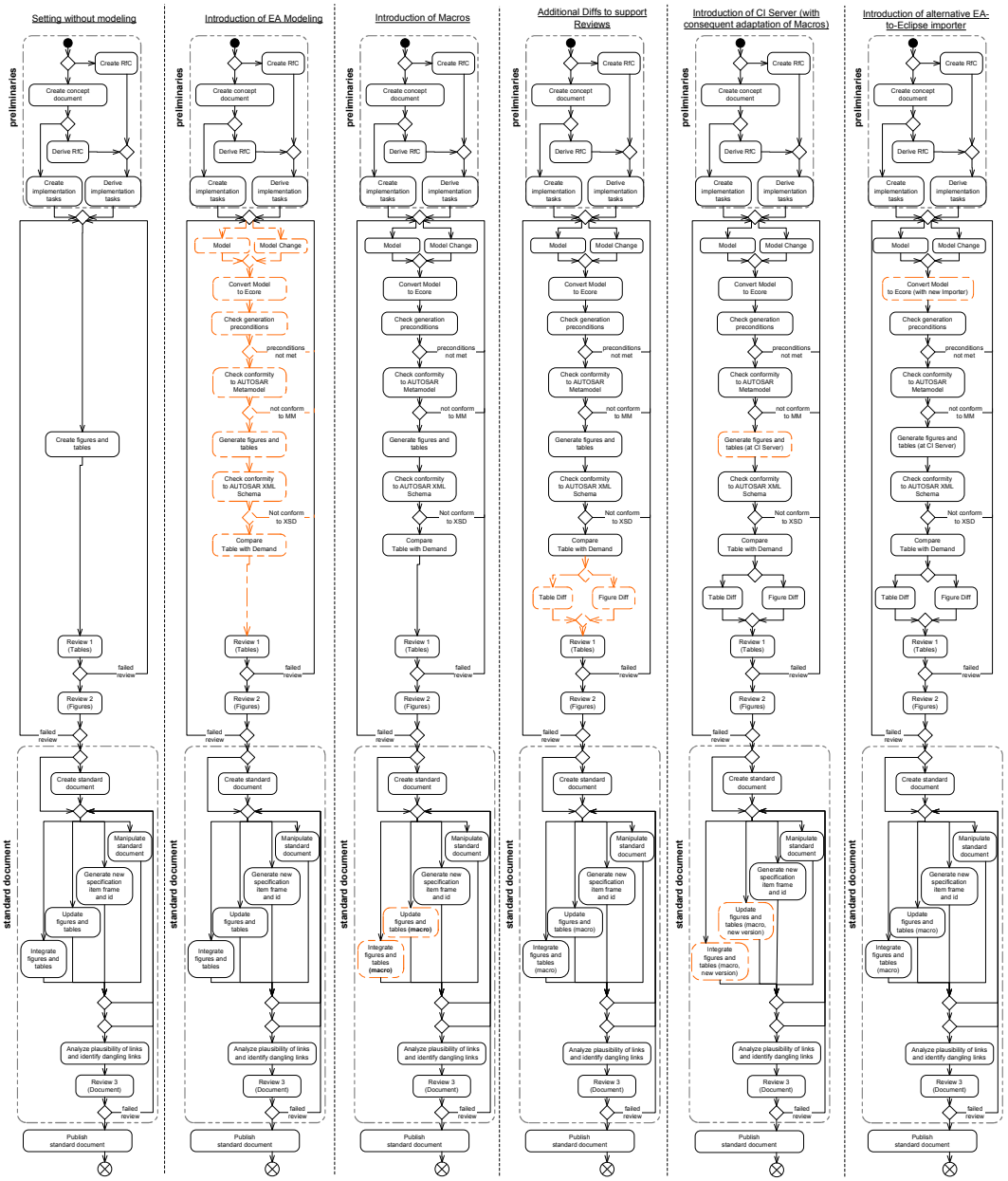

Figure 4.1.: Evolution steps of modeling tasks within the document creation setting. The illustration does not include the evolution of preceding and succeeding processes e. g. for the creation of concepts and concept documents, or for the detailed work on the standard documents. Activities and control flow in one evolution step that changed with respect to the previous one are highlighted by dotted lines.

5. Changeability support of the MDE settings

An MDE setting can influence the changeability of the software or artifacts under construction in multiple ways. The main positive influences stem from the way how information is split over different artifacts and represented on different abstraction levels as well as from the degree of automation reached. However, complex MDE settings can also imply risks for the changeability of software. For example, the order of manual and automated activities can be quite complex and lead to long chains of activities that have to be performed to implement a change.

In the following, we analyze the length of the activity chain that has to be applied to implement a change. Therefore, we count the manual activities that have to be applied on different artifacts to bring a change in a specific artifact consistent into the implementation. Note that when counting automated activities we do not count each automated activity individually. Rather, we count one continuous sequence of automated activities as one automated activity. For example, the continuous sequence of automated activities within the OAW workflow is counted as a single automated activity from the modeler's point of view:

> *"OAW workflow" = "Convert Model to Ecore" → "Check generation preconditions" → "Check Conformance to AUTOSAR Metamodel"→ "Generate Figures and Tables" → "Check Conformance to AUTOSAR XML Schema"*.

Similarly, if two manual activities can be executed in one step, they are counted as a single activity here as well. In case of the manipulation and adaption of the word standard document, the macros for the automated integration of the figures and tables are used several times. Both the manipulation and the integration using the macros affect the same main artifact. We expect that this is experienced as inherent part of the manipulation task, by the document owners. Therefore, the combination of possibly multiple calls of macros (for integration of figures and tables, but also other macros) and the manipulation of the word document are counted as a single semi-automated activity in the following.

> *"Work on standard document" = "Manipulate standard document"*

→ "Integrate figures and tables" → "Update figures and tables" → "Generate new specification item frame and id" → "Manipulate standard document" → etc.

When analyzing the length of the activity chain for the introduction of changes, two groups of artifacts (notated here in form of `RoleName:Type`) are considered. First, there are start artifacts, i. e., artifacts which are manipulated or created in order to introduce a change. Start artifacts for the setting documented here are:

- a `Concept Document:Word Document (.doc)` that introduces a new concept (`start 1`),
- a `Concept Document:Word Document (.doc)` that describes changes to existing concepts (`start 2`), and
- a `Request for Change:Bugzilla RFC` that describes a desired smaller change (`start 3`).

Second, there are target artifacts, i. e., the artifacts that are result of the setting and should include the change at the end. Target artifacts for the setting documented here are:

- a new `AUTOSAR Standard Document:Word Document (.doc)` (`target 1`),
- an existing `AUTOSAR Standard Document:Word Document (.doc)` (`target 2`),
- a new `AUTOSAR Standard Document:Latex Document (.tex)` (`target 3`), and
- an existing `AUTOSAR Standard Document:Latex Document (.tex)` (`target 4`).

The length of activity chains is the number of activities that need to be executed to propagate a change from a start artifact to a target artifact. Thus, there is a length of activity chain for each pair of start and target artifacts. Iterations within these chains, which might be caused by failing checks, are not taken into account (i. e., the length of activity chains represents a minimum of performed activities, which can be reached when no checks fail).

The main part of the activity chain (between *"Model"* or *"Model Change"* and *"Review 2 (Figures)"*) is equal for all pairs of start and target artifacts:

"Model" (or *"Model Change"*) → *"OAW workflow"* → *"Compare Table with Document"* → *"Table Diff"* → *"Figure Diff"* → *"Review 1 (Tables)"* → *"Review 2 (Figures)"*.

Thus, the length of this main part of the activity chain is seven. Four manual and three automated activities are involved. One of the manual activities (*"Compare Table with Document"*) is optional and occurs only seldom. Therefore, the minimum length of activity chain for this main part is counted as six (three manual and three automated activities) in the following consideration.

For the different start artifacts the start of the activity chain differs.

For `start 1` it is two manual activities:

"Create concept document" → *"Create tickets"*.

For `start 2` it is three manual activities:

"Create concept document" → *"Derive RFC"* → *"Derive implementation tasks"*.

For `start 3` it is two manual activities:

"Create RFC" → *"Derive implementation tasks"*.

Finally, the end of the activity chain differs for the different target artifacts.

For `target 1` it is four activities (2 manual activities, 1 semiautomated activity, and 1 automated activity):

"Create standard document" → *"Work on standard document"*→ *"Analyze plausibility of links and identify dangling links"*→ *"Review 3 (document)"*.

For `target 2` it is three activities (1 manual activity, 1 semiautomated activity, and 1 automated activity):

"Work on standard document"→ *"Analyze plausibility of links and identify dangling links"*→ *"Review 3 (document)"*.

For the creation and manipulation of the Latex targets no macros are used. Figures and tables are directly introduced by the concept owner per links. Checks of links within the documents are a by result of the compilation.

For `target 3` it is four activities (3 manual activities, 1 automated activity):

"Create standard document (latex)"→ *"Manipulate standard document (latex)"*→ *"Compile latex document to PDF"*→ *"Review 3 (document latex)"*.

For `target 4` it is three activities (2 manual activities, 1 automated activity):

"Manipulate standard document (latex)"→ "Compile latex document to PDF"→ "Review 3 (document latex)".

All in all, the activity chains are between 11 and 12 activities long, containing between four and five automated and semiautomated activities. In the following, the lengths of activity chain for all pairs of start and target artifacts are listed. Note: It is assumed that only the introduction of a new concept leads to new standard documents.

- `start 1` to `target 1`: 12 activities
  (7 manual, 1 semiautomated, and 4 automated activities)
- `start 1` to `target 3`: 12 activities
  (8 manual and 4 automated activities)
- `start 2` to `target 2`: 12 activities
  (7 manual, 1 semiautomated, and 4 automated activities)
- `start 2` to `target 4`: 12 activities
  (8 manual and 4 automated activities)
- `start 3` to `target 2`: 11 activities
  (6 manual, 1 semiautomated, and 4 automated activities)
- `start 3` to `target 4`: 11 activities
  (7 manual and 4 automated activities)

6. Evolvability of the MDE settings

MDE settings might evolve over time. The setting itself can influence how easy or difficult such evolution is (evolvability). Concerning the setting that is documented in this report some evolvability issues attracted our attention during the interviews.

