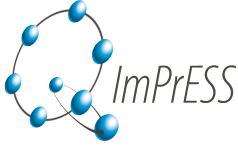


Project Deliverable D3.2

Prediction Models Generation Tools

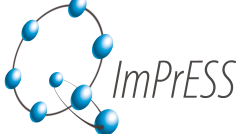
Project name:	Q-ImPrESS
Contract number:	FP7-215013
Project deliverable:	D3.2: Prediction Models Generation Tools
Author(s):	Steffen Becker, Mauro Luigi Drago, Antonio Filieri, Michael Hauck, Raffaella Mirandola, Johannes Stammel
Work package:	WP3
Work package leader:	PMI
Planned delivery date:	M20
Delivery date:	October 11, 2009
Last change:	October 11, 2009
Version number:	1.0

Abstract This document introduces and describes the prediction models generation tools which support the estimate of the quality attributes considered in the Q-ImPrESS project. The model-driven approach followed to derive performance and reliability models starting from SAMM metamodel is described. Furthermore, the document describes with some details the transformation from SAMM to simulation models to obtain performance predictions and the transformation from SAMM to DTMC to predict the system reliability. Finally, the tool suite for the prediction of maintainability property is illustrated.

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

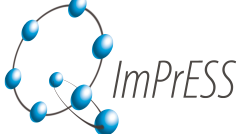
Revision history

Version	Change Date	Author(s)	Description
0.1	02-07-2009	PMI	Initial version of the document
0.2	15-07-2009	FZI	Added PCM sections
0.3	17-07-2009	PMI	Added MDD approach
0.4	27-07-2009	PMI	Added KLAPER sections
0.5	07-09-2009	PMI, FZI	Document revision
0.6	06-10-2009	PMI,FZI	Added Maintainability tool, Document revision
0.7	07-10-2009	PMI, FZI	Document revision
1.0	07-10-2009	PMI	Final Version

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

Contents

1	Introduction	4
2	Model Driven Development Approach	5
2.1	Performance prediction path	5
2.2	Reliability prediction path	5
3	SAMM-to-PCM transformation	7
4	PCM-to-simulation transformation	9
5	SAMM-to-DTMC transformation	11
5.1	SAMM to KLAPER	11
5.2	KLAPER to DTMC	12
6	Maintainability Prediction Tool	14
7	Conclusions	19

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

1 Introduction

This document represents Deliverable D3.2 of the Q-ImPRESS description of work.
This document is structured as follows:

Chapter 2: Chapter 2 describes the MDD approach adopted in the Q-ImPRESS project which supports the prediction of non-functional properties of high level software descriptions specified according to the Service Architecture Meta-Model.

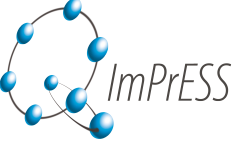
Chapter 3: Chapter 3 provides details about Service Architecture Meta-Model-to PCM transformations

Chapter 4: Chapter 4 provides details about PCM to simulation transformations

Chapter 5: Chapter 5 provides details about Service Architecture Meta-Model-to KLAPER and KLAPER-to Markov models transformations

Chapter 6: Chapter 6 provides details about the maintainability approach and tool in Q-ImPRESS project.

Chapter 7: Chapter 7 concludes the presentation of the Q-ImPRESS tools and provides an outlook to future work and extensions.

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

2 Model Driven Development Approach

In this Chapter we shortly describe the model-driven approach we have followed to derive performance and reliability models starting from SAMM metamodel.

2.1 Performance prediction path

To perform performance predictions for software systems that are being modelled by means of the Q-ImPrESS Service Architecture Meta-Model (SAMM), such a model instance has to be transformed into a different model that can be used for performance analysis. In this context, we focus on performance predictions with the Palladio Component Model (PCM).

To conduct performance analysis with the PCM, the SAMM instance has to be transformed into a PCM instance first. The PCM instance then is being transformed into a model which can be used for performance analysis. In the following, the PCM SimuCom framework is used, which is the PCM's tool for performing simulations to analyze the performance of software systems. However, the PCM also allows for transforming PCM instances into other analysis models, such as Layered Queueing Networks or Queueing Petri Nets.

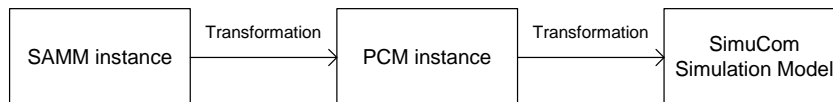


Figure 2.1: The Performance Transformation Chain

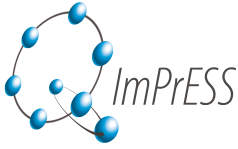
The transformation chain for achieving performance predictions is shown in Figure 2.1. First, a SAMM instance is being transformed into a PCM instance by means of the QVT Operational transformation language (QVT-O), whose execution is supported by the Eclipse M2M project¹. Based on this PCM instance, a SimuCom simulation model is being created by transforming the PCM instance into Java code that integrates into the SimuCom framework. For this transformation, transformation languages of the openArchitectureWare (oAW) framework² are used.

2.2 Reliability prediction path

Reliability analysis is based on Discrete Time Markov Chain (DTMC) models. DTMC models provide probabilistic information concerning system behavior evolution and their possible outcomes in terms of correct or incorrect execution. These models can be analyzed in many different

¹<http://www.eclipse.org/m2m/>

²<http://www.openarchitectureware.org>

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

ways, from model-checking to simulation, in order to predict the system's reliability.

In Q-ImPRESS, reliability prediction is obtained by a two steps, waterfall-style, transformation chain which allows the PRISM Model Checker³ to provide the requested analysis' results. (see Figure 2.2).

The first step extracts information from the SAMM instance to be analyzed. In particular the transformation requires the SAM model (thus its static structure), the components' repository, a behaviour description (so far it must be provided in SEFF) and a QoS annotations repository. The KLAPER model is produced by means of a QVT-O transformation. The output model summarizes SAM information by providing:

- a KLAPER Resource for each SAM component.
- a KLAPER Service for each SAM operation, which will be held at the resource representing the provider component.

