

Project Deliverable D9.7


Publishable Project Summary Report

| | |
|-------------------------------|---|
| Project name: | Q-ImPrESS |
| Contract number: | FP7-215013 |
| Project deliverable: | D9.7: Publishable Project Summary Report |
| Author(s): | Ivica Crnkovic, Juan Carlos Flores Beltran, Michael Hauck, Jan Kofron, Heiko Koziolk, Klaus Krogmann, Marco Masetti, Raffaella Mirandola, Cristina Seceleanu, Marijan Zemljic |
| Work package: | WP9 |
| Work package leader: | FZI |
| Planned delivery date: | M36 |
| Delivery date: | 31.01.2011 |
| Last change: | 31.01.2011 |
| Version number: | 1.0 |

Abstract

This final report summarises the results of the Q-ImPrESS project. It presents the developed method, supporting tooling, the evaluation, and accompanying documentation of the project. This document briefly summarises the challenges, benefits and essential contributions of the project. The last chapters contain an analysis of potential impacts and outcomes of the project results, lessons learned, and future activities already foreseen.

Keywords: Final project report, public summary

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

Revision history

| Version | Change date | Author(s) | Description |
|---------|-------------|-------------|--|
| 0.1 | 10.01.2010 | K. Krogmann | Created initial document skeleton |
| 0.2 | 31.01.2010 | M. Hauck | Integrated input parts of all partners |
| 1.0 | 31.01.2010 | M. Hauck | Final version |
| | | | |
| | | | |




| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

Table of contents

| | | |
|-----------|--|-----------|
| 1 | The challenge | 6 |
| 2 | Addressing the challenge: the project's proposition..... | 7 |
| 3 | Who can benefit from Q-ImPrESS | 9 |
| 4 | Highlights of achievements | 10 |
| 5 | The results | 11 |
| 5.1 | <i>Analysis of multiple quality dimensions.....</i> | <i>11</i> |
| 5.2 | <i>Trade-off decisions and support of design alternatives</i> | <i>12</i> |
| 5.3 | <i>Integrated tooling / reverse engineering.....</i> | <i>14</i> |
| 5.3.1 | Integrated tool support of design alternatives | 14 |
| 5.3.2 | Central model infrastructure..... | 14 |
| 5.3.3 | Editors and viewers | 15 |
| 5.3.4 | Quality prediction and trade-off analysis | 15 |
| 5.3.5 | Reverse engineering..... | 15 |
| 5.3.6 | Consistency checking..... | 16 |
| 5.4 | <i>The service architecture model and extensible analysis workflows.....</i> | <i>16</i> |
| 5.4.1 | SAMM (Service Architecture Meta-Model) | 16 |
| 5.4.2 | Extensible workflow | 19 |
| 5.5 | <i>Resource models</i> | <i>22</i> |
| 5.5.1 | Resource sharing | 22 |
| 5.5.2 | Sharing investigation..... | 22 |
| 5.5.3 | Validation methodology..... | 23 |
| 5.5.4 | Resource models | 24 |
| 6 | The pilots | 26 |
| 6.1 | <i>ENT demonstrator</i> | <i>26</i> |
| 6.2 | <i>ABB demonstrator</i> | <i>27</i> |
| 6.3 | <i>The eSOA showcase.....</i> | <i>28</i> |
| 7 | Availability of results | 30 |
| 8 | Potential impact of the results | 31 |
| 8.1 | <i>The main dissemination activities.....</i> | <i>31</i> |
| 8.2 | <i>Exploitation of the results.....</i> | <i>32</i> |
| 9 | Lessons learned during the project..... | 34 |
| 9.1 | <i>Applicability of Q-ImPrESS.....</i> | <i>34</i> |
| 9.2 | <i>Lessons learned</i> | <i>35</i> |
| 9.3 | <i>Future work</i> | <i>36</i> |
| 10 | Partners | 37 |
| | <i>Forschungszentrum Informatik (FZI) an der Universität Karlsruhe.....</i> | <i>37</i> |

| | | |
|---|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

| | |
|---|-----------|
| <i>ABB AG Forschungszentrum Deutschland (ABB)</i> | 37 |
| <i>Mälardalen University (MDU)</i> | 38 |
| <i>Politecnico di Milano (PMI)</i> | 38 |
| <i>Univerzita Karlova v Praze (CUNI)</i> | 38 |
| <i>Itemis GmbH & Co. KG (ITE)</i> | 38 |
| <i>Softeco Sismat S.p.A. (SFT)</i> | 38 |
| <i>Ericsson Nikola Tesla d.d. (ENT)</i> | 38 |
| 11 Contacts | 39 |

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

Executive summary

The Q-ImPRESS project brings the advantages of service orientation to highly relevant application domains such as industrial production control, telecommunication and critical enterprise applications, where guaranteed end-to-end quality of service is particularly important.

The methods and tools developed in Q-ImPRESS allow for cost-effective development and evolution of service-oriented software. Developers are able to try out different design alternatives and choose the best possible alternative with respect to the impact of these decisions on the quality of the software, before ever writing any single line of code.

The developed Q-ImPRESS method allows a balanced software design. Q-ImPRESS allows balancing the trade-off between the external quality of service properties in terms of performance and reliability and the internal software quality in terms of maintainability.


Choosing the right design from the beginning and thus avoiding later restructuring, tweaking and project delays to get certain quality attributes such as performance under control showed to be suitable to save significant effort in large software projects. Industrial demonstrators of the production control systems, telecommunication, and enterprise SOA domain proved the applicability of the developed method and tools.

Q-ImPRESS defines a new service design model of a software system augmented with information about service encapsulation and deployment, called the service architecture model (SAM). Such SAMs can be created for newly designed services or extracted from existing code. The SAM is designed to be fully transformable, supporting the simulation of architectural changes such as redistributing functionality between services, adding functionality to a service or service re-composition without changing the existing implementation.

Starting from this service architecture model and using model-driven techniques, separate prediction models for each considered quality dimension (performance, reliability, and maintainability) can be derived and used to predict the quality attributes for a given service architecture design. The integrated quality impact and trade-off analysis provides the software engineer with a complete picture of the consequences of his design changes and enable him to experiment with different service design alternatives and select the best one for his system.

Q-ImPRESS represents a unique and outstanding approach which combines the quality of service properties performance, reliability, and maintainability in a single method, model and tooling and integrates analyses for all of these quality dimensions.

The Q-ImPRESS method is accompanied by extensive general and domain-specific guidelines for the establishment of high end-to-end quality of service, tooling documentation, automated checks of service compositions, detailed execution environment resource (which affect the quality of services) analyses, and an explaining eSOA showcase. The method and tools are all open source, publicly available, and successfully used by third-party research projects.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

1 The challenge

Today, software is an essential part of many products of our daily life. Likewise, software supports in an essential way many critical systems in various industries, ranging from enterprise information systems to production control or telecommunication infrastructure. Software failure of such systems can cause severe damage to the enterprise, including its closure. Besides such dramatic consequences of a failure, critical systems also have in common that (i) their quality of service properties, including performance, scalability, reliability, etc., are of high importance, and (ii) they often evolve and exist over decades. All these properties pose specific challenges to software engineering techniques. In spite of these challenges, however, there is a strong motivation to broaden the application of software in critical systems: the ability of software to adapt existing and include new functionality makes broadened software use mandatory in a competition driven economy. In particular, the design and operation principle of “service orientation” gained enormous attention for its promises of improving a software system’s adaptability, extendibility and re-configurability. However, current practice clearly shows that most critical processes are not designed in a service oriented way. This is due to severe conceptual shortcomings of service oriented standards, which do not support the analysis of software systems with respect to their performance, scalability and reliability. Without such analytical abilities, one cannot provide services with guaranteed end-to-end quality. Even worse, during the evolution it is hard to foresee the consequences of software changes on its overall quality of service. As the lack of this kind of analyses makes the evolution of critical software systems impossible, the benefits of service orientation are currently abandoned in critical system development and operation. The fact that many major innovative industries, including production systems, banking, insurances, eGovernment, eHealth and telecommunication, are not fully exploiting the advantages of service-orientation is the main motivation of the proposed project.

The challenge is to enable service orientation for critical systems. The impact of design decisions on quality of service properties needs to become predictable such that one can engineer critical system right from the beginning and avoid unnecessary and costly trial-and-error-cycles.

2 Addressing the challenge: the project's proposition

The main idea of the project is to enable service orientation for critical systems by creating a tool-supported method to perform what-if analyses during software evolution. Through this method, system developers, administrators and maintainers are able to foresee the impact of software design decisions and maintenance actions on quality of service and maintainability in advance. In our method, users and developers tightly interact to create and evolve service-oriented software with predictable end-to-end quality. Important properties of the method are: tight integration of design, usage and evolution, predictable quality of services and service compositions and integration of legacy code.