First, there is an implicit interrelation between the implementation of the generator, which is used to create the figures and tables, and the implementation of the macros, which are used to integrate the generated artifacts into the Word documents. The generator is executed by the CI-Server and generates the figures and tables into a given repository structure. On the other hand, the macros extract the figures and tables from this repository structure (strictly speaking from a structurally mirrored repository where the figures and tables are stored after the manual reviews). Thus, macros implicitly implement the same knowledge about the repository structure as the generator does. In consequence, each time this repository structure is evolved together with the generator the macros need to be changed, too.

Two further evolvability issues arise from the special situation that the AUTOSAR metamodel is still under development and subject to regular evolution, too. Therefore, a further look at the modeling technology is required here. The AUTOSAR standard specifies on the one hand the AUTOSAR metamodel. On the other hand, parts of the software and with it parts of the basic software model are already specified. This BSW needs to be conform to the AUTOSAR metamodel.

Now both, AUTOSAR metamodel and BSW, are still under development. Therefore, both are modeled using the tool Enterprise Architect as EA-UML models. The UML model of the BSW is conform to an AUTOSAR Profile within the Enterprise Architect, which, to a certain degree, ensures conformance to the AUTOSAR metamodel. Since this EA AUTOSAR Profile ensures no full conformance, a second AUTOSAR Profile is used within Eclipse after the BSW is transformed into an Ecore model. The actual and final conformance check, is performed on the AUTOSAR XML file that is the result of the generation. This file is checked against the XSD that is generated from the AUTOSAR metamodel in the EA.

Consequently, there are different artifacts that include information about the AUTOSAR metamodel: the AUTOSAR metamodel within the Enterprise Architect,

the EA UML profile for AUTOSAR, the Ecore AUTOSAR profile and the XSD that specifies how AUTOSAR XMLs look like.

On the one hand, this situation is disadvantageous for evolution of the metamodel. Only the XSD is generated automatically from the AUTOSAR metamodel. In contrast, the profiles need to be adapted manually, when the metamodel evolves. This leads to additional manual effort.

On the other hand this constellation has also advantages for the evolution of the metamodel, since it supports the co-evolution of the BSW. A classic problem when evolving a metamodel is, that the editors for the models do evolve, too. The new editor might not be capable of manipulating models that conform to the old version of the metamodel. Further, all existing models that are in use need to be adapted to the new metamodel. To what degree this process can be automated is still subject to research. Thus, with constrained available human resources the need of an immediate adaption of the existing models can be disadvantageous. In contrast, the constellation that the original BSW model is a UML model, allows modelers to postpone the adaption to the new metamodel and focus on tasks with higher priority.

A final evolvability issue was mentioned by one of our interviewees. Although the usage of Ecore as a basis for generation was a pragmatic solution and not part of the AUTOSAR standard, today it is difficult to remove this step from the process. The reason is that several customers started to reuse the generator. Thus, an evolution of the setting in a direction where Ecore is no longer used, would affect more engineers than only the modelers at Carmeq.

7. Conclusion

This report documents the MDE setting that is applied at Carmeq to develop and extend the AUTOSAR standard specification. The documentation is accompanied by a set of models that document the setting. Further, it is documented how the MDE setting evolved over time. The MDE setting was analyzed with focus on the changeability for the artifacts under construction. Finally, some issues on evolvability of the MDE setting were discussed.

Acknowledgments

We are grateful to Carmeq GmbH and our interviewees for their support of this study. Further, we like to thank Johannes Dyck for his feedback and help.

References

[1] R. Hebig, A. Seibel, and H. Giese. Toward a Comparable Characterization for Software Development Activities in Context of MDE. In *Proceedings of the 2011 International Conference on Software and Systems Process*, ICSSP '11, pages 33–42, New York, NY, USA, 2011. ACM.