Such an approach allows the representation of each operation in KLAPER for further analysis. Notice that a service is realized by a component instance and that each internal action is represented by a KLAPER step whose failure probability value is kept and remain identifiable after the transformation, hence very fine-grained information is maintained.

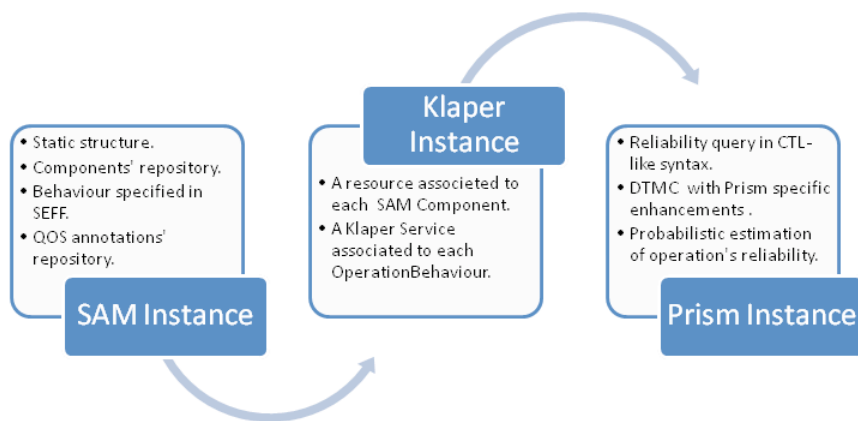
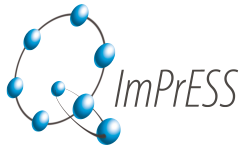


Figure 2.2: The Reliability Transformation Chain

The second step gets the KLAPER model to be analyzed and produces a DTMC model. Due to the fact that this model will be analyzed by means of the PRISM Model Checker, some PRISM specific constructs were adopted (such as Modules) in order to keep the analysis more efficient and the output models more readable. Reliability analysis of the produced PRISM model is driven by a reliability query which allows the user to specify which part of the model has to be analyzed.

³<http://www.prismmodelchecker.org/>



3 SAMM-to-PCM transformation

To transform a SAMM instance into a PCM instance, the Q-ImPrESS tool landscape has been extended by a transformation script that provides a transformation from a SAMM instance into a PCM instance. This transformation is written in QVT Operational (QVT-O), a standardized transformation language. For QVT-O, sophisticated tools are available that integrate into the Eclipse platform which is also used for the other Q-ImPrESS tools.

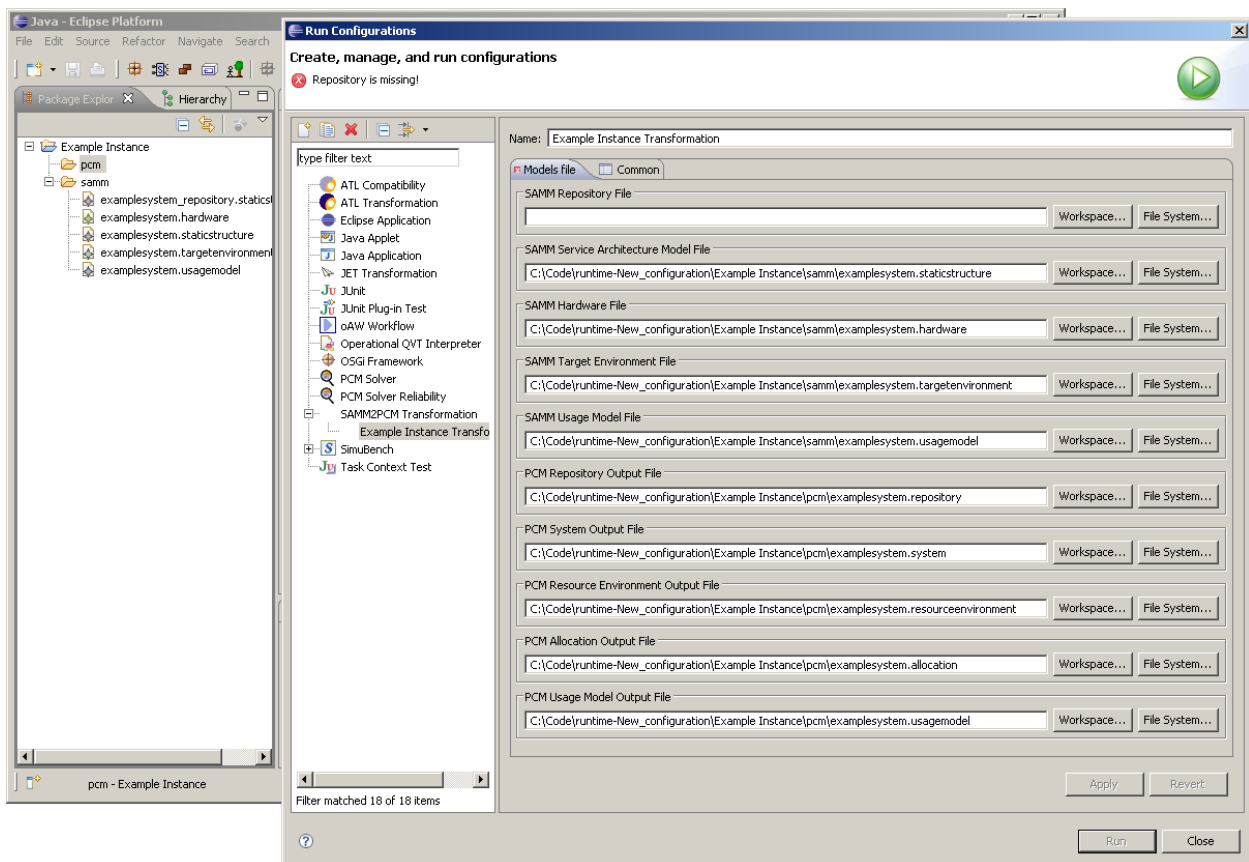


Figure 3.1: Specifying a SAMM to PCM transformation run

A prototypical dialog to specify a transformation from a SAMM instance to a PCM instance is shown in Fig. 3.1. The dialog asks the user to specify the SAMM input files as well as the name of the PCM output files. If a input file is not correctly specified, a warning is issued. After specifying all relevant transformation parameters, the transform is performed. In our example, different PCM output files are being created in the “pcm” output folder and can be displayed by PCM editors (see Fig. 3.2).