The method supports quality-driven software development, where the consequences of design decisions and system resource changes are made explicit to the user through quality impact analysis and simulation.

The vision of the proposed project is to enable dynamic service oriented architectures with guaranteed end-to-end quality using analysis and simulation techniques for quality impact analysis. This allows to understand the consequences of changes in the context of evolving service oriented systems. Design alternatives with possibly varying and conflicting quality of service properties can be interactively compared in a trade-off analysis. Only the best evolution alternative is then selected and actually implemented. Figure 1 presents a visual representation of the Q-ImPRESS approach which has been realised in the Q-ImPRESS IDE (Integrated Development Environment).

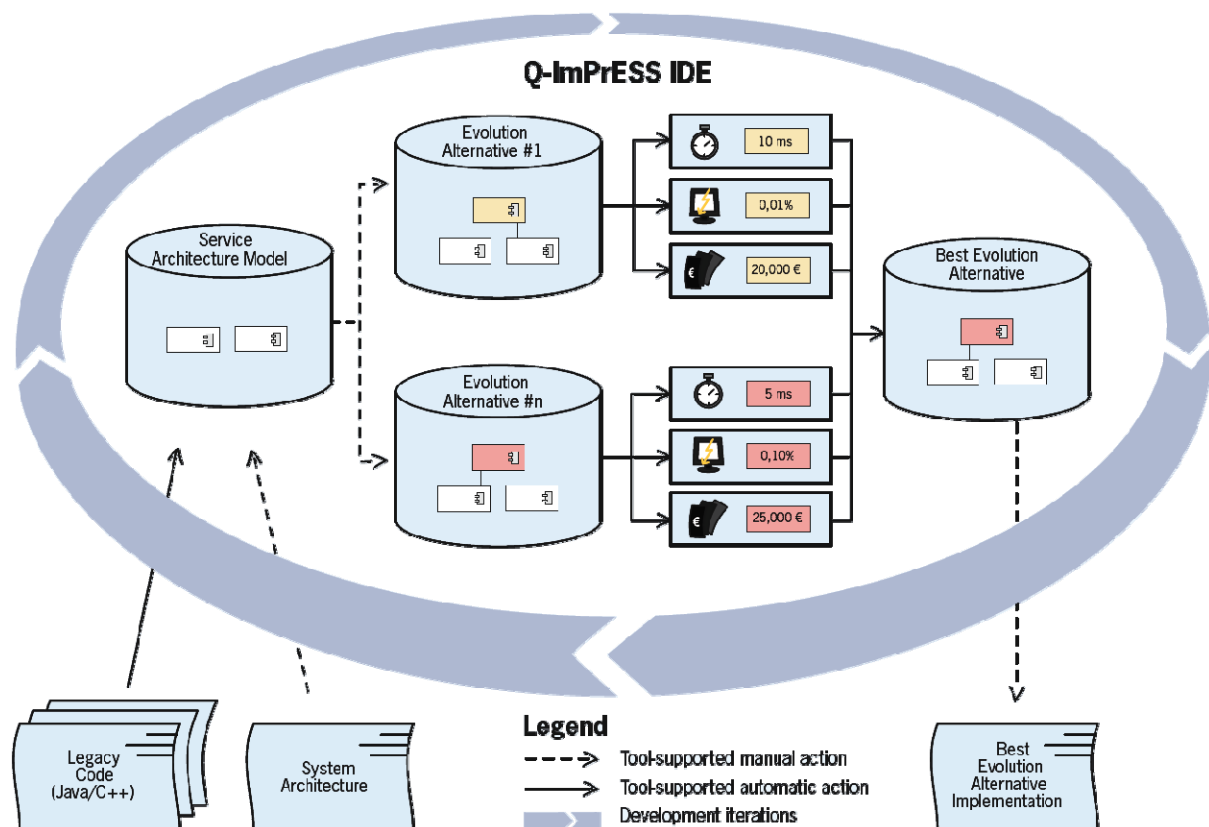




Figure 1: Overview of the Q-ImPRESS IDE

| | | |
|---|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

The quality prediction models are used to predict the non-functional quality attributes for a given service architecture. Together with the quality model for maintainability, the models thus provide the software designer and maintainer with a complete picture of the consequences of his design changes at both the service level and the service architecture level. The software designer or maintainer can experiment with different service designs and architectures and select the best one for his system, without having to write a single line of code.

| | | |
|---|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |


3 Who can benefit from Q-ImPrESS

The benefit of using this approach is to enable the creation of service oriented systems with guaranteed end-to-end quality. Moreover, the equally important ability to evolve a software system is supported and the quality of service is sustained over the lifetime of the system. Overall, this enables the exploitation of the benefits of service orientation for critical software systems. It is obvious that a quality impact analysis of newly developed and evolving systems is a crucial pre-requisite for service oriented systems to take off. In particular, the often envisioned close integration, especially by the user, can only be achieved if such a tool-supported quality impact analysis approach is present. The key aspect here is the efficient and economical construction of modern service-based software systems, which nowadays cannot be envisioned without large-scale software reuse through integration of existing legacy code. Unfortunately, legacy code is neither service-oriented nor easy to change, which means that methods and tools are required to facilitate the reengineering and integration of such code in service-oriented systems, as well as to extract the additional information needed to make quality impact predictions about legacy code.

The beneficiaries:

- **Software architects** are assisted in the selection of the right design decisions for the creation of service-oriented architectures.
- **Software deployers** can estimate the performance and reliability of a certain setup of the execution environment for service-oriented software systems.
- Performance analysts can anticipate the throughput and response time of software system, evaluate scalability, and avoid bottlenecks in critical software systems.
- **Reliability analysts** can estimate the expectable reliability of the software systems.
- **Evolution experts** can predict the expectable costs of design decisions from the perspective of likely evolution scenarios.

End users can expect more reliable software systems with higher performance at lower costs. Critical software systems can offer increased quality of service to their users.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

4 Highlights of achievements

The project's global goal was to define a new service engineering method to create and evolve service-oriented software with predictable end-to-end quality. The goal has been achieved and the following paragraphs provide a more detailed description highlighting the most important results. These results cover in particular (1) the Q-ImPRESS method, (2) the reverse engineering approach, (3) performance analysis, (4) reliability analysis, (5) maintainability analysis, (6) trade-off analysis, and (6) the Q-ImPRESS IDE (Integrated Development Environment).


The developed Q-ImPRESS method allows users (i.e. software developers) to easily evaluate qualities of multiple versions of a service-oriented system (e.g. different design alternatives for a system) and/or scenarios of its usage, but without the necessity to actually implement all of these versions/scenarios. Analysis results of different versions or scenarios can be compared by performing a trade-off analysis. The method defines a workflow and all necessary steps that have to be applied, i.e. analyses to be performed and tools to be used (all of them described below). Besides, the method defines necessary models (to be precise – their meta-models) that are used during the workflow and in which the intermediate and final results are captured. The basic model is the Q-ImPRESS Service Architecture Model (SAM), which allows describing service-oriented software in a unified way. The SAM is specified by the Service Architecture Meta-Model (SAMM).

In order to evaluate multiple different versions/scenarios, a Q-ImPRESS SAM model of the software has to exist. However, in case of already existing large legacy systems, such a model does not exist. To create it by hand would be very tedious and error-prone task. Therefore, the Q-ImPRESS toolset offers possibilities to automatically or semi-automatically derive the required model from the system source code via reverse engineering methods. The model will contain both the structural information (components and their connections) but also behavioural information (behaviour of the identified components).

Once the model exists, the user can set up several versions/scenarios with different parameters and then perform analyses of them with respect to different aspects, i.e. performance, reliability, and maintainability.

One of the most important results is that all of the above mentioned analyses and processes are available for direct usage from a single tool – the Q-ImPRESS IDE. This IDE is an extension of the Eclipse IDE. As the Eclipse IDE is one of the most popular IDEs and the Q-ImPRESS IDE follows its usage approaches, adoption of the Q-ImPRESS IDE by users should not impose any obstacles (this was also shown by the project's industrial partners that used the IDE during the project). Using the IDE, new software and its SAM can be created, however old legacy systems can also be imported and “reverse engineered” into a SAM. Then, several version/scenarios (in IDE called evaluation alternatives) can be prepared and analyses applied on them – again directly from IDE

As mentioned above, the Q-ImPRESS IDE was used by industrial partners and therefore the Q-ImPRESS method and all analyses were applied on the developed demonstrators.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

5 The results

This chapter provides an overview on the major project results of Q-ImPrESS.

5.1 Analysis of multiple quality dimensions

One of today's issues in software engineering is to find new effective ways to deal with the increasing complexity of software-intensive computing systems. In this context, a crucial role is played by meeting quality requirements, such as performance and reliability.

In the scope of the Q-ImPrESS project, multiple quality prediction tools have been integrated into a single Q-ImPrESS Integrated Development Environment (IDE).