A. Models

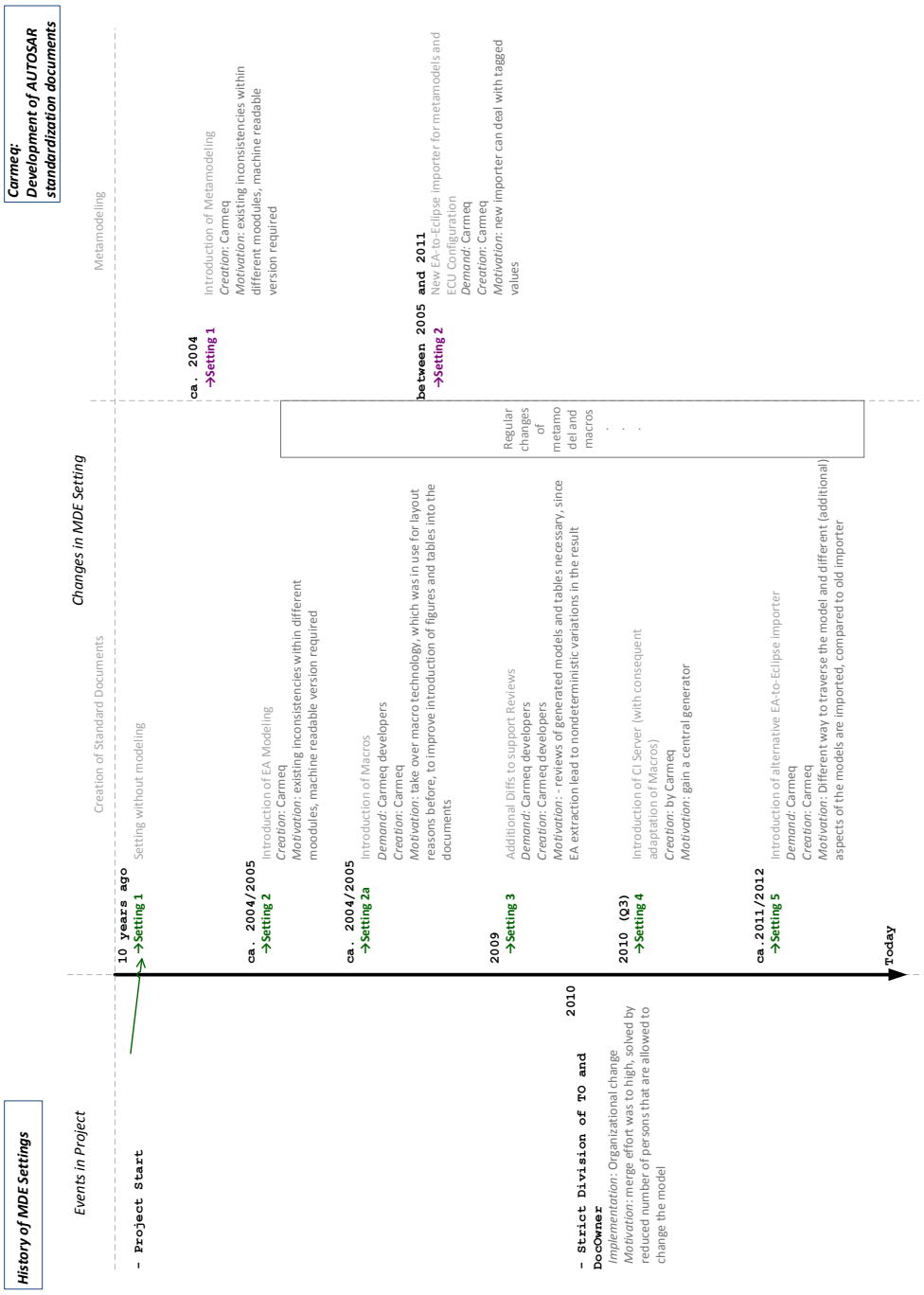

Figure A.1.: models\history_overview

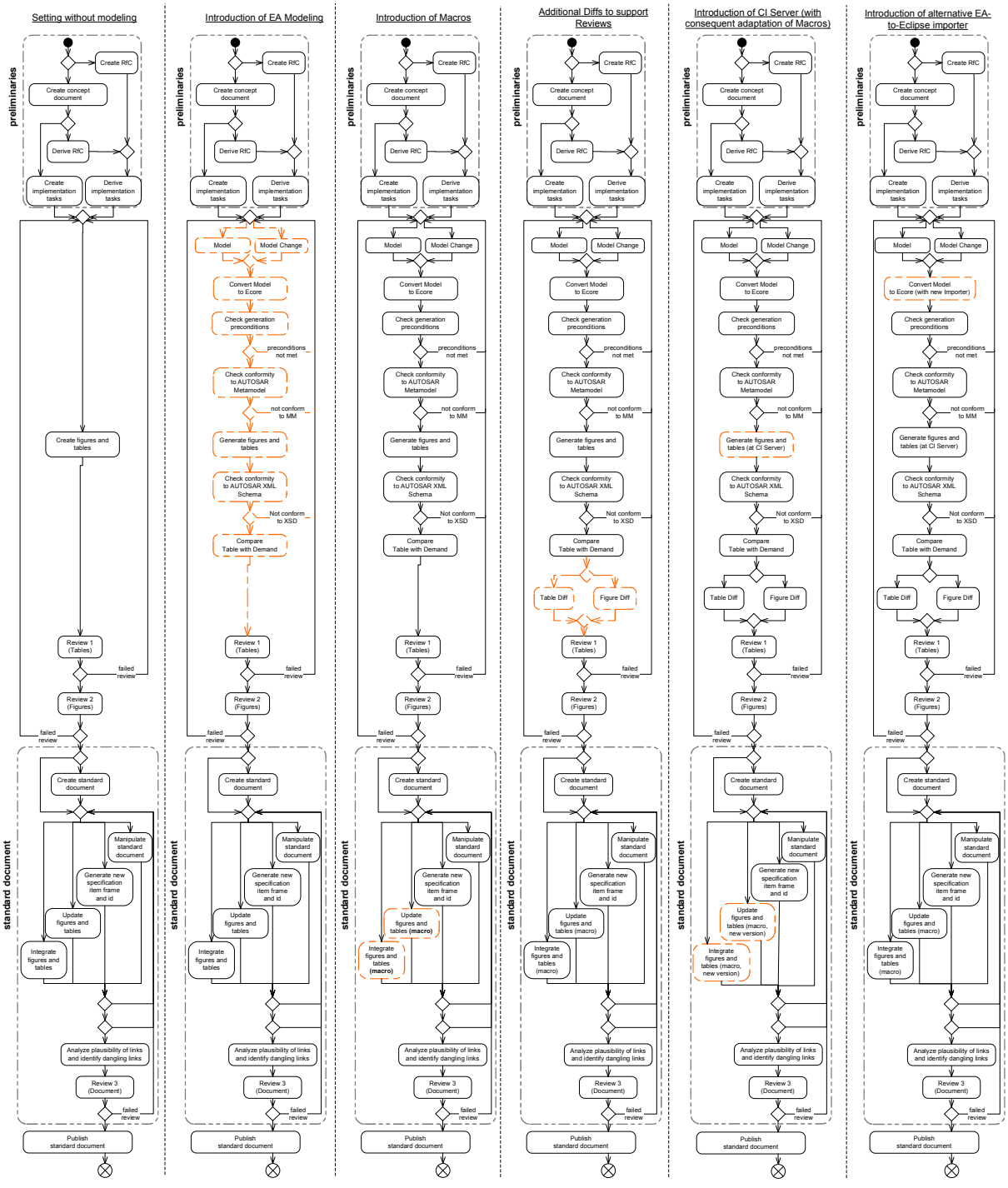

Figure A.2.: models\process_versions

A. Models

Preliminaries

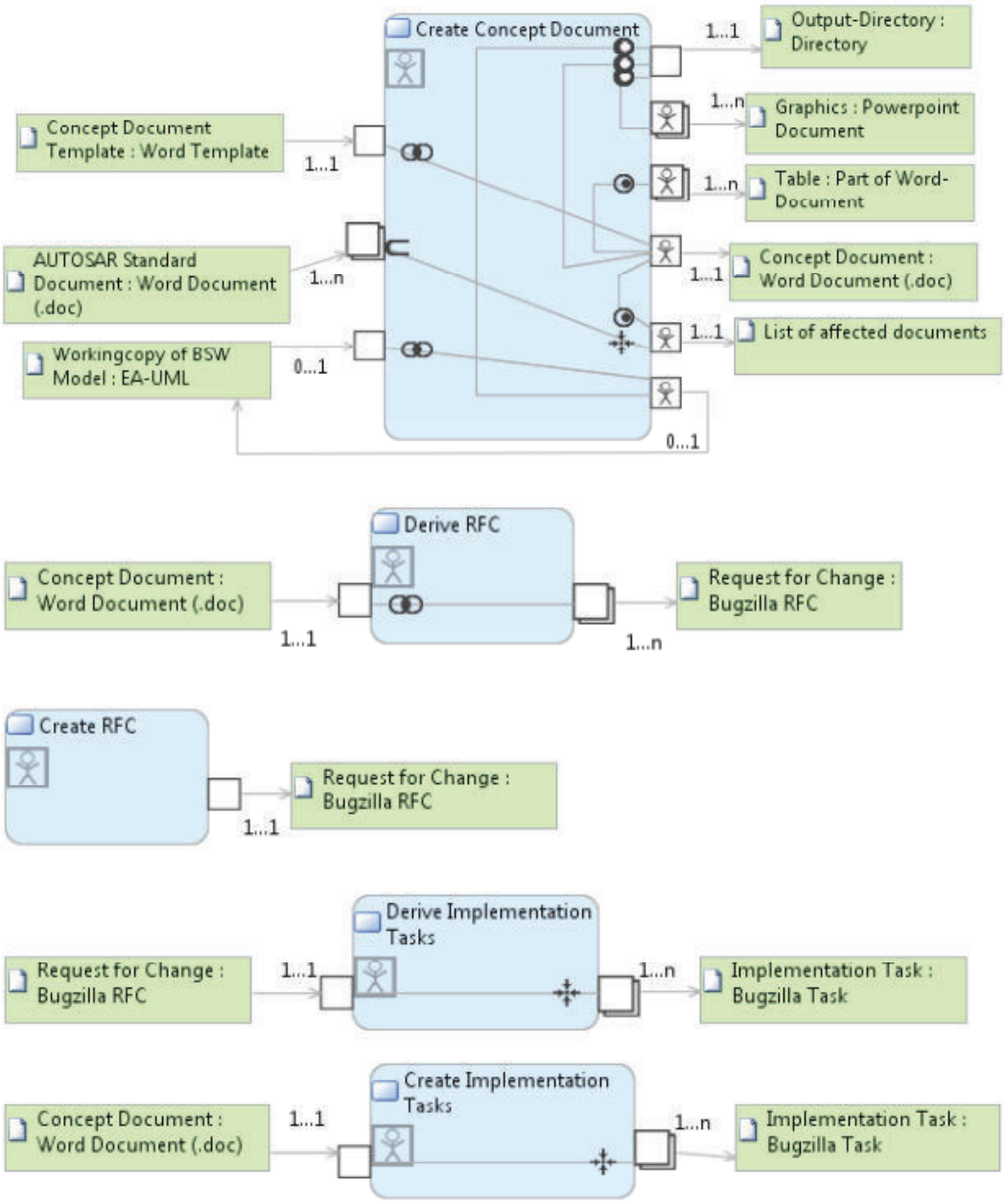

Figure A.3.: models\swmamo_activities, part 1 of 7

Main activities

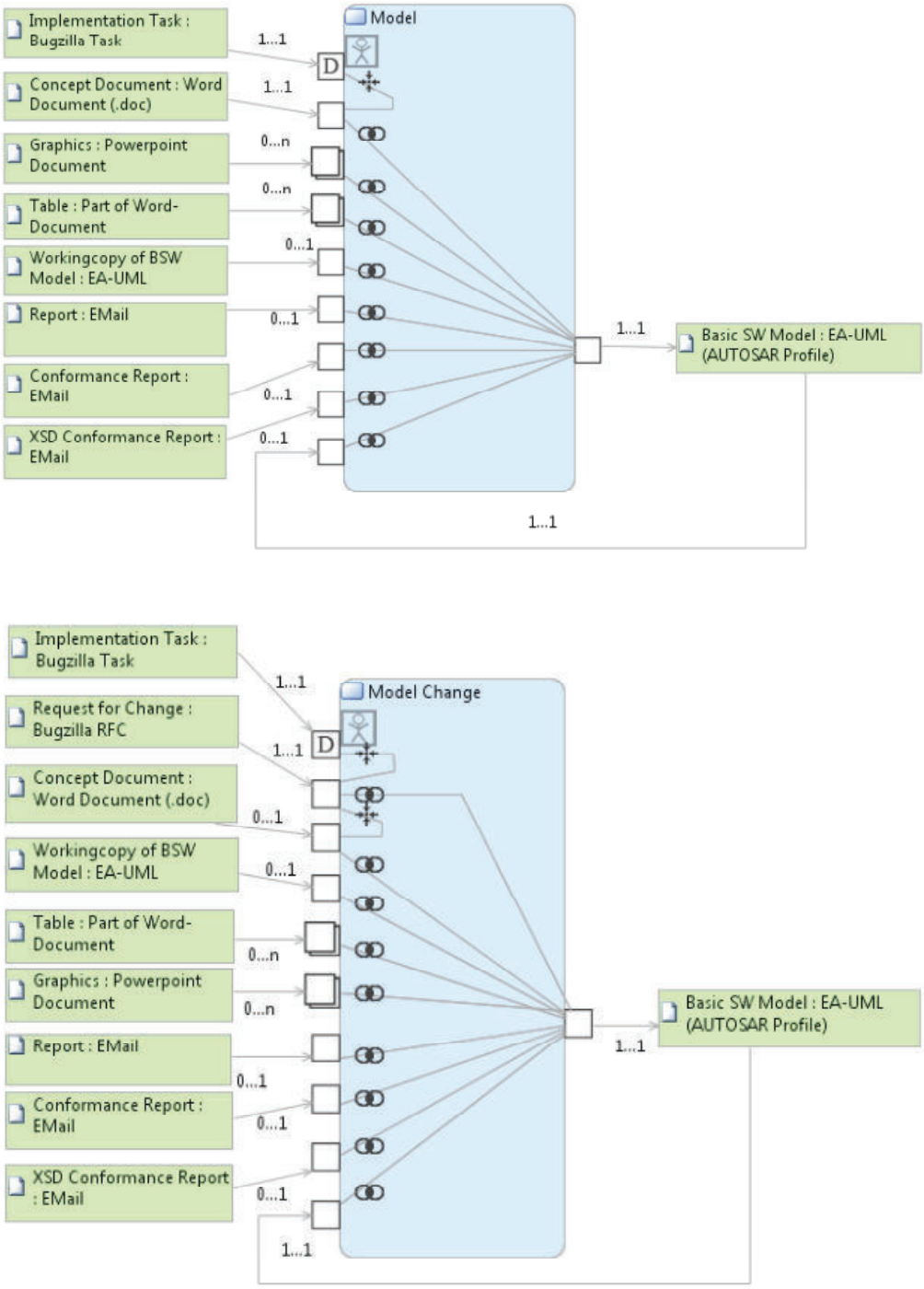

Figure A.4.: models\swmamo_activities, part 2 of 7

A. Models