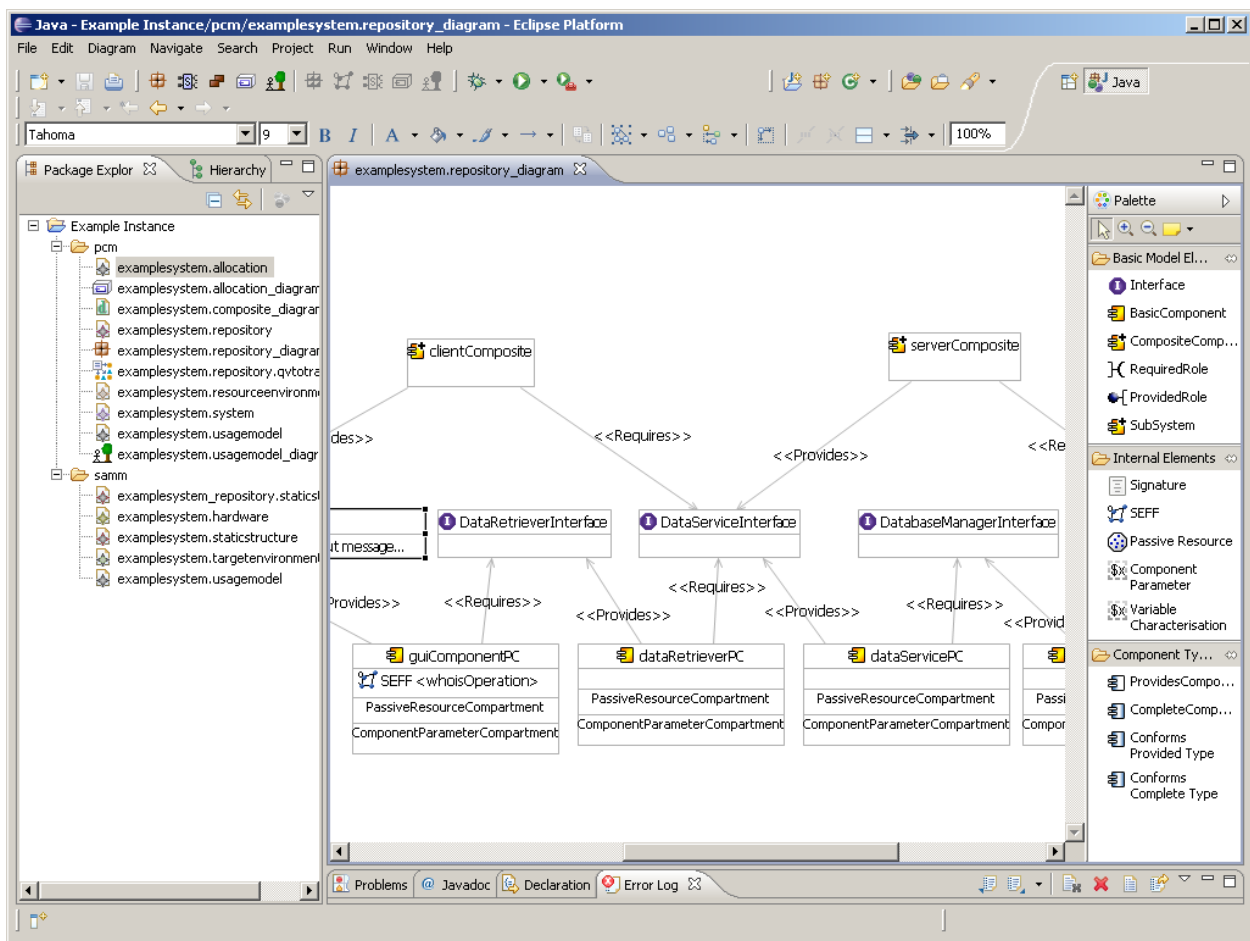


Figure 3.2: The output PCM model of the SAMP to PCM transformation

4 PCM-to-simulation transformation

Based on the output of the SAMM to PCM transformation, the resulting PCM instance can now be used for performance analysis.