The majority of quality analysis methods usually addresses a single quality attribute (e.g., performance or availability), whereas the major challenge tackled by the Q-ImPrESS project is exactly finding the best balance between different, possibly conflicting quality requirements that a system has to meet, and cost constraints (e.g., maximise performance and availability, while minimising cost).

For these multi-attribute problems, there is usually no single global solution, and a promising way to deal with them is to exploit multi-objective optimisation where the objectives represent different quality attributes. The aim of these techniques is to devise a set of solutions, called Pareto optimal solutions or Pareto front, each of which assures a trade-off between the conflicting qualities.


In other words, while moving from one Pareto solution to another, there is a certain amount of sacrifice in one objective(s) to achieve a certain amount of gain in the other(s). This activity is time consuming, thus the software architect needs an automated method that efficiently explores the architectural design space with respect to the multiple quality attributes. According to Q-ImPrESS goals, this method has to be automatic, allowing a quick exploration of a potentially large design space and efficient so to guarantee a decision making process able to find the optimal solution against multiple criteria without violating the time constraints imposed by development processes.

The Q-ImPrESS project has developed in this direction an evolutionary algorithm. Yet still, the derived optimisation process is time-consuming.

To overcome these drawbacks, we have developed a model builder where different design alternatives are automatically generated and evaluated with respect to different quality attributes, providing the software architect with a powerful decision making tool. The latter enables the selection of the architecture that fits multiple quality objectives in the best way.

The approach is centered on a hybrid technique, where an initial architecture of the system (fulfilling its functional requirements) is taken as input. Based on this initial solution, a search problem is formulated by defining "degrees of freedom". The identification of a significant set of design alternatives is then based on evolutionary algorithms.

The goal of the analytical optimisation step is to derive very efficiently an approximated Pareto front with respect to a simplified search space.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

The obtained results are used as input candidates for an evolutionary optimisation of the original search problem. In this way, more accurate estimates for availability and performance metrics and a larger Pareto optimal solution set can be obtained. The proposed method can lead both to a reduction of development costs and to an improvement of the quality of the final system, since an automated and efficient search is able to identify more and better design alternatives.

5.2 Trade-off decisions and support of design alternatives


It is often the case that the designed service architecture does not fulfil the imposed quality requirements, or, due to evolving contexts, there is an imperious need of improving certain quality attributes, such as performance, reliability, or maintainability. Hence, the software architect needs to produce variants of the original software architecture, in order to address the respective issue. In many cases, there are more than one or two variants that could be considered as feasible solutions to the mentioned problems. Consequently, the architect needs to be able to compare architectural alternatives by performing the trade-off analysis between maintainability, performance and reliability. Examples of possibly considered changes are behavioural model changes, deployment model changes, etc.

In Q-ImPrESS, we have focused on the automatic generation of implicit architectural alternatives imposing a set of degrees of freedom. The generated architectural alternatives give rise to different choices. Sometimes it is the case that the set of architectural changes is too large to be determined by the architect himself. In such case, as described in the previous section, the trade-off analysis equals to Pareto-optimal analysis, which provides a set of Pareto-optimal solutions, out of which one is to be selected, by the architect, for implementation.

If the number of alternatives is less than 10, the ranking of the selected solutions is obtained by expressing the quality of the architecture as a weighted sum of individual attributes. This is done by the Analytic Hierarchy Process (AHP) that has been implemented in the Q-ImPrESS IDE as a standard SWT wizard. The implemented AHP method assumes comparisons of the quality of service (QoS) criteria, which in Q-ImPrESS are the (potentially conflicting) non-functional requirements on the system (given by the performance, reliability, and maintainability metrics). The decision making is concerned with which alternative architecture solution suits these criteria in the best way.

To perform a trade-off analysis, the result model is created and filled with the analysis results provided by the Q-ImPrESS IDE, which are passed to the trade-off analysis tool that computes the optimal architectural solutions, with respect to one or more QoS attribute metrics. To achieve this, we have developed and integrated into the backbone, the Q-ImPrESS Result Viewer, which is the starting point for activating the AHP Wizard. The Result Viewer enables users to get an overview of the performance, reliability and maintainability data gathered from the prediction tools provided by the Q-ImPrESS IDE.

The Q-ImPrESS IDE is connected to the AHP Wizard by invocation. In order to access the AHP Wizard, the user has to mark the alternative designs that are going to be considered for the trade-off analysis. In order to make this selection as easy as possible, and to maintain a

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

consistent work-flow, we have integrated the invocation of the AHP Wizard into the Result Viewer, which is used to display the results of the already run quality predictions.

The AHP trade-off analysis results are both textually (in a table), as well as visually displayed, at the end of the trade-off computation. The quality of a design alternative is calculated as a weighted sum over performance, reliability and maintainability values, hence we have chosen to represent the trade-off data via stacked bar graphs, which show almost all the data involved in the AHP method. Each architectural alternative considered for the trade-off analysis has a corresponding stacked bar. The stacked bar shows the contribution (weight) of each quality attribute, as well as the global value of the weighted sum of QoS values, per alternative.

In order to identify trouble spots in SAMM-based design models, with respect to performance and reliability, and to support the software architect to generate feasible ways of mitigating the identified problem, we have proposed a rule-based methodology to address the previously mentioned issues. The set of rules is intended to assist the software architect in exploring the possible design solutions.

To accomplish this goal, we have focused on the definition of high-level rules able to help the software designers to gain insights from the results obtained by running the performance and reliability tools. We have defined the methodology for localising and mitigating performance and reliability problems, based on an abstract system model.

The performance problem diagnosis has two main steps:


- Step 1. identification and localisation of the performance problem
- Step 2. definition of rules for the mitigation of the performance problem

We have considered as causes for the potential failure to meet the performance target, bottlenecks and long paths; consequently, we developed rules for both bottleneck and long path identification and mitigation, which we have applied on the client-server example of Q-ImPRESS, using actual predicted values coming from the Q-ImPRESS IDE. The result was promising, in the sense that by applying our rules, several architectural alternatives have emerged.

Regarding the reliability rules, the method exploits information about the reliability properties of each component, as well as architectural information about how components are assembled. In particular, when a failure occurs, a two-step approach is followed:

- Step 1. identification of the source of the failure based on *sensitivity* analysis
- Step 2. definition of *rules* for the mitigation of the reliability problem, based on *fault-tolerance techniques*

In particular, we used design and data diversity techniques for the definition of rules mitigating the reliability problem. The designer can decide to explore the solution space trying to apply all the rules together and then he can select the alternative architecture that leads to the best trade-off between reliability, cost and performance. We have also applied the reliability rules on the client-server example with computed component sensitivity values, and delivered a set of possible alternative solutions to removing the reliability trouble-spots.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

5.3 Integrated tooling / reverse engineering

Q-ImPRESS offers a unique integrated toolset providing performance, reliability and maintainability analyses that are based on the Q-ImPRESS service architecture model (SAM). All of the analyses can be started from the single Q-ImPRESS Integrated Development Environment (IDE), which is based on the Eclipse platform. All model data, design alternatives, analyses, and results are accessible using the IDE. The IDE unifies the way, software architects, developers, and domain experts interact with modelling, predicting, and analysing approaches for service-oriented software architectures. Furthermore, the above mentioned trade-off decision support maps all quality dimensions to a single result representation, which respects project-specific, personal, and evolution-supporting preferences for the selected design alternatives.

The IDE offers a unified way to configure analysis executions and to represent analysis results for any quality analysis. The SAM enables an existence of a single software architecture representation with reusable services, which are ready for any quality analysis.


5.3.1 Integrated tool support of design alternatives

Design alternatives can be easily managed and versioned from the IDE. For example, an alternative representing a certain evolution scenario can be initialised from the IDE and analysed and then subsequently changed again and re-analysed. The IDE represents, manages, and accesses these design alternatives in a uniform way for all quality dimensions and analyses. Hence, the whole design space spanned by performance, reliability, and maintainability can be explored with this single IDE. Instead of single-dimension optimisations (e.g. only optimal reliability as supported by pure reliability analysis IDEs) or the need to manually translate model representations from a certain formalism into another, the developed IDE allows for arbitrary design space exploration. Each single design alternative of such a design space exploration can combine optimisations of the following dimensions: performance, reliability, and maintainability. For typical software systems, the most optimal design alternatives are likely to comprise optimisations of all quality dimensions, those solutions are supported by the IDE.

5.3.2 Central model infrastructure

As the lowest level, the IDE has a central backbone that (i) provides access to the design alternatives, which hold instance of the SAM, (ii) an unified workflow engine allowing execution of analysis tools, and (iii) a central result repository for analysis results, which then can be accessed by automated optimisation approaches and/or by the presented trade-off analysis. All of these three parts (and also the whole IDE) are extensible via plugins, i.e. it is easy to add support for further quality dimensions (e.g. energy consumption) and/or to add new types of analyses (e.g. energy consumption prediction). In the following paragraphs, each of these three parts is described in more detail.