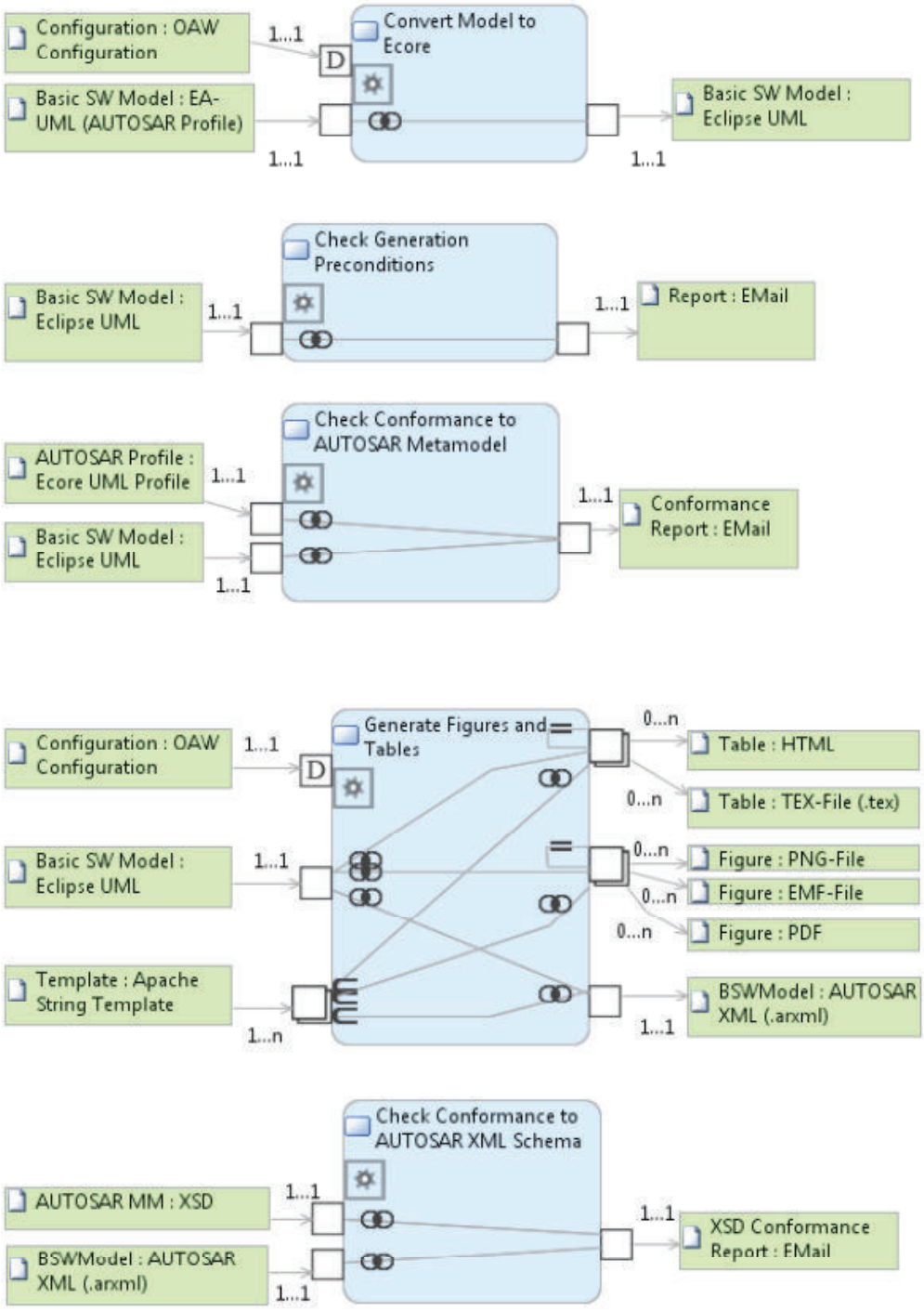

Figure A.5.: models\swmamo_activities, part 3 of 7

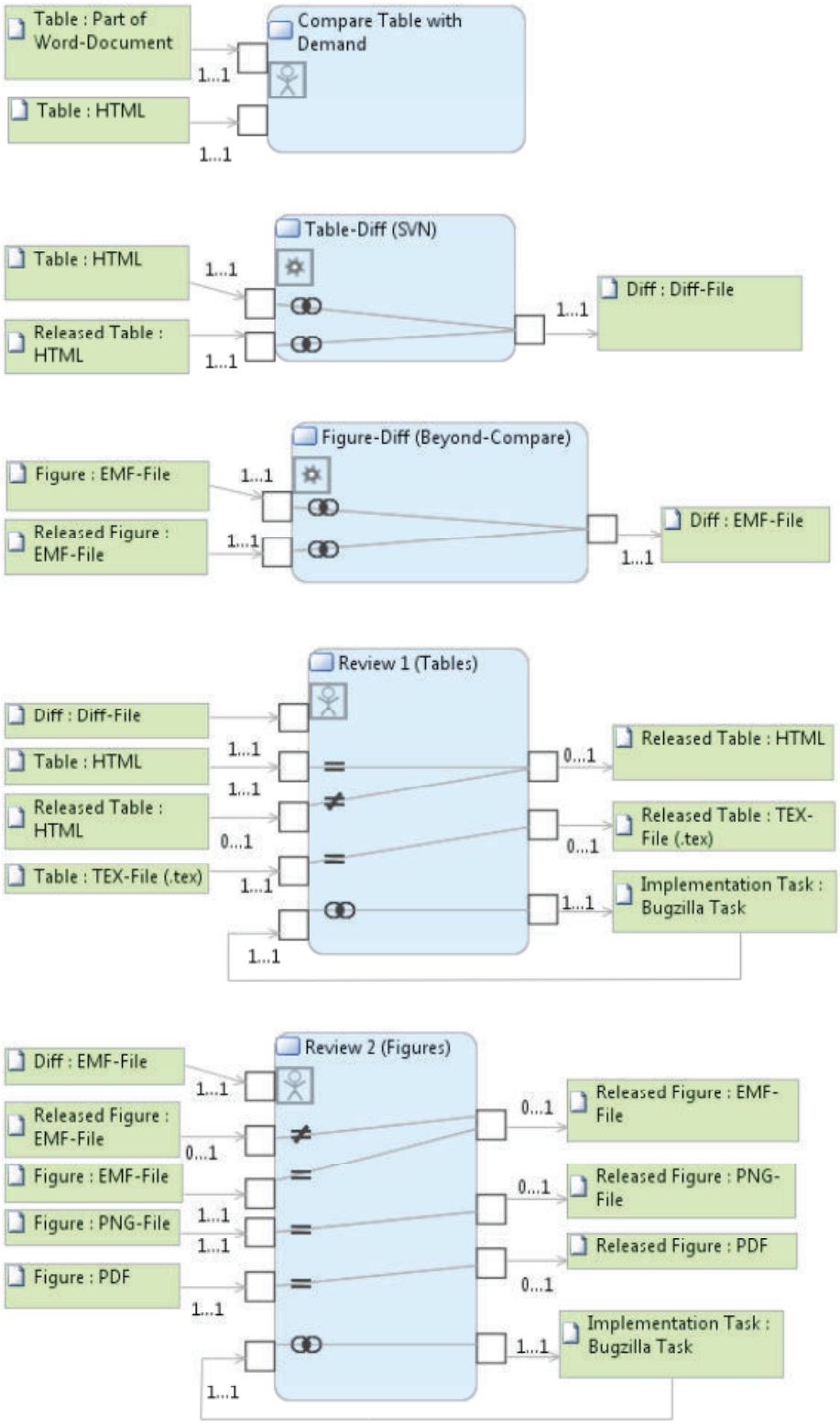

Figure A.6.: models\swmamo_activities, part 4 of 7

A. Models

Standard Document Word

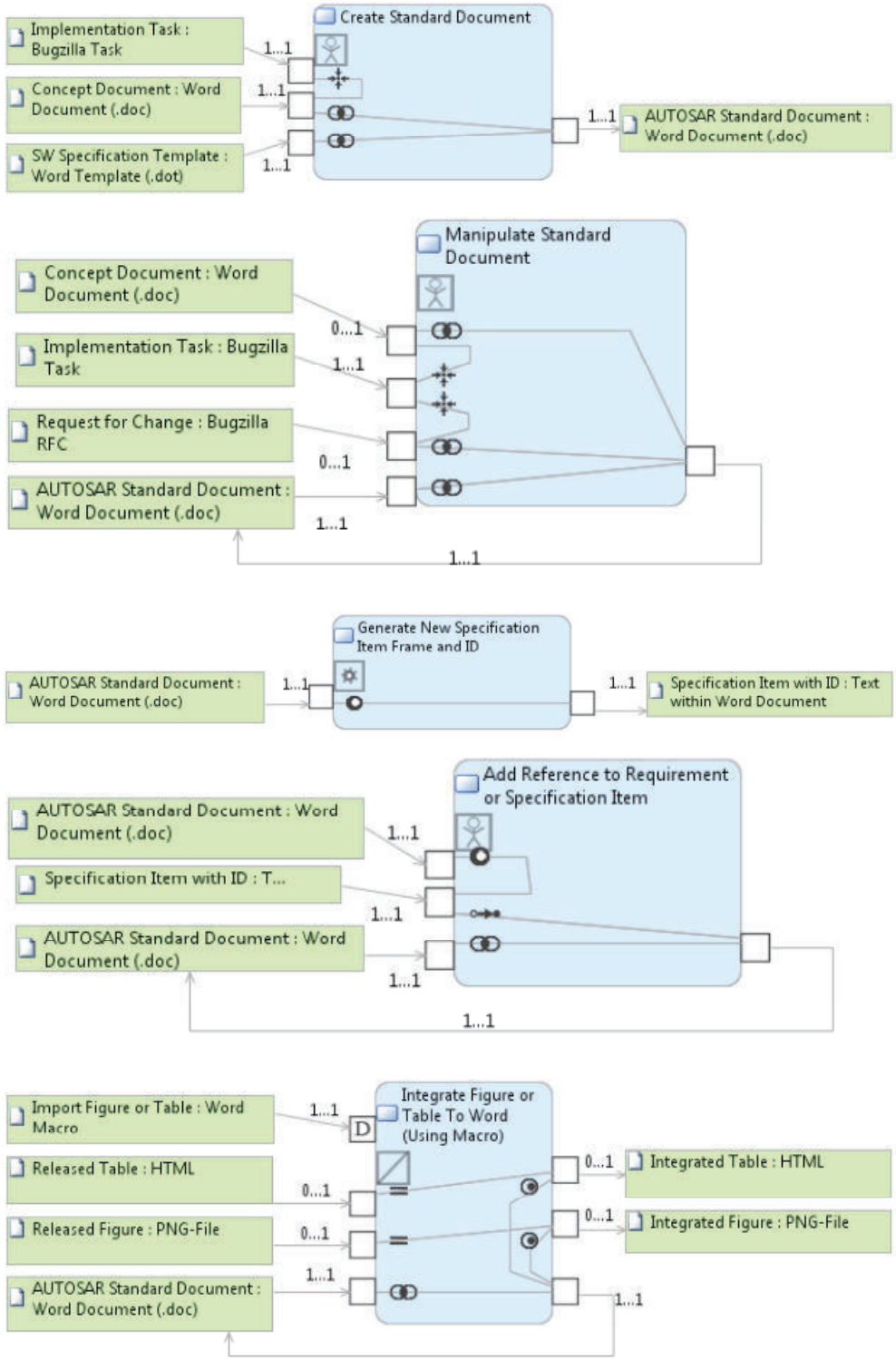

Figure A.7.: models\swmamo_activities, part 5 of 7

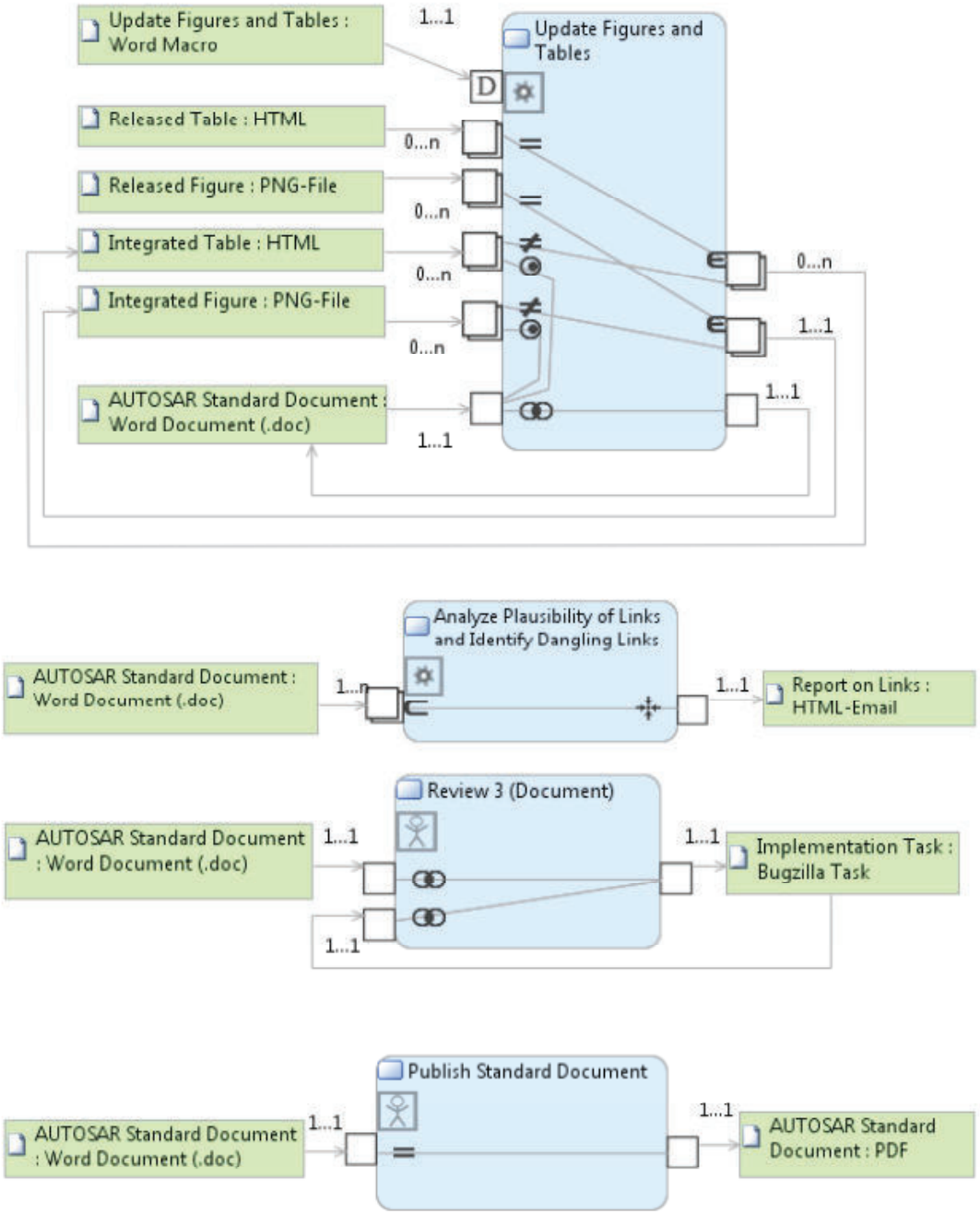

Figure A.8.: models\swmamo_activities, part 6 of 7

A. Models