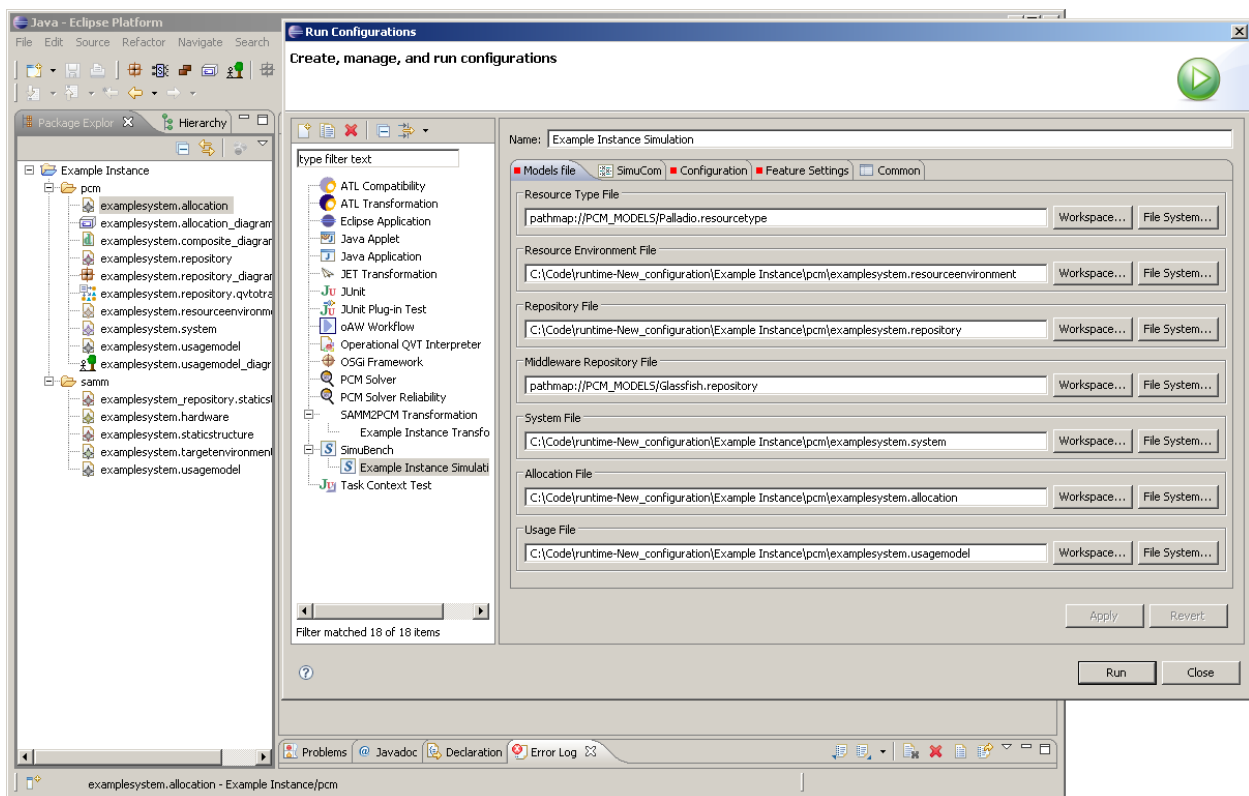


Figure 4.1: Specifying a PCM simulation run

To analyse the performance of the output system, the PCM simulation can be used. Fig. 4.1 shows the dialog to specify a PCM simulation run. As input parameters for the run, the files of the PCM instance are being specified.

After specifying all necessary information for a simulation run, the simulation starts. Fig. 4.2 shows the prototype IDE with a running PCM simulation. Note that the project “de.uka.ipd.sdq.codegen.simucominstance” shown in the package explorer on the right denotes the project that contains the automatically generated code needed for executing the simulation of the specified PCM instance.

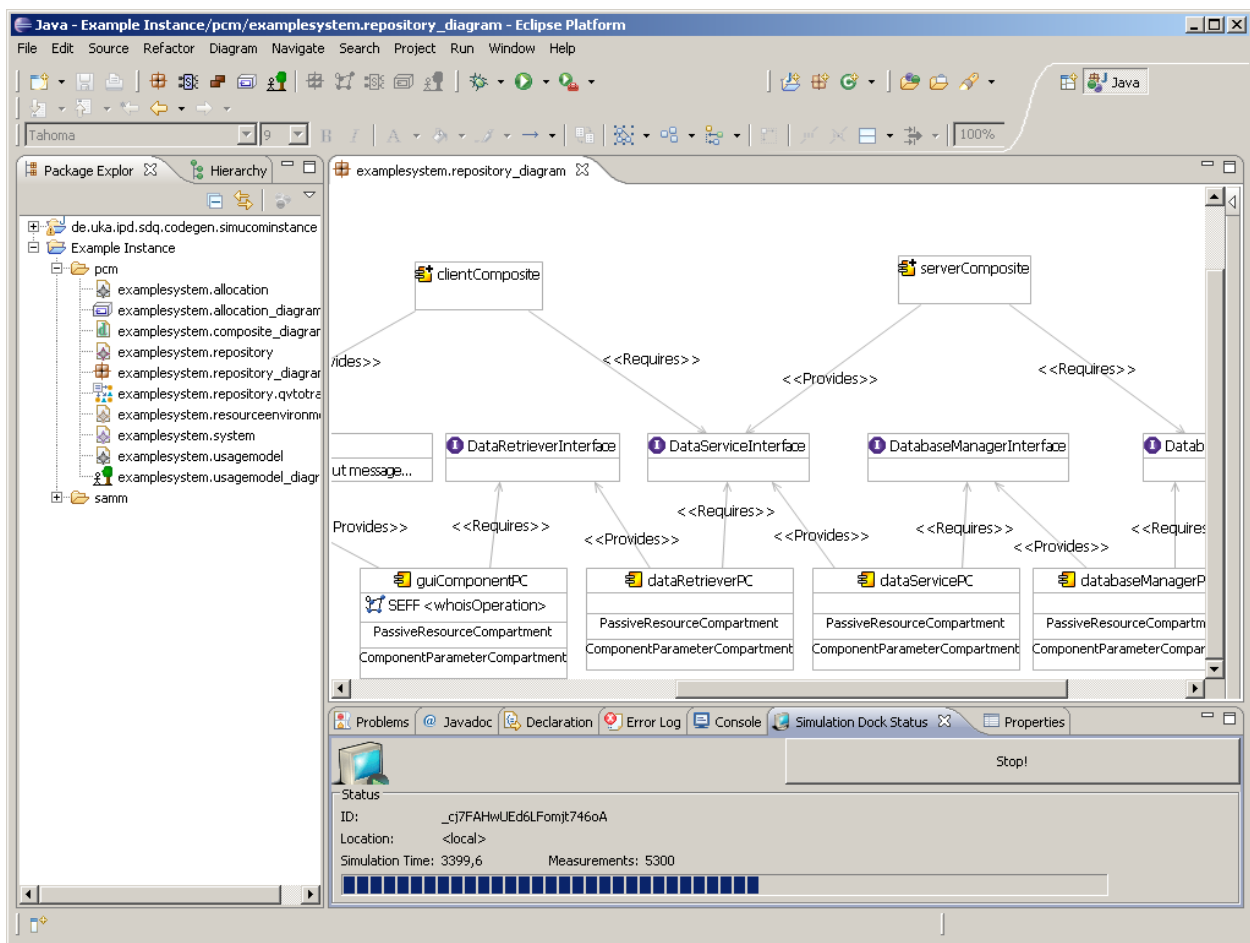
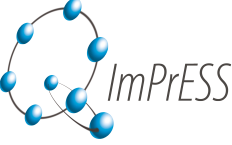