The backbone manages design alternatives, stores them in a file-based database and allows IDE users to choose an active alternative and work with it. From the user point of view, an alternative can be seen a folder storing instances of SAM models (usage, hardware, resource environment, quality of service annotation, service effect and architecture models). All plugins of the IDE can access these models via the backbone.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

The workflow engine executes sequences of jobs which are typically required in model-driven software engineering. Since Q-ImPRESS makes heavy use of model-driven techniques, model transformations (e.g. an instance of SAM is being transformed into an instance of the Palladio component model, which serves as input for the integrated performance analysis) and analyses (e.g. performance and reliability analyses) are executed using the workflow engine. The results repository is a central storage entity for analysis results of different quality dimensions (i.e. performance, reliability, and maintainability). Metrics for every dimension (e.g. response time and costs) are stored in the result repository and can be uniformly accessed from consequent elements of the Q-ImPRESS tool chain (e.g. trade-off analysis). Results visualisation plugins also take data from this repository.

5.3.3 Editors and viewers

The IDE offers multiple integrated views and editors for the SAM. For each model, there is a tree-based editor allowing easy (but not very comfortable) editing. Furthermore, for most of the models, user-friendly textual and visual editors are available. Very large service oriented architectures can be also visualised and interactively browsed in a hyperbolic tree and/or in a software map view. Hence, depending on the needs of users, the size of the software system to visualise and analyse, and the intended analysis or modelling activity, users can select appropriate views and/or editors.

Users preferring textual representation of service oriented software system (e.g. more code-centric thinking) can view and edit the instances of SAM using editors with syntax highlighting and code completion support. Users, which need or prefer UML-like representations, can use corresponding visual editors (e.g. to visualise the components involved in a service-oriented architecture). If full model details are preferred by a user, tree-based editors are the best choice within the IDE. Those editors offer full access to all model properties. For large software systems or if users aim at getting an overview of a software architecture (e.g. to understand the current state or to derive solutions to evolution problems), the hyperbolic visualisation is the most suitable view.

Another viewer is available for visualising the different quality prediction results.


All views and editors are seamlessly integrated in the IDE and access the central service architecture model.

5.3.4 Quality prediction and trade-off analysis

The IDE includes tools for quality prediction (performance, reliability, maintainability) and trade-off analysis. It provides a unified way to perform quality predictions based on SAM models. All prediction results are fed into a result repository. The result viewer allows for selecting alternatives which should be compared according to its prediction results by the trade-off analysis. Based on this selection, the trade-off analysis tool can be launched.

5.3.5 Reverse engineering

In scenarios where source code or portions of the source code exist (i.e. typical evolution scenarios), the SoMoX reverse engineering approach assists in the creation of service architecture models (static architecture and control flow of the behaviour) to easily apply the Q-ImPRESS method for evolution scenarios of existing software systems. SoMoX entirely

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

integrated in the Q-ImPRESS toolset and, by means of a few mouse clicks, creates an instance of the service architecture model from Java, C, and C++ source code. To create instances of the service architecture model, SoMoX relies on the abilities of Sissy which creates abstract syntax representations from source code. From the abstract syntax representation, GAST2SEFF creates Q-ImPRESS behaviour models. These behaviour models are service-level abstractions of behaviour expressed in source code and a prerequisite to further analysis of the service architecture model. All tools, SoMoX, Sissy, and GAST2SEFF, entirely integrate in the Q-ImPRESS reverse engineering tool chain.

The reverse engineered models represent an initial design alternative, from which further design alternatives (each representing an evolution scenario) can be derived. Furthermore, all editors and viewers can be used together with the reverse engineered models.

5.3.6 Consistency checking

The Consistency checker is a tool for checking consistency between implementation of a service in the Java language and its behaviour model in the TBP formalism. We say that the Java implementation is consistent with the behaviour model in TBP if the actual behaviour of the implementation reflects the behaviour model and vice versa. The Consistency checker enables protocol checks for the interaction between services. Hence, service-oriented architectures created from the Q-ImPRESS toolset, where each service is compliant with a corresponding protocol, provide the expectable reliability of services for end-users.

5.4 The service architecture model and extensible analysis workflows

The basis of the Q-ImPRESS approach forms (i) the Service Architecture Meta-Model, which is used to specify models of a service architecture and (ii) a workflow representing the Q-ImPRESS overall process.

5.4.1 SAMM (Service Architecture Meta-Model)

The Service Architecture Meta-Model (SAMM) specifies how to describe service-oriented architectures in a way that allows latter quality trade-off analyses for evolving software systems. As such, it provides the schema used by the users of the Q-ImPRESS method to describe architectures.

Additionally, the Service Architecture Meta-Model serves as foundation for all academic activities in order to define the Q-ImPRESS method and develop its tools. Particularly, it

- defines the storage layout for reverse engineered source code;
- serves as input needed to generate editors and transformations dealing with model instances;
- and finally, allows trade-off analyses to be performed.

SAMM as input for prediction methods

The SAMM's information serve as input to multiple models capable of predicting certain quality attributes. The quality attributes of interest are defined in D1.1 [1, Section 5.3]. They contain performance, reliability, and maintainability.

SAMM as input for trade-off analysis

The SAMM supports multiple quality attributes to reach the Q-ImPrESS aim to make trade-off analyses for evolving systems (defined in D1.1 [1, Section 5.4]). The SAMM is flexible enough to provide sufficient details to allow the generation of different kinds of prediction models – at least one per quality attribute of interest.

SAMM views

The SAMM supports multiple views to ease the modelling of different system aspects (see examples defined in D1.1 [1, Section 5.1]). They are reflected in SAMM decomposition into packages including static package, behaviour package, deployment package, usage package, and quality annotations (see Figure 2).

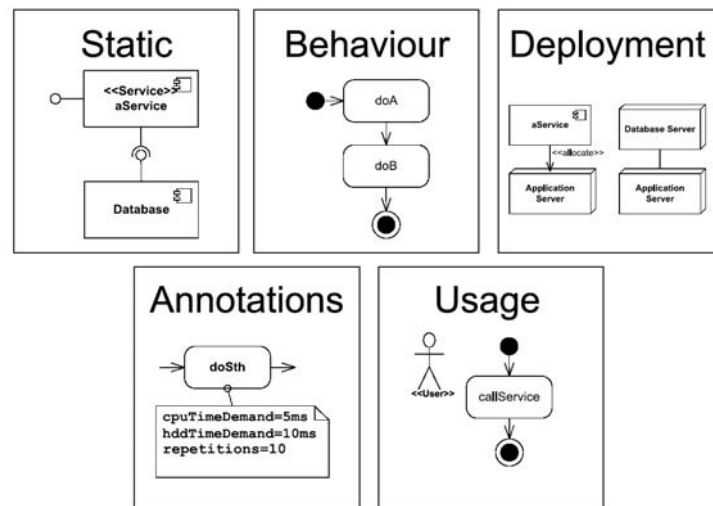



Figure 2: The different SAMM views

The following gives details on the introduced SAMM packages.

- Static Package*: This class of information contains details on interfaces, services, service interconnection, and service implementation (defined in D1.1 [1, Section 5.1]).
- Black-Box System Parts*: Important to Q-ImPrESS is the ability to model system parts whose internal structure is not known (defined in D1.1 [1, Section 5.1]). The method has to use estimates based on measurements of these parts. Gathering such information may need highly specialised measurements of the black-box parts. A prerequisite is that the black box parts can be seen as components, i.e., they have an explicit interface such as the one specified in the previous enumeration paragraph.
- Behaviour*: Behaviour characterises the control and data flow of the service-oriented architecture while executing the system (defined in D1.1 [1, Section 5.1]).
- Quality Annotations*: Quality annotations should express constraints on certain quality attributes, e.g. service level agreements (SLAs) (defined in D1.1 [1, Section 5.1]).
- Resource Model*: For the prediction of run-time properties information on the executing hardware parts and supporting software systems like operating systems or middleware layers is needed. The meta-model should clearly specify the amount of mandatory information. However, some parts of this information may be gathered by

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

tools, e.g., benchmarks of the underlying middle-ware layer (defined in D1.1 [1, Section 5.1]).

- (f) *Special Shared Resources*: Special attention is paid to the implicitly shared resources such as processors or disks. As defined in D1.1, the pivotal shared resources are processor, memory, disks, and networks (defined in D1.1 [1, Section 5.3]).
- (g) *Allocation Information*: The association of components to hard- and software nodes is needed to estimate the resource consumption and contention effects in prediction methods (defined in D1.1 [1, Section 5.1]).
- (h) *Usage Profile*: This view specifies the workload intensity caused by the users of the service oriented system and information on the parameters passed to the system in requests for service. It is needed to respect different types of system usages in quality predictions (defined in D1.1 [1, Section 5.1]).