Standard Document Latex

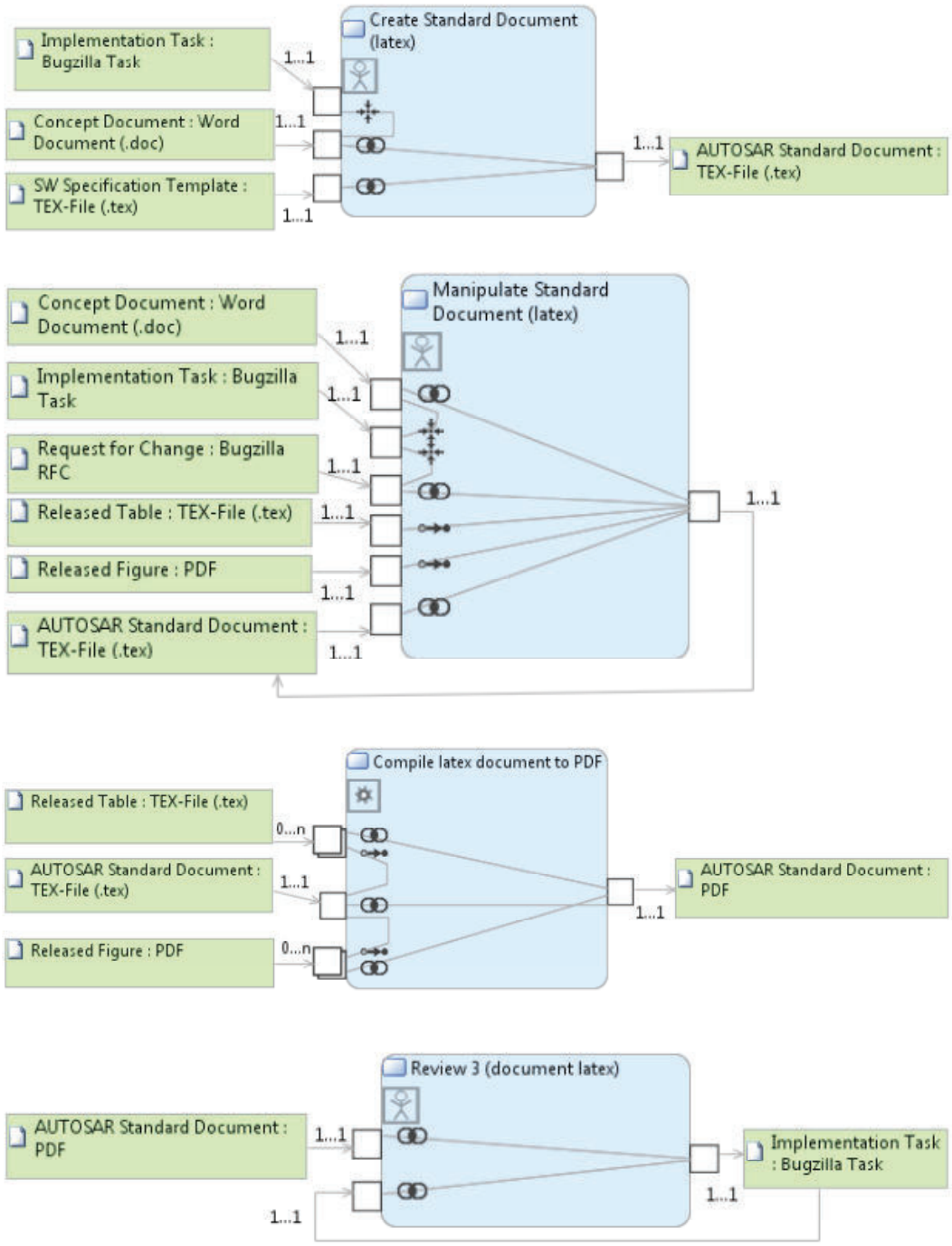

Figure A.9.: models\swmamo_activities, part 7 of 7

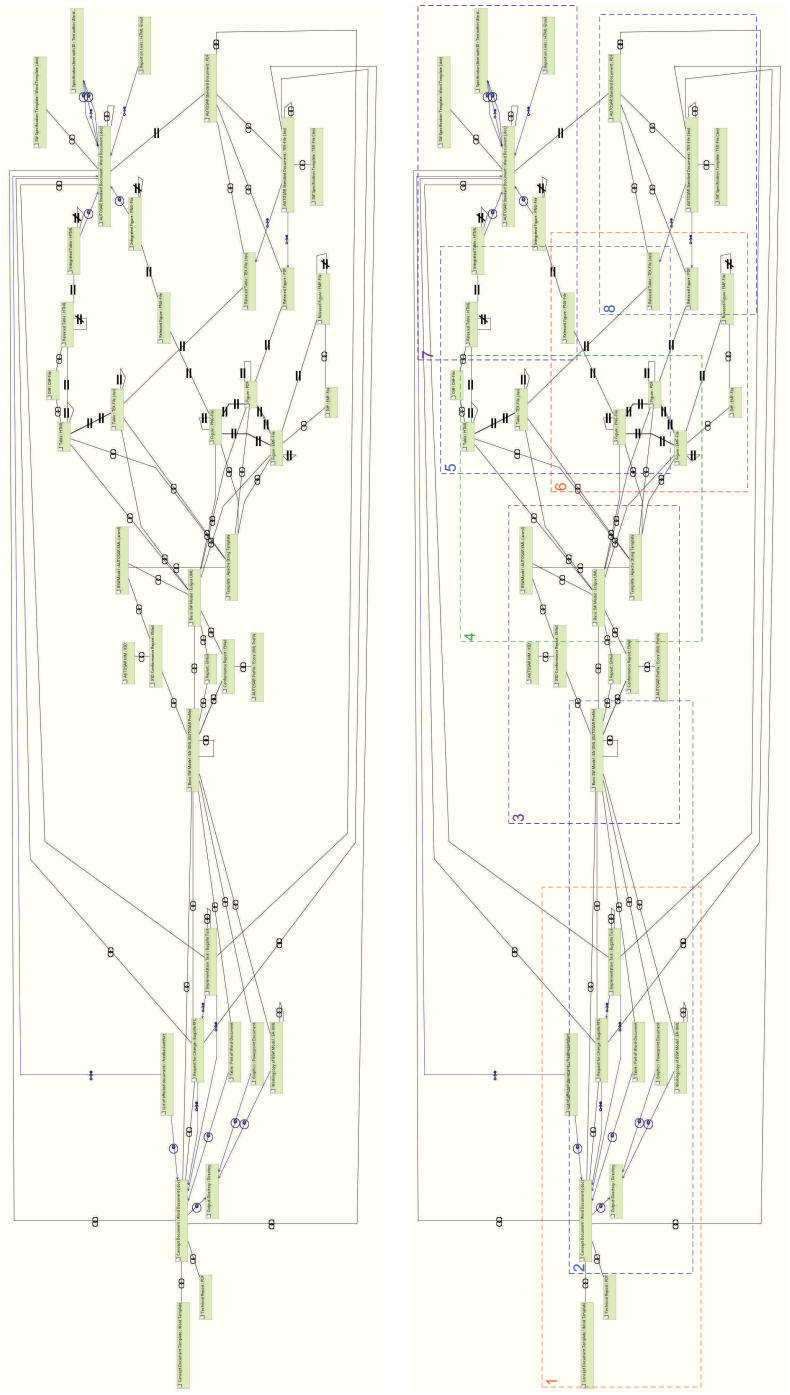

Figure A.10.: models\megamodel, overview - The overview should give an impression of the size and complexity of the megamodel. Please note that the details of the overview are not meant to be readable. Details can be looked up on the following enlargements