Figure 4.2: A running PCM simulation

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

5 SAMM-to-DTMC transformation

In order to accomplish a reliability analysis of required operations, a suitable DTMC model has to be obtained from the original SAM model.

The transformation is split in two main steps: SAMM to KLAPER and then KLAPER to DTMC. Some details about these transformations are provided in the next subsections.

The transformation setup is quite simple: the user has to point out the alternatives to be analyzed among the ones in the Q-ImPRESS workspace, as shown in figure 5.1:

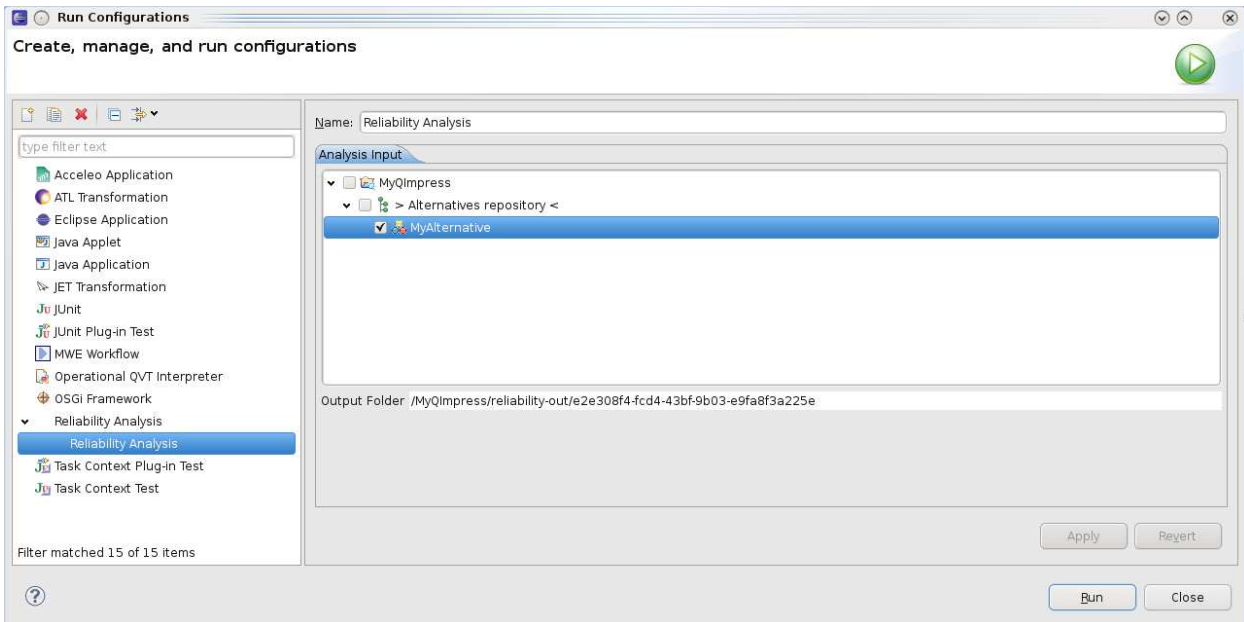


Figure 5.1: Specifying a reliability analysis run.

5.1 SAMM to KLAPER

SAMM to KLAPER transformation is defined via a QVT Operational script. As it was said before, many tools are supported by the eclipse platform to deal with QVT-O transformation.

A standard form to configure QVT-O scripts execution is shown in figure 5.2. The user has just to specify input and output files, and then run the execution engine.

Because of the integration in the Eclipse Modeling Framework, input checking is available in terms of metamodel conformance check. The output model is fully compliant with the KLAPER metamodel.