SAMM editors

The Q-ImPRESS IDE provides four types of SAMM editors – (i) EMF tree editors, (ii) textual editors, (iii) graphical editors, and (iv) ad-hoc editors.

The tree based editors (i) cover the editing of any part of a SAMM meta-model instance. They show the repository content as a tree and allow arbitrary manipulation of the repository and arbitrary setting of properties of objects in the repository. In this sense, they are relatively low-level, especially for editing complex relationships, but on the other hand, they provide a useful backup for situations when something is wrong with the repository and other editors fail.


Textual editors (ii) cover the entire SAM. They define a concrete textual syntax for the information defined by SAMM meta-model. They allow displaying a part of the repository in the textual form and also propagating changes back from the textual form to the repository. This allows a user to edit the repository as any other source file with all benefits that modern text editors bring (syntax highlighting, auto-completion, etc.).

As some of the models are easier to interpret by using graphical model editors, the consortium has also developed a set of graphical model editors (iii). These graphical editors are only available for selected SAMM meta-model entities describing structural concerns. Particularly, they are the SAMM Repository model, SAMM Composite Components and the SAMM ServiceArchitectureModel. For these element types, graphical editors facilitate a graphical overview of the relationship between contained elements, which is especially useful for the ServiceArchitectureModel and Repository model. Additionally, visualisations of component repositories have been implemented that provide an appropriate visualisation of repositories containing a large number of components.

Ad-hoc editors (iv) are developed for SAMM entities which require special handling. The editors are often based on a wizard or form concept and allow step-by-step creation and editing of a model. An ad-hoc editor is, for example, provided for Annotation repository and it allows users to create and store a new quality annotation by selecting SAMM element to be annotated.

SAMM integration into the Q-ImPRESS IDE

Since SAMM represents an integral part of Q-ImPRESS methods, it has been integrated into the Q-ImPRESS IDE. The core part of the Q-ImPRESS IDE which is responsible for

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

management SAMM models is a *model repository*. It maintains various modelling scenarios called alternatives. Such alternatives contain SAMM model instances representing a given scenario. The model repository is an inherent part of every Q-ImPRESS project. The Q-ImPRESS IDE provides a model repository view which visualises model repository entities and provides CRUD operation over them. Furthermore, the Q-ImPRESS IDE exhibits an API for quality prediction tools to access the model repository.

SAMM extensibility

SAMM represents a static part of Q-ImPRESS method. However, it also declares two extension points for (i) a behavioural and (ii) an annotation package which permits introduction of new modelling entities. The motivation for these extensions is that not all possible entities (e.g., behaviour models, various annotation types) can be part of SAMM meta-model. Therefore, SAMM does not enforce any of them, but provides a way how to integrate them into the SAMM meta-model.

From the perspective of (i) the behaviour package, for each operation offered by a service or a component at least some information about the behaviour executed when the operation is triggered is needed. However, several types of formalisms for such a behaviour exist with different focus on different quality aspects, such as Stochastic Process Algebras, Activity-like languages, and Stochastic Petri Nets. There may not be a common ground for these modelling styles. As a consequence, the SAMM objective is to support a variety of behaviour modelling languages. To realise this requirement, the SAMM contains an extensible behaviour package. This package can be extended with different behaviour modelling languages like Service Effect Specifications (SEFFs), Process Algebra Expressions, or even the service's AST.


From the perspective of (ii) the annotation package, the prediction of run-time properties, such as performance or reliability, requires additional information, such as resource demands caused by certain processing steps in the architecture, loop iteration counts or failure probabilities. Therefore, this type of information is attached to SAMM elements as annotations. They are defined flexibly to reuse them in the various behaviour modelling styles. They also support specifying the source of the annotation, i.e., whether the value is a requirement, a predicted value, an estimation, or a measured value.

SAMM limitations

The prediction of quality attributes, such as performance, reliability and maintainability, still has its limitations due to the complexity of the resulting analysis models. The limitations caused by component and service internal behaviour and component and service interactions concern memory availability, memory transactions, internal component or service state, concurrency and OS schedulers, data streaming connectors, dynamic architectures or lack of exceptions.

5.4.2 Extensible workflow

The Q-ImPRESS method has been derived taking into consideration Deliverable D1.1 (Requirements Document) and Deliverable D2.1 (Service Architecture Meta-Model (SAMM) specification). The Q-ImPRESS toolkit aims at assessing the quality of complex, service oriented software systems, such as automation or telecommunication systems. These systems

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

are also characterised by long life expectations, so usually a deployed system is kept for several years before being replaced with a brand new one. Usually software systems are replaced when system maintainability or performance degrade below an acceptable threshold. Such big systems usually involve the work of several people with different roles: product managers (business level decisions), software architects (responsible for the whole system (or a single sub-system), high-level architecture and work planning), software engineers (responsible for component design and implementation).

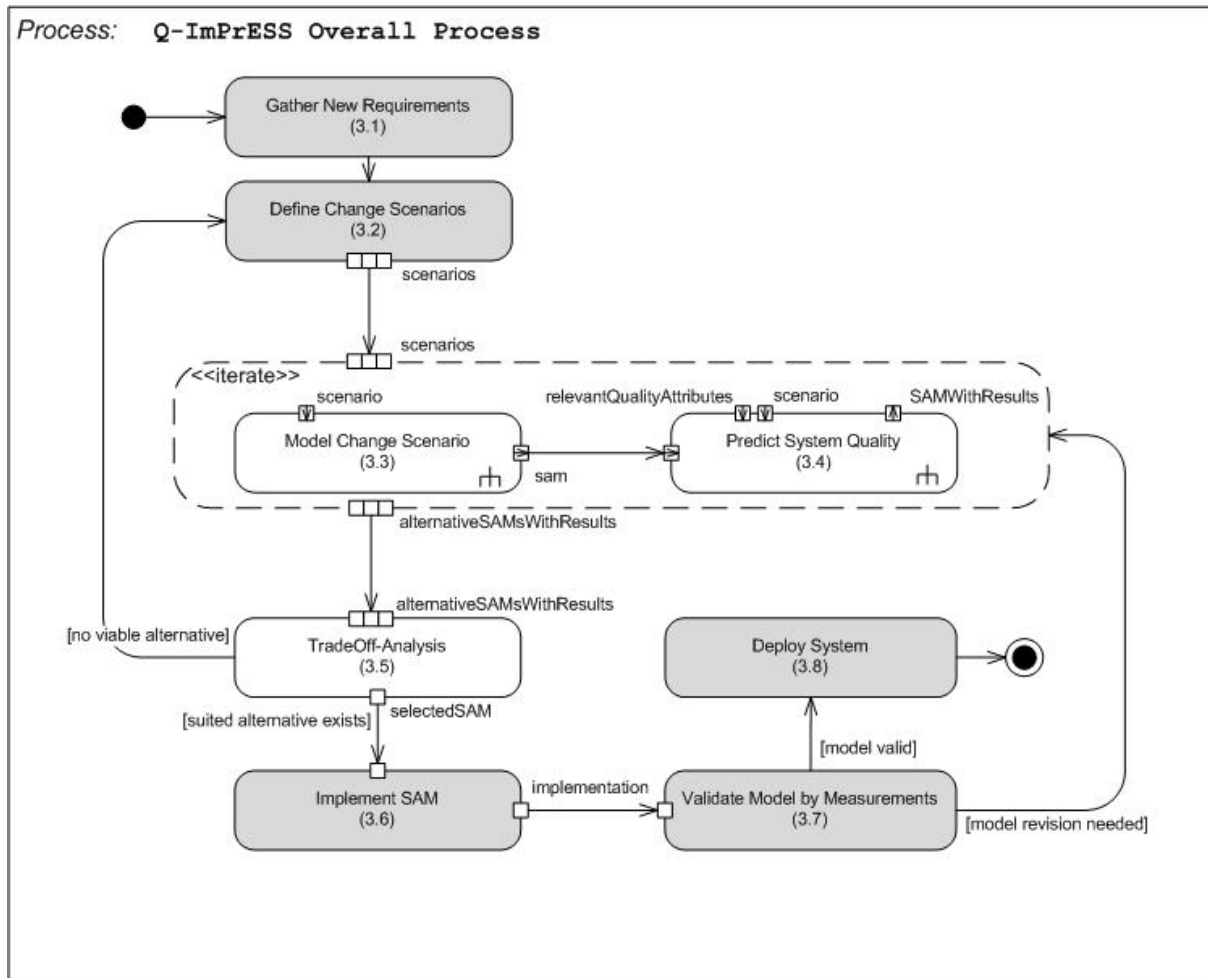
The Q-ImPrESS toolkit achieves its goal by building a Service Architecture Model (SAM) of the target system and performing analysis on the model. The adherence of the Service Architecture Model with the system it describes is crucial, as the predictability of the results heavily relies on this.