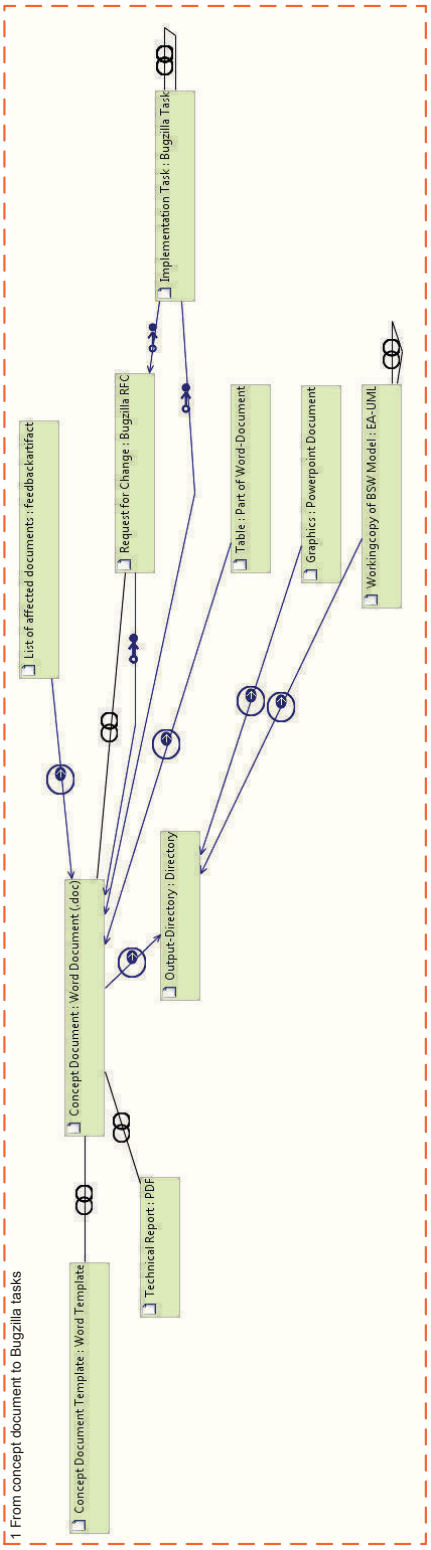

Figure A.11.: models\megamodel, part 1 of 8

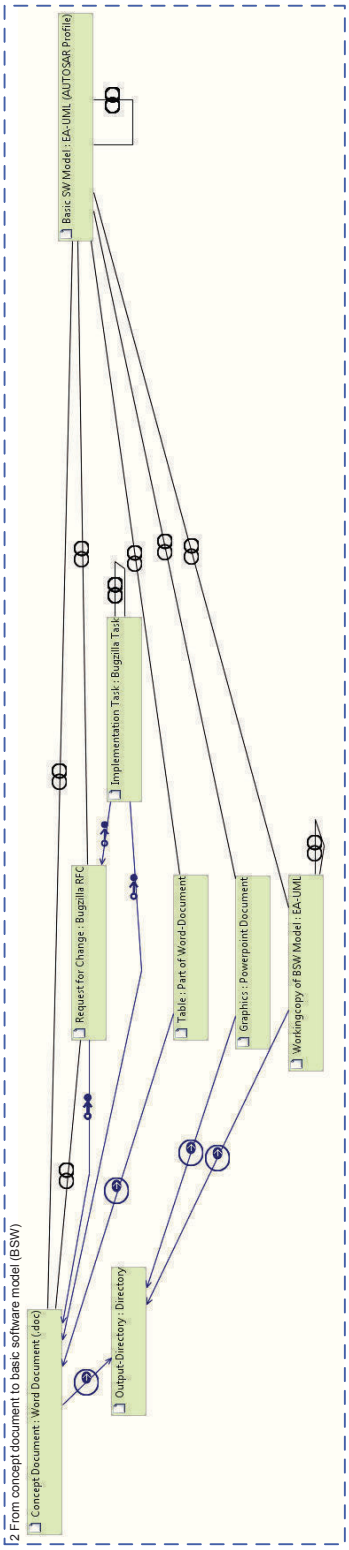

Figure A.12.: models\megamodel, part 2 of 8

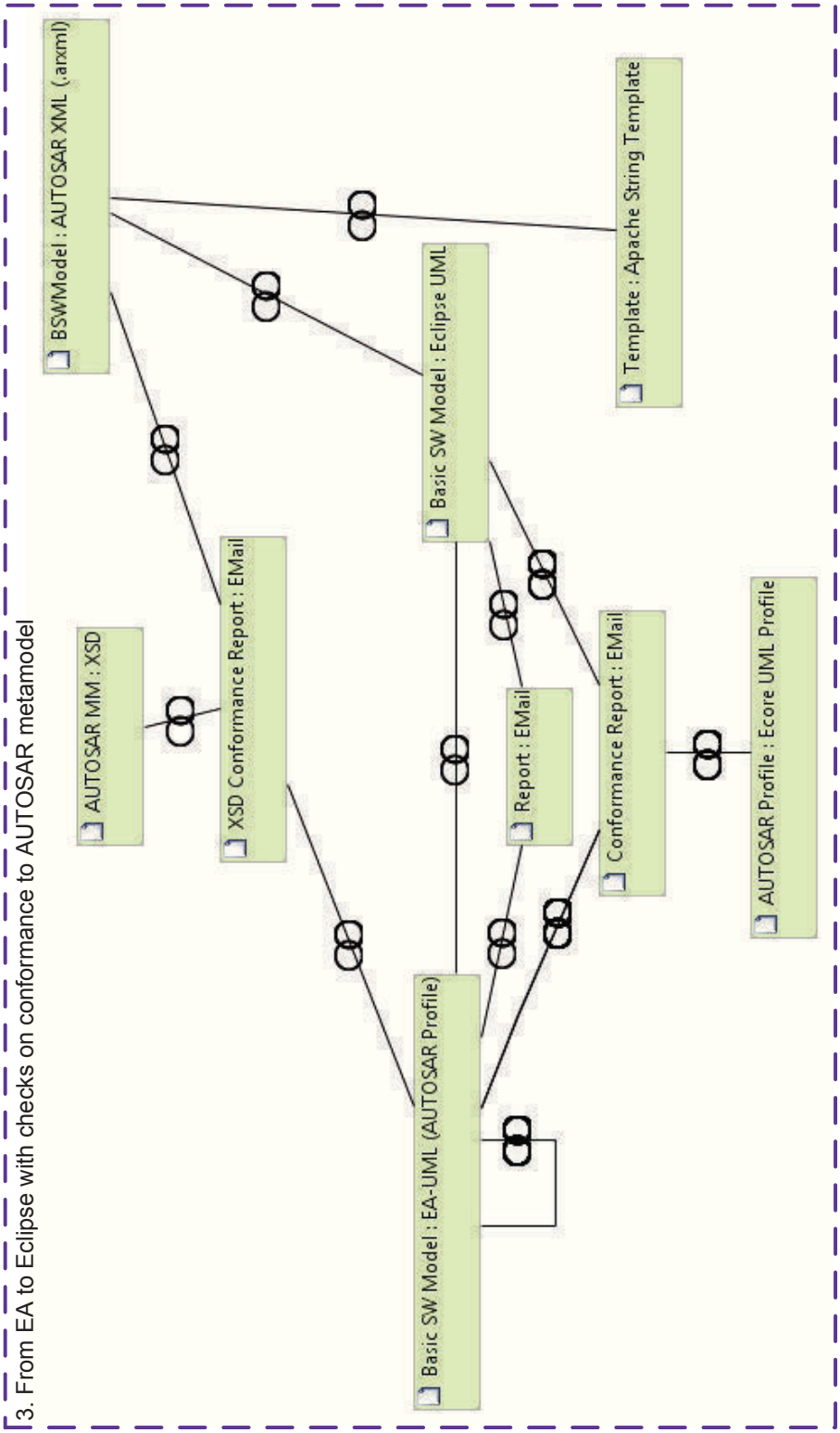

Figure A.13.: models\megamodel, part 3 of 8

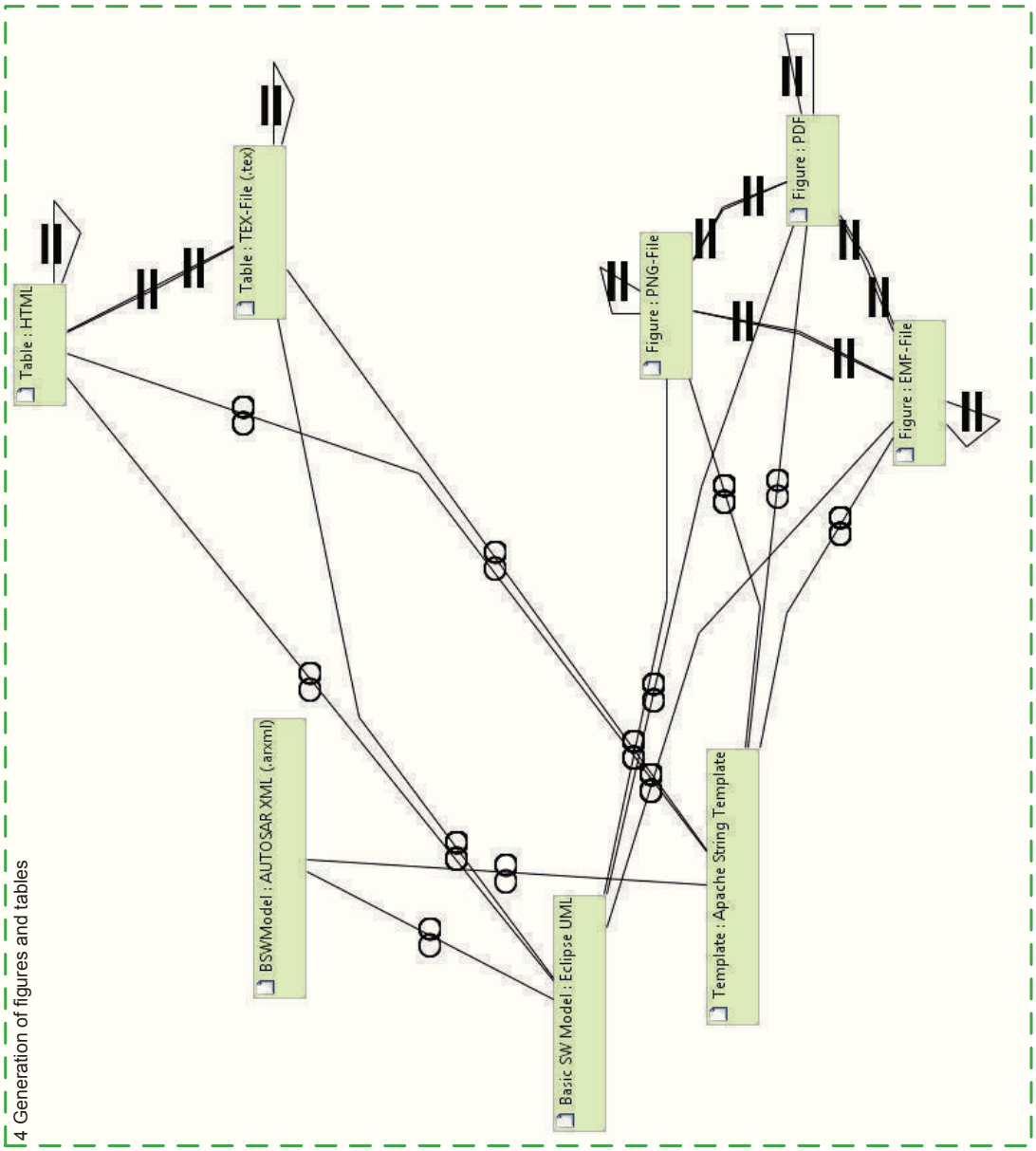

Figure A.14.: models\megamodel, part 4 of 8

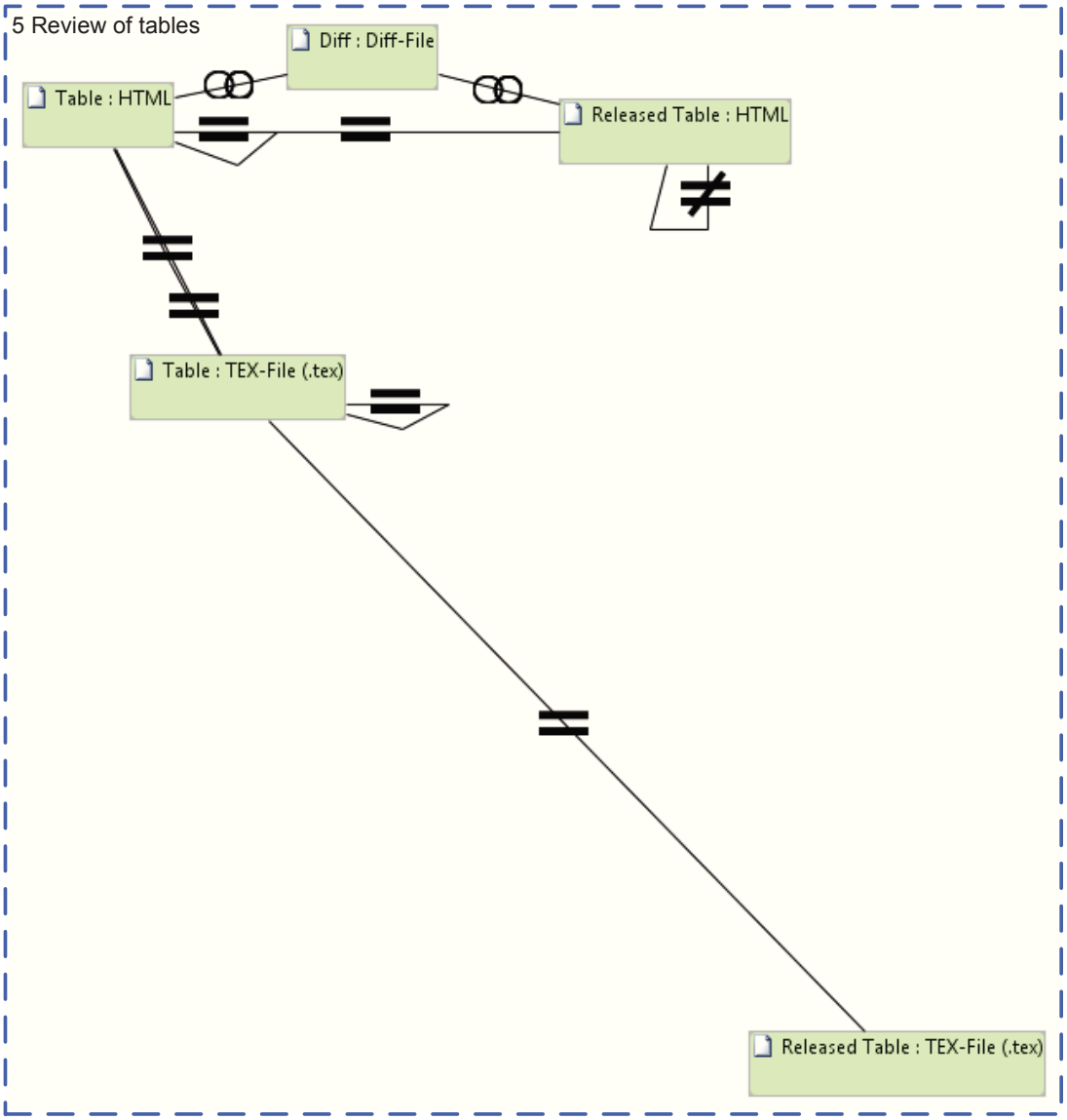

Figure A.15.: models\megamodel, part 5 of 8

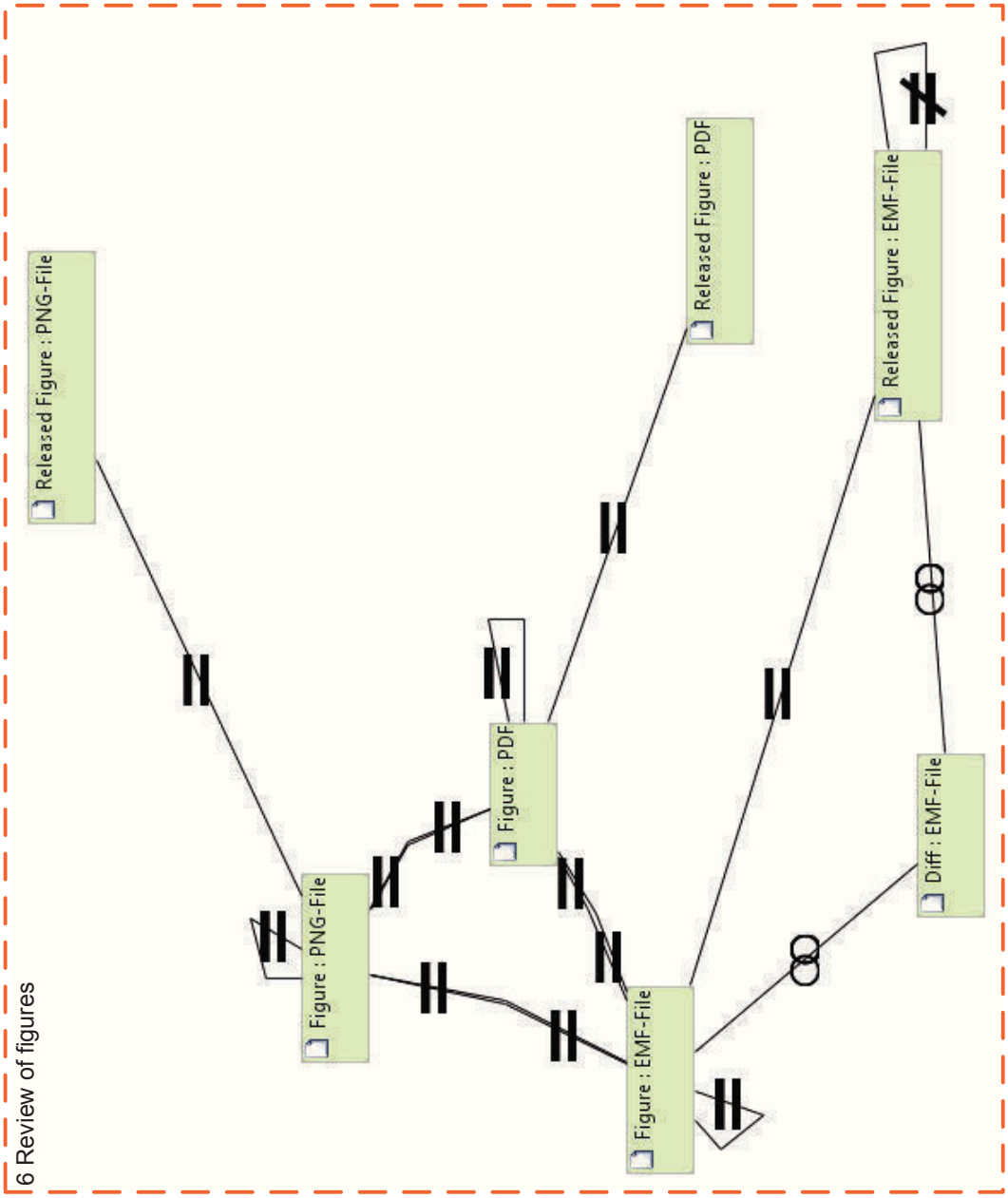

Figure A.16.: models\megamodel, part 6 of 8

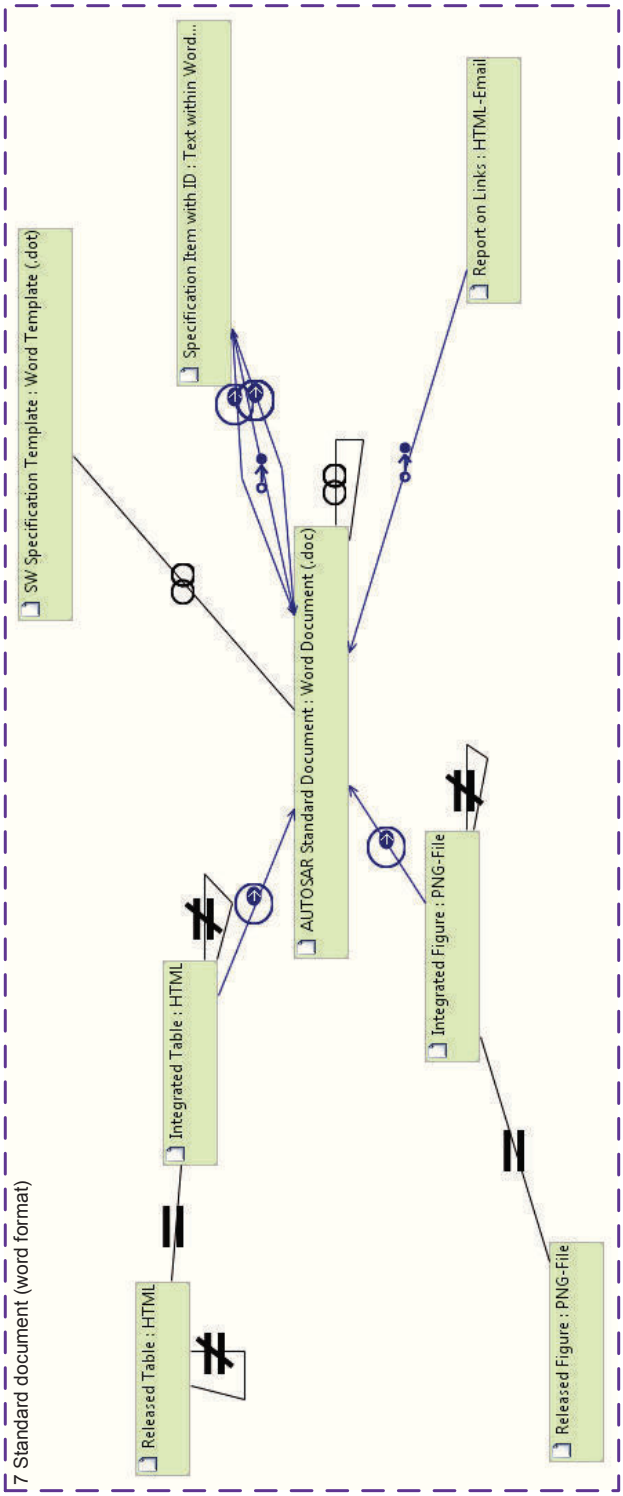

Figure A.17.: models\megamodel, part 7 of 8

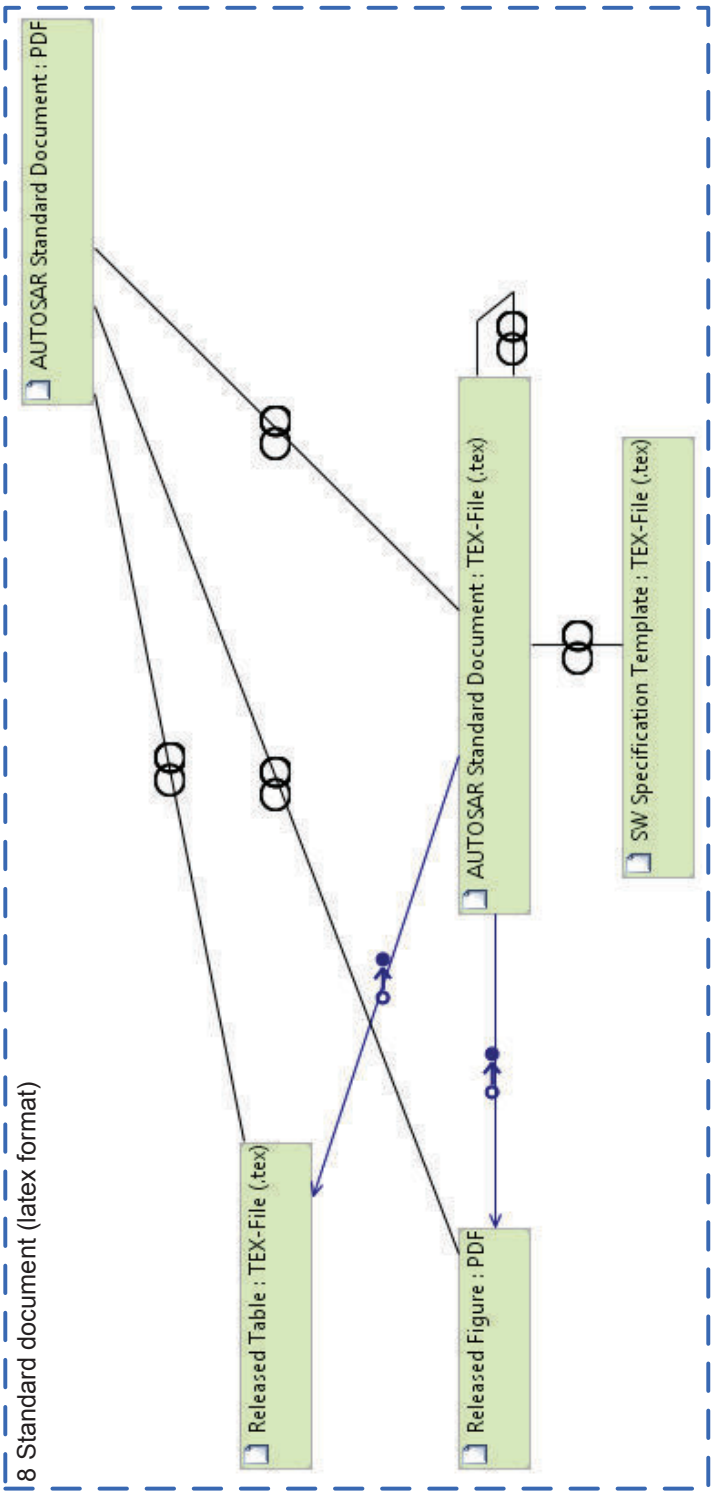

Figure A.18.: models\megamodel, part 8 of 8

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
91	978-3-86956-303-9	**Weak Conformance between Process Models and Synchronized Object Life Cycles**	Andreas Meyer, Mathias Weske