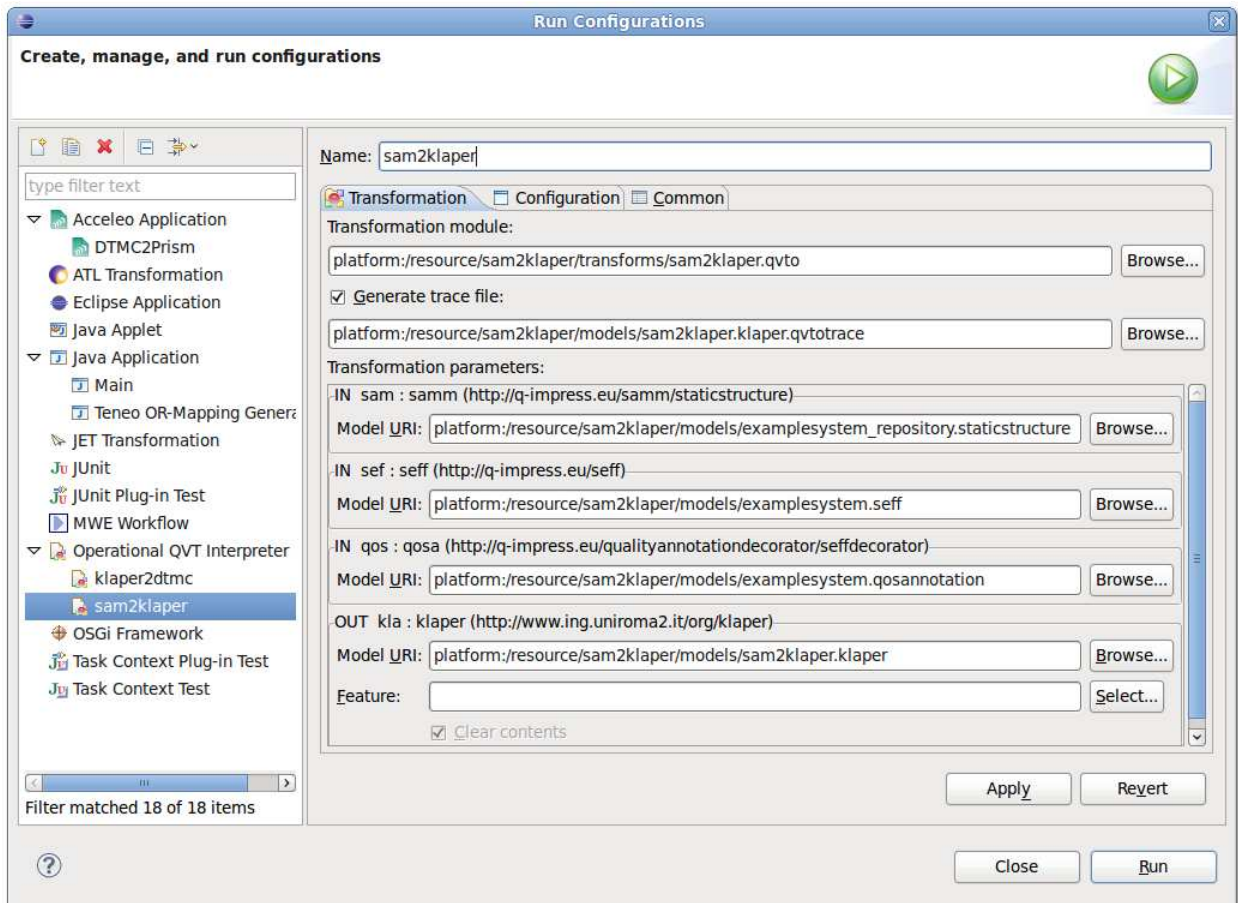
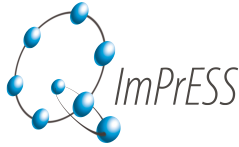


Figure 5.2: Configuring a SAMM to KLAPER transformation run

5.2 KLAPER to DTMC

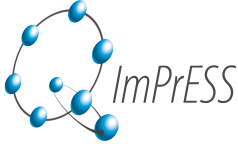
Based on the output of the SAMM to KLAPER transformation, the resulting KLAPER instance can now be used for reliability analysis.

Reliability analysis is conducted by producing DTMC models to be analyzed by the PRISM model-checker.

KLAPER to DTMC is implemented by means of QVT-O. It shares the same advantages already discussed for the SAMM to KLAPER transformation, namely it adopts the current standard from OMG, it keeps traceability data, it is imperative and, due to EMF-M2M engine, provides also incremental model transformation.

DTMC to PRISM source code is implemented via Acceleo. Acceleo¹ is a model-to-text transformation engine that natively uses EMF environment. It is currently supported by the Eclipse

¹<http://www.acceleo.org>

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

Modeling Project and is available as an Eclipse 3.5 plugin. It allows the definition of modular, easily maintainable transformations.

The products of DTMC to PRISM are a PRISM model and a Query set which asks the model-checker to look for reliability of operations currently used by the system user (as defined in the usagemodel).

The result of the analysis is a the success probability of the required operation, that is, the *a priori* expectation for the operation to be correctly completed. Such a result is available in a textual report.

6 Maintainability Prediction Tool

In this section we describe the tool development of the maintainability prediction tool suite. As we already showed in other deliverables (see Deliverable D3.1), maintainability prediction is a more human-oriented task than prediction of performance or reliability where a formal prediction model exists. Maintainability Prediction is done by describing change requests in an architecture model and estimation of change efforts. For a given pair of architecture alternative and change request the user derives a work plan which contains activities describing the steps of implementing the changes. The work plan is derived in a semi-automatic way. The tool-suite guides the user during identification of work activities.

Preparation Phase The tool provides an user input for selection of architecture alternatives and selection of change requests. As shown in Figure 6.1 the tooling provides input forms for entering descriptions of architecture alternatives and change requests. Architecture models which are present in Q-ImPRESS Backbone can be loaded here and associated with architecture alternatives intended to be analysed in maintainability prediction.

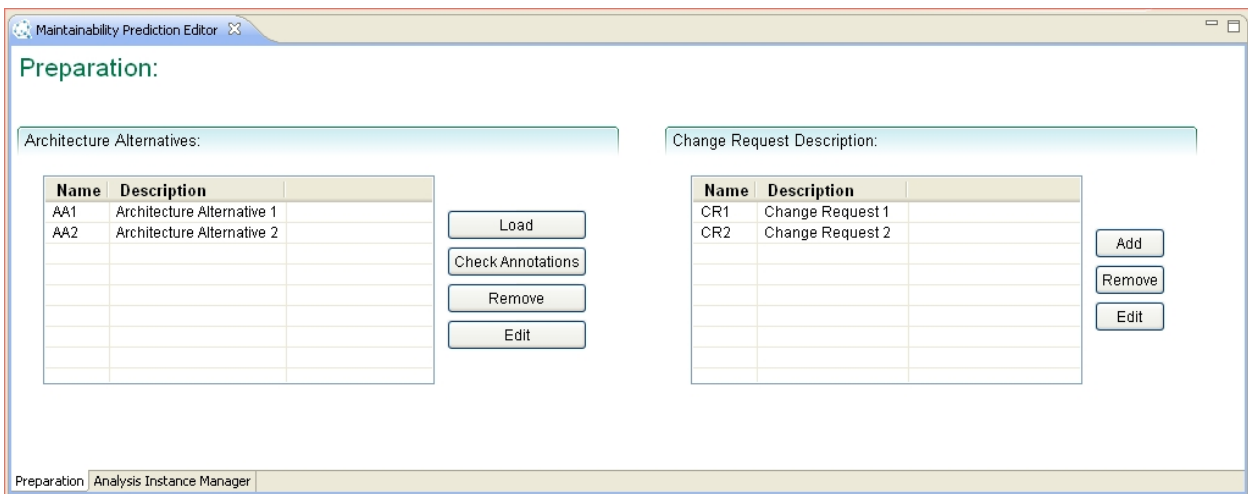
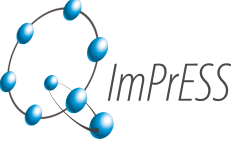