The system is embedded in a runtime environment that is not fixed but usually changes during system life-time; as runtime environment we consider the runtime platform the system is deployed upon, the system users and third-party service providers the system interacts with; these changes lead to new requirements, which produce change demands for the system.

The runtime environment is not the only responsible for change requests: for example new requirements can come from the product manager, as the competitors' solutions evolve in time and the product under analysis has to cope with new features required by the market.

Q-ImPrESS addresses the impact of these changes on the system quality, such as system performance, reliability and maintainability.

No matter what caused them, changes can affect the topology of the software system (assembly change scenarios), the deployment environment (allocation change scenarios), or the user behaviour (usage change scenarios).




Workflow tools

The Q-ImPRESS method is accompanied by different tools. Most of these tools are integrated within the Q-ImPRESS IDE:

The tools landscape is represented by the following tool classes:

- Reverse engineering tools to analyse and inspect software source code (of white-box components) in order to derive a standard, vendor-independent partial service architecture model;
- Tools that derive behaviour models from source code
- Model editing tools to update system models;
- Tools to derive and solve prediction models for quality attributes of services (i.e. performance, reliability and maintainability);
- Tools for model transformations between the different tools provided including methods to ensure consistency and traceability between tools specific models;
- Tools for analysis and simulation to make predictions on the quality attributes of services;
- Tools for trade-off analysis.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

To support the Q-ImPrESS workflow within the Q-ImPrESS IDE, the workflow engine has been designed in a generic way that provides the functionality to execute different workflow jobs, such as prediction analyses, or reverse engineering runs.

Workflow extensibility

The workflow captures multiple quality dimensions by integrating tools supporting performance, reliability, maintainability predictions. However, a new quality dimension can be added by introducing a new tool which will manage a demand. New tool introduction means integration the tool into Q-ImPrESS IDE which is inherently extensible due to plug-in nature.

The workflow engine is also designed in an extensible way, so that new tools in the Q-ImPrESS IDE can share the same mechanisms for configuring and starting the tool execution.

5.5 Resource models

This section gives an overview about the research on resource sharing and resource models that has been carried out in the Q-ImPrESS project.

5.5.1 Resource sharing

Many existing performance models only reflect coarse grained resources, such as idealised processors or idealised disks, which interact explicitly through system control flow. The Q-ImPrESS project has advanced state of the art by analysing the impact of resource sharing on a much finer scale, considering closer-to-real systems with memory caches and managed memory rather than their idealised counterparts. The importance of this particular contribution is magnified by the surge of modern computing platforms where memory caches or managed memory play an important role in the observed performance – in the delivered results, we demonstrate how the prediction error can be reduced from almost 100% down to around 10% when the resource models are applied in resource-intensive systems.


The contribution of the Q-ImPrESS project with respect to resource sharing is described in three parts, specifically the outcome of the resource sharing investigation, the methodology of the prediction validation, and the implementation of the resource models.

5.5.2 Sharing investigation

One significant outcome of the Q-ImPrESS project is a systematic documentation of the resource sharing effects that can occur on modern computing platforms, from processor execution units through memory architecture to operating systems and virtual machines.

The potential resource sharing effects are described and analysed through extensive measurements, described in the Q-ImPrESS Deliverable D3.3 on Resource Usage Modelling. For each of the resources investigated in the Q-ImPrESS project, experiments were carried out to carefully document how sharing occurs in various modes¹ of service composition, what are the typical and maximum effects of sharing and what is the workload under which such

¹ We distinguish especially sequential and parallel composition, with details outlined in the deliverable.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

effects are observed, and, finally, how the effects can be quantified. The highlights of the deliverable include:

- An investigation of detailed architectural parameters of hardware resources, such as the exact number and layout of caches. Knowledge of the detailed architectural parameters of hardware resources is required for analysing experimental results and designing performance models. The parameters, however, are seldom available in vendor documentation of hardware resources, severely limiting both result analysis and model design.
- An extensive set of experiments on sharing of memory content caches, which demonstrates that the performance overhead associated with the memory content caches depends not only on the (typically considered) capacity misses, but also on the (rarely modelled) cache bandwidth limits and hardware prefetching effects. The experimental results also demonstrate a significant difference in the duration of a memory access depending on whether the access hit or missed the cache, emphasizing the need to model the uneven progress of processes that share memory content caches.
- Experiments on sharing of collected heap, which demonstrate that the performance overhead associated with the garbage collector strongly varies with heap usage, suggesting that introduction of heap usage into performance models can improve modelling accuracy.
- Artificial workloads, designed to determine the limits of the performance overhead associated with sharing of individual resources.


A resource sharing experiment framework has been implemented that allows investigating the effects of resource sharing in selected modes of service composition (sequential execution on a call path and parallel execution in independent threads). The framework has been used to carry out experiments on multiple computing platforms and is now available as a generally applicable tool for conducting benchmarking experiments.

5.5.3 Validation methodology

Further work related to resource sharing has been based on the prediction validation activities, as described in the Q-ImPrESS Deliverable D4.2 on Prediction Validation. The general approach to prediction validation in the Q-ImPrESS project was to compare the predicted and the measured values of the quality attributes whose prediction is being validated. The validation process has been improved by automatically generating software systems together with their models and then using the systems for measurement and the models for prediction (this has allowed us to validate the quality predictions on a large number of systems).

Each validated system is constructed as an executable composed of randomly selected and randomly interconnected building blocks. When launched, the executable measures and reports its own quality of service attributes, which can be compared with the predictions. The building blocks are selected from real application workloads and the random properties of the validation system are highly configurable, making it possible to generate executables that resemble real applications. Although the actions performed by the executables obviously do not produce any useful application output, the workload exerted by the executables on the platform is the same as the workload exerted by real applications, and therefore suitable for validation.

The tool that generates software systems for validation purposes is now also available as a generally applicable tool, together with a library of measurements collected and used

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

throughout the project. Both the tool and the library are a valuable resource that can make validation of other performance modelling projects significantly more efficient (Q-ImPRESS is likely the only project to ever deliver a performance modelling methodology validated on thousands of systems).

From the resource sharing perspective, the validation has provided opportunities for resource sharing in a spectrum of randomly generated systems (whose implementations were instrumented so that performance measurements could be collected automatically). Besides information on prediction accuracy, the validation experiments have provided data useful for analysing the cases where prediction accuracy was low, possibly due to resource sharing. These cases were analysed and approaches to improve the prediction accuracy were proposed in the form of resource models.

5.5.4 Resource models

Based on the listed results, the Q-ImPRESS project has delivered and evaluated resource models for two important resources – memory caches and managed heaps. Both models provide choices of varying model complexity, which is obviously tied to model accuracy, but also – and that more importantly – to the amount of input data that the model requires. Dealing with the amount of input data is important from practical perspective, not only because large input data sets require long computation times during prediction, but also because some types of data are very difficult to collect. This holds for both the cache models and the heap models – the cache models require the stack distance profile,² and the heap models require the lifetime profile.³ Collecting both profiles requires specialised tools, which were also developed as a part of the Q-ImPRESS project.


Our most detailed memory cache model predicts the effects of cache capacity sharing on the number of cache misses. The model, described in Deliverable D3.4, is unique in that it is composable – the model outputs are of the same type as the model inputs –, it is therefore possible to model the effects of cache sharing between several components that make up a composite, and use the results as inputs for modelling the effects of cache sharing between several composites. As a simpler sharing model for the memory cache, we have developed a method based on the observation that applications appear to have a specific sensitivity to cache sharing. By measuring the sensitivity and predicting the utilisation of each component, we get enough data to predict timing adjustments in performance models.

For the managed heap, we have introduced a simulation model that predicts the occurrence of garbage collection in both single generation and multiple generation collectors. The model, also described in Deliverable D3.4, has been validated on multiple virtual machine implementations – the garbage collector behaviour is very much virtual machine specific, but the model tries to abstract away from as much of the collector complexity as possible to remain more generally applicable. A simpler sharing model, based on averaging the simulation inputs, is also available.

The accuracy of the models varies depending on many circumstances. In the case of the cache models and the architectures where resource sharing accounts for at least 10% prediction error, the maximum observed throughput prediction error was reduced from 96% to 29%, from 62% to 51% for response time prediction. The median observed throughput prediction


² A distribution of distances of the memory accesses in the least-recently-used access stack.

³ A distribution of lifetimes of the allocated objects measured in allocation time.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

error was reduced from 16% to 6%, from 15% to 8% for response time prediction. In the case of the heap models, the prediction of the garbage collection time is not sufficient for exact estimates, however, the shape of the prediction result graphs suggests that applications such as better vs. worse analysis or identification of situations with significant garbage collection overhead would work reliably. In both cases, the resource models advance state of the art in performance modelling considerably.