90	978-3-86956-296-4	**Embedded Operating System Projects**	Andreas Grapentin, Kirstin Heidler, Dimitri Korsch, Rakesh Kumar-Sah, Nicco Kunzmann, Johannes Henning, Toni Mattis, Patrick Rein, Eric Seckler, Björn Groneberg, Florian Zimmermann, Uwe Hentschel, Daniel Richter, Andreas Polze
89	978-3-86956-291-9	**openHPI : 哈索•普拉特纳研究院的 MOOC（大规模公开在线课）计划**	Christoph Meinel, Christian Willems
88	978-3-86956-282-7	**HPI Future SOC Lab : Proceedings 2013**	Christoph Meinel; Andreas Polze, Gerhard Oswald, Rolf Strotmann, Ulrich Seibold, Bernhard Schulzki (Hrsg.)
87	978-3-86956-281-0	**Cloud Security Mechanisms**	Christian Neuhaus, Andreas Polze (Hrsg.)
86	978-3-86956-280-3	**Batch Regions**	Luise Pufahl, Andreas Meyer, Mathias Weske
85	978-3-86956-276-6	**HPI Future SOC Lab: Proceedings 2012**	Christoph Meinel, Andreas Polze, Gerhard Oswald, Rolf Strotmann, Ulrich Seibold, Bernhard Schulzki (Hrsg.)
84	978-3-86956-274-2	**Anbieter von Cloud Speicherdiensten im Überblick**	Christoph Meinel, Maxim Schnjakin, Tobias Metzke, Markus Freitag
83	978-3-86956-273-5	**Proceedings of the 7th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Christoph Meinel, Hasso Plattner, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch (Hrsg.)
82	978-3-86956-266-7	**Extending a Java Virtual Machine to Dynamic Object-oriented Languages**	Tobias Pape, Arian Treffer, Robert Hirschfeld
82	978-3-86956-266-7	**Extending a Java Virtual Machine to Dynamic Object-oriented Languages**	Tobias Pape, Arian Treffer, Robert Hirschfeld
81	978-3-86956-265-0	**Babelsberg: Specifying and Solving Constraints on Object Behavior**	Tim Felgentreff, Alan Borning, Robert Hirschfeld

ISBN 978-3-86956-317-6
ISSN 1613-5652