Figure 6.1: Preparation Phase

Maintainability Analysis Phase After the preparation phase is done the tool allows proceeding with an overview table for management of maintainability analysis instances, see Figure 6.2. Each analysis instance is a pair of an architecture alternative and a change request. Each row in the overview table represents one analysis pair. For each analysis pair the user can trigger several analysis steps by pushing respective buttons in the table. The following paragraphs give a short overview of the analysis steps.

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

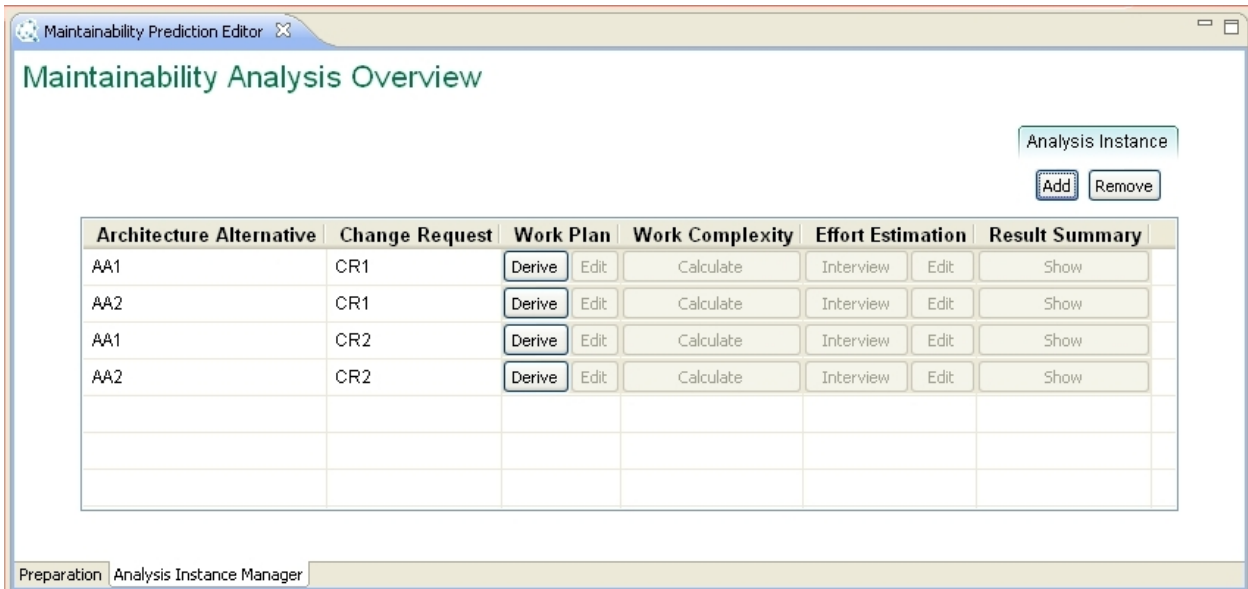


Figure 6.2: Analysis Instance Management

Workplan Derivation The first step for each analysis instance is to derive a workplan which describes the work activities for implementing the change request. These activities are composed of a basic activity (ADD, CHANGE, REMOVE) and an architecture element (i.e. component, interface, operation, etc.). The tool suite provides a wizard mechanism to systematically derive work plans and also for refinement of already existing work plans. In Figure 6.3 a part of the derivation wizard is illustrated. In the shown wizard page the user can select affected components from a complete list of components in the system and also determine basic activities for selected components. Similar wizard pages are provided for work activities regarding other elements (e. g. interfaces, operations, data types, etc.)

Workplan Editing After derivation of a first work plan, the user can use editing capabilities to systematically refine and extend work plans. Figure 6.4 shows a draft of the editing page which will be provided by the tool. Work activities are presented in a tree structure in order to reflect the hierarchical structure of work activities.

Work Complexity Calculation By using complexity annotations in the architecture model the user can trigger a process which calculates complexity metric values regarding work activities present in work plan. Calculated metric values are presented below the work plan table as shown in Figure 6.5.

Collect Effort Estimations When the change request for an analysis instance is fully reflecting in the work plan the user triggers a collection process for effort estimates. The user can choose

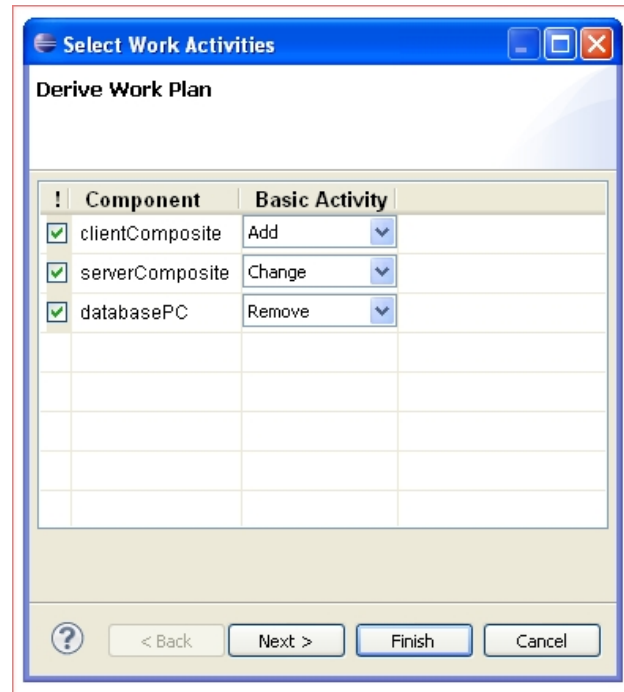


Figure 6.3: Workplan Derivation

between an interview mode and a direct editing mode. In the *interview mode* the user will follow a wizard which asks for time effort estimates for work activities on the lowest abstraction level. The user can also select *direct editing mode* which leads to a slightly modified version of the work plan editor as illustrated in Figure 6.5.