As a result of the resource usage modelling effort, the models have been carefully described and implemented as prototypes. In all cases, however, the application of the models is not a one-click affair – the developer has to decide what resources are worth modelling under given circumstances, use the specialised tools to capture the model inputs, and then use the resource models to adjust the annotations of the performance prediction models. The underlying technologies that the resource models are predicting are simply too complex to allow for predictions that would be reasonably generic and still completely automated. It is, however, reasonably easy to identify situations where taking the time to use the resource models pays off in terms of accuracy – these situations include for example architectures with major computationally intensive components or architectures that are required to run in memory constrained environments.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

6 The pilots

In the Q-ImPRESS project, two demonstrators and a showcase have been developed and analysed w.r.t. quality of service with the Q-ImPRESS approach by the project partners ABB, ENT and ITE.


6.1 ENT demonstrator

The ENT demonstrator assesses the applicability and usefulness of the Q-ImPRESS method and tools in a typical telecommunications domain scenario. Detailed information on the ENT demonstrator can be found in D7.1, D7.2 and the official project website. A description of the modelling procedure along with the detailed information on the evaluation results is available in D7.3.

The ENT demonstrator represents a typical research and development challenge found in current telecommunication networks. The challenge is to shift from a vertical industry model toward a horizontal one. The ENT demonstrator explores this challenge by creating a prototype system that extends ENT's existing, so-called legacy system with a subsystem comprised of 3rd party commercial off-the-shelf software (COTS). The legacy part of the demonstrator consists of a call control system that governs all operations regarding call handling, routing, etc. The call control system is extended by a DIAMETER subsystem, built from 3rd party COTS, that offers standardised authentication, authorisation, and accounting functionality according to the widely accepted DIAMETER standard. The DIAMETER extension is organised into two clusters: client cluster and server cluster. The clusters consist of active and passive (redundant) nodes that utilise OpenDIAMETER software for delivering DIAMETER functionality and rely on the OpenSAF middleware to regulate the availability of nodes within a cluster.

ENT evaluated performance, reliability, and maintainability tools as well as the AHP based trade-off analysis tool on the telecommunications demonstrator. The focus was on performance analysis as performance-related data can be collected in a reasonable amount of time, which is not the case for data related to reliability and especially maintainability. We could not use the reverse engineering tools to extract our models as a significant part of our code is written in proprietary or simple procedural languages such as plain C, which are not supported by the tools. Thus, all models have been created manually.

The most advanced part of Q-ImPRESS proved to be the performance prediction tool. It provides extensive performance modelling and predicting facilities which were used to analyse the relevant evolution scenarios from the telecom domain. Given the proper input, these performance facilities provided accurate predictions even for high level abstraction models of a system (in our test cases, on average less than 15% discrepancy between predicted performance and real-world actual performance). Current support for modelling and predicting reliability properties in Q-ImPRESS is still simplistic. The reliability tool misses support for important reliability (and availability) characteristics of a telecommunications system such as service redundancy, failover procedures and times, replication patterns, etc. ENT utilised the Q-ImPRESS reliability tool to analyse the availability properties of the demonstrator, which is a more important parameter for the telecom domain. Required failure probabilities were calculated from the component's availability estimations based on the service failover measurements. Our analysis has shown that the demonstrator has good

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

potential for achieving high availability properties. Application of the maintainability tool on the demonstrator has shown promising initial results but the tool also needs further refinements. Finally, the trade-off analysis has shown that the AHP method can indeed be very useful for quantifying the results provided by the prediction tools, but only when working with a limited set of considered alternatives and quality parameters.

6.2 ABB demonstrator

To evaluate the benefits of Q-ImPRESS in the industrial automation domain, ABB applied the methods and tools on the so-called Process Control System (PCS). Details on the architecture of the ABB PCS are provided in Q-ImPRESS deliverable D7.1, while details on the modelling and prediction results are provided in D7.3.

The ABB PCS system represents a typical distributed process control system. ABB customers use it to control and supervise industrial processes, such as power generation or pulp and paper handling. The system consists of a number of field devices (e.g. sensors and actuators) to manage the industrial processes. Networks connect the field devices to programmable logic controllers (PLC), which handle real-time data inputs and outputs. The PLCs in turn are connected to PC server nodes, which collect the data, manage long-term logs of sensor values, and handle alarms. Clients, such as operator workplaces, access the servers to interact with the industrial process. Each workplace offers a graphical representation of the industrial process as well as embedded live data from the connected field devices. Operators can manipulate the process (e.g., activate valves and manage alarms) directly through the software.


The ABB demonstrator for Q-ImPRESS focused on the server-side software of the ABB PCS. The server-side software consists of a number of Windows services, which can be remotely accessed by client applications. It includes several million lines of C++ code and is implemented using Microsoft technologies, such as ATL and COM.

ABB evaluated the Q-ImPRESS reverse engineering, performance prediction, reliability prediction, and trade-off analyses tools on the ABB PCS demonstrator system. The maintainability modelling and analysis tools were not applied due to missing cost estimations for the evolution scenarios, which are a necessary input. The threaded behavioural protocol checker was also not applied, because it supports only Java code but not the C++ code of the ABB PCS.

The performance prediction part of Q-ImPRESS proved to be the most mature. An expressive Q-ImPRESS performance model was built for the system based on measurements in a testbed. On average a maximal error of 10% between predicted and measured performance indices was achieved. Multiple evolution scenarios (e.g. usage profile evolution, resource environment evolution, allocation evolution) were analysed and the expected throughputs, response times, and utilisations were derived using the Q-ImPRESS tools.

The reliability prediction part still required substantial manual effort. It was not obvious which data to collect for the Q-ImPRESS models and how to collect this data. The validation of the data and the evaluation of reliability prediction remain difficult (e.g. systems would have to be monitored for years). ABB extrapolated failure report dates from a bug tracking system to derive failure probabilities for individual components. Using the Q-ImPRESS tools, an initial reliability prediction was conducted.

The Q-ImPRESS reverse engineering toolchain (SISSy and SoMoX) could only be applied to a limited extent on the ABB PCS code because of their insufficient support for C++ code with

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

Microsoft-specific extensions and missing experience for the clustering algorithms. ABB additionally applied the design space exploration tool PerOpteryx and the trade-off analysis tool AHP wizard on the PCS demonstrator model with encouraging initial results.

6.3 The eSOA showcase

The Enterprise SOA showcase has been developed by ITE in order to evaluate the benefits of Q-ImPRESS in the enterprise domain. Details on the architecture of the eSOA showcase as well as on the results are provided in the documents “D8.6-EnterpriseSOA_Showcase”, “D6.2-4-domain-specific-guidelines.doc” and “D7.3-Demonstrator_Results_Documentation”.


The showcase illustrates a typical and non-trivial SOA/ESB (Service Oriented Architecture / Enterprise Service BUS) scenario, which is the integration of various systems. The software components the showcase is composed of represent an order management system, a product management system, a customer management system, a pricing management system, a shipment management system, and an inventory management system. All components emulate real software systems and therefore provide the main characteristic functions of such systems. More important than the provided functions is how the components are connected to each other and how they interchange information data. The two most important interface technologies are Web Services and Messaging Services. Both have been modelled in the Q-ImPRESS main models. Last but not least a database is part of the showcase.

ITE evaluated the Q-ImPRESS method on the Enterprise SOA showcase, which embraces performance prediction, reliability prediction, maintainability prediction and trade-off analysis. The main alternative and two additional evolution scenarios have been modelled manually. Furthermore, collecting data for Q-ImPRESS took 40% of the time consumed for modelling the showcase models.

Enterprise applications typically rely on middleware software. That means that gathering performance measures of a given enterprise application includes the amount of time the middleware software consumes for performing its tasks. Further some tasks performed by middleware software can be easily modelled in Q-ImPRESS as additional components or services. Other tasks – e.g. transaction management – will lead to a huge modelling overhead. In order to cope with this issue an alternative model has been modelled implying the tasks of middleware implicitly as part of the components and services. If the Q-ImPRESS model has to reflect a varying middleware configuration an additional model alternative has to be created.

All analyses have been performed successfully, whereas the performance prediction seems to be the most mature kind of analysis. The performance prediction model was modelled based on measured values and the provided predictions had less than 5% discrepancy between the measured the predicted values. Additionally, an evolution scenario was modelled in order to compare different alternatives regarding performance behaviour. By using the Q-ImPRESS IDE support of design alternatives and the IDE editors, ITE was able to model different scenarios for QoS comparison efficiently.


In the reliability prediction model of the eSOA showcase, the failure probability for each database access has been set to 1%. Determining adequate values for the reliability prediction

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

model is difficult, because in a real enterprise application world the calculation of adequate values has to be determined by tracking the frequency of failures of a real system over a given time period.

The Q-ImPRESS maintainability prediction tooling provides a good possibility to automatically create a work plan, which shows all model elements that have to be changed, added or removed when moving from one model alternative to another. The work plan provides the possibility to define the efforts for each work plan entity based on a service architecture model. That means that the model has to cover the complete software system in order to get a realistic effort estimation.

The trade-off analysis tool helps quantifying the results provided by the prediction tools. The trade-off analysis has been tested successfully with two alternatives. When analysing the eSOA showcase, the trade-off analysis tool turned out to work well with a small amount of alternatives.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

7 Availability of results


Q-ImPRESS provides open source tools under EPL licence which are hosted on the OW2 global open source community (<http://jira.ow2.org>). There is no patent filed for Q-ImPRESS which could hinder its free availability.

Further information about the Q-ImPRESS project can be downloaded from the Q-ImPRESS web page (<http://www.q-impress.eu>). The section “Downloads” provide several resources:

- **imQ-ImPRESS IDE:** The official latest release of the Q-ImPRESS release
- **Q-ImPRESS IDE Drop:** A ready-to-use drop of the Q-ImPRESS IDE for Win32 platforms.
- **Eclipse installation instructions**
- **Q-ImPRESS IDE installation instructions**
- **Q-ImPRESS example project:** This project is very helpful for doing first steps in the IDE to understand how Q-ImPRESS works.
- **Q-ImPRESS source code:** A link to the public source code repository (<http://forge.ow2.org/projects/q-impress/>).

Further documents about the Q-ImPRESS project can be found in the section “Documentation”. The subsection “Deliverables” provides information about all deliverables of the project. The deliverables cover scientific and industrial results gained from the Q-ImPRESS project, as well as reports on further project activities. Information about the Q-ImPRESS method and guidelines, as well as the Q-ImPRESS Tool Manuals can also be found in this subsection. All deliverables are provided as PDF files.

A list of all academic publications that have been published in the scope of the Q-ImPRESS project is provided in the subsection “Publications”. Finally, several screencasts are provided in the subsection “Screencasts” in order to give a more intensive and clarifying explanation how Q-ImPRESS can be used.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

8 Potential impact of the results

The topic addressed by Q-ImPRESS is getting increasingly important in development and maintenance of software systems and software-intensive systems, and that made opportunities for increased impact of the results.

The socio-economic impact and the wider societal implications of the project are manifold. First, in a software evolution process the project aims for significant savings in development and verification efforts, and finding the best solution when different strategic discussions and constraints are taken into account. Second, the project emphasises on quality aspects in software products, which addresses the software development in general. Third, it provides certain concrete results that can be used in software industry in different domains, including software-intensive systems. Forth, it provides good examples in research for complex systems design. For this reason the dissemination and exploitation activities have been focused on a) concrete research and exploitation results and b) dissemination of general modelling and design principles for evolving software.

During their dissemination and exploitation initiatives, the members of the consortium have seen increased interest from both industry and academia, which have led to several opportunities for further development of the approach and concrete exploitation of the results.

8.1 The main dissemination activities


The internal dissemination has been achieved by several internal seminars by each partner. The partners from academia have used results as input for other research projects (for example MDU has combined results from Q-ImPRESS and national project PROGRESS and CONTESSE) and in education, master and postgraduate studies. Industrial partners have organised several seminars and workshops with other departments as well as internships for master students that worked with Q-ImPRESS, used Q-ImPRESS tools and Q-ImPRESS analysis methods.

The external dissemination in academia was realised through publications and organisation of research events.

The consortium has published 70 peer-reviewed papers, including 4 journal papers, expecting at least 10 more papers already submitted and expected to be published during 2011.

The consortium has organised several events at the prestigious conferences, for example:

- COMPSAC Workshop CORCS - Component-Based Design of Resource-Constrained Systems, 25 July 2009, Seattle, WA, USA
- ESEC/FSE Workshop QUASOSS - Quality of Service-Oriented Software Systems, 25 August 2009, Amsterdam, Netherlands
- Euromicro SEAA 2009: Model Driven Engineering Session, Service-oriented and Component-based SE track, 28 August 2009, Patras, Greece
- MODELS'10 Workshop QUASOSS - Quality of Service-Oriented Software Systems, 25 October 2010, Oslo, Norway

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

In addition, Q-ImPRESS has organised several discussion panels and tutorials at prestigious conferences, such as ASE, CBSE and QoSA.

The results of these activities are increased visibility of Q-ImPRESS and its methodology in the research communities of Software architecture domain, model-driven and component-based development domains, software quality domains.

A complementary dissemination was realised via the project web page, which includes a set of screencast of the Q-ImPRESS technologies, tools and methods.

8.2 Exploitation of the results

The industrial exploitation activities are subdivided into internal and external exploitation. The internal exploitation has its focus on internal projects where methods and tools provided by Q-ImPRESS are used or will be used for project implementation and quality assurance. The external exploitation has the focus on applying Q-ImPRESS in other companies. The partners ABB and ENT emphasize internal exploitation. ABB has started and is planning a number of exploitation activities. Some of them are:

- ABB Corporate Research plans to start a follow-up project in a close cooperation with an ABB business unit to analyse and start deployment of the Q-ImPRESS technology to have an immediate impact on ABB business.
- ABB is currently developing global guidelines for software development and software architecture design to be used by all ABB business units. The plan is to include the Q-ImPRESS method into these guidelines, so that all business units become aware of the method and its benefits.

ENT has similar plans. Some of the concrete activities are:


- Introduce Q-ImPRESS methodology to internal e-Health projects. The goal is to model existing projects' implementations and to compare model-based performance predictions with performance data seen in practice.
- Use Q-ImPRESS modelling and prediction tools in the upcoming research project on load balancing in next-generation telecommunications systems.

The SME partners (Softeco Sismat and itemis) focus more on external exploitation that includes deployment of the Q-ImPRESS methodology and technology in their projects with the external customers. For this reason they will offer training of the technology to their customers.

The academic partners (FZI, MDU, PMI, CUNI) have already started, or are planning with inclusion of methodology and technology in other research projects.

FZI is using the results of the projects to acquire new projects including the applications of the Q-ImPRESS method with new companies, and generally to enhance its marketing activities, presenting itself as leading expert in the fields of service oriented architectures and quality predictions. Concretely, FZI plans new collaboration Q-ImPRESS deployment projects with ABB and IBM. Further exploitation activities of the FZI include the establishment of a follow-up EU FI-PPP project.


MDU is integrating Q-ImPRESS results with the research results from other research projects, mostly with the PROGRESS Strategic Research Center for predictable development of embedded systems. Other examples are the following: A trade-off methodology is used in a

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

research projects about software evolvability with ABB Corporate research. At the moment MDU is involved in modelling the architecture of the ENT demonstrator in the ProCom component model and the behaviour in REMES that provides basis for modelling and analysis of embedded resources (memory, energy, CPU, etc.) and timing behaviour.

The exploitation interests of Politecnico di Milano are concerned with the use of project results in: (i) courses at Politecnico, for example teaching the Q-ImPRESS approach in a university course on “Computer Performance Evaluation, (ii) different research projects, for example inclusion of Q-ImPRESS results and methodology in the Italian project proposal DARWIN, and investigation of the possibility Q-ImPRESS in different application areas, such as cloud computing, energy efficiency and healthcare and aging well fields, and (iii) new consultancy opportunities (technology transfer initiatives).

The exploitation plans of Charles University lie mainly in the domains of research, research collaborations and teaching. This includes integrating the results of the project in the open source software maintained by the group, especially projects SOFA 2 and BEEN, which are hosted by the ObjectWeb (OW2) international consortium. Furthermore, a new Performance Evaluation course has been developed. The course reflects activities of the Q-ImPRESS project.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

9 Lessons learned during the project


This section reports on the industrial applicability of Q-ImPRESS (Section 9.1), some lessons learned during the project (Section 9.2), and potential for future work (Section 9.3).

9.1 Applicability of Q-ImPRESS

Due to its extensive tool support, the industrial applicability of the Q-ImPRESS method is potentially higher compared to many similar model-based prediction methods. It should be noted that the Q-ImPRESS method inherits risks and pitfalls that apply to any model-based prediction method. For example, missing goal orientation can lead to inappropriate models. Choosing the wrong abstraction level for modelling can neglect important hot spots in the system. Decision makers have to be made aware of the assumptions of the models to trust the prediction results.

Some additional applicability issues are however specific for the Q-ImPRESS method.


- **Integrated approach with comprehensive tool support:** The method uniquely integrates multiple mature research approaches in a single environment; the tool represents an adequate research prototype, but needs more stabilisation for wide industrial application. The method is technology-agnostic and has been applied in different domains (e.g. automation, telecommunication, or business information systems).
- **Limited integration with other tools and methods:** While the Q-ImPRESS tools are well-integrated with each other and can be installed into Eclipse-based development environments, they would have to be adapted for integration into different development environments. Existing UML models need to be recreated for Q-ImPRESS, as there are currently no model transformations from UML to SAMM.
- **