Show Result Summary In order to present a result summary the tool aggregates effort estimates and presents them in a results overview page.

Note that tool development for maintainability prediction is still work in progress. The user interface in the final version of the tool might slightly differ from the examples shown in this document.

Work Plan Editor

Order Activities by

- Containment Relation
- Follow-Up Relation

Activity Structure	Architecture Element	Basic Activity
+... Component Impl.	Component C1	Change
+... Interface Impl.	Interface Port IP3	Change
+... Operation Impl.	Operation Impl. OP3	Add

Add Directly Known Activities Refine Abstraction Level

Figure 6.4: Workplan Editing

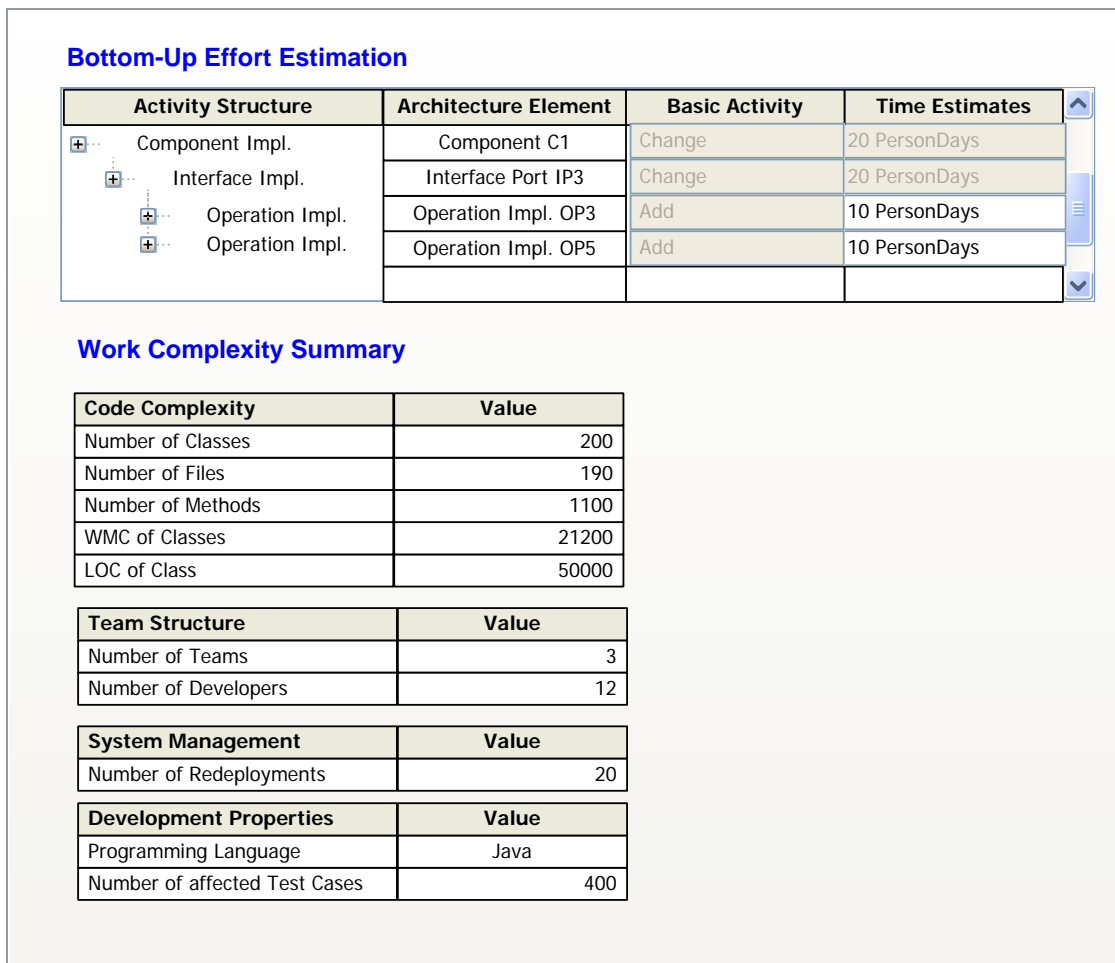
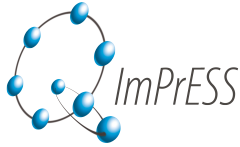
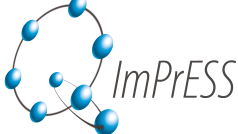


Figure 6.5: Effort Estimation and Complexity Metrics

	D3.2: Prediction Models Generation Tools	
	Version: 1.0	Last change: October 11, 2009

7 Conclusions

This document introduces and describes the prediction models generation tools which support the estimate of the quality attributes considered in the Q-ImPRESS project. The model-driven approach followed in the derivation of performance and reliability models starting from SAMM metamodel has been described and some details are given about the transformations from SAMM to simulation models for performance predictions and from SAMM to DTMC for system reliability. Finally, the document illustrates the tool suite for the prediction of maintainability property based on the Q-ImPRESS approach for maintainability described in Deliverable D3.1.

Future work will focus on the integration of the transformation tools in the overall Q-ImPRESS tool chain and in the validation of the quality prediction results with QoS metrics provided by the industrial case studies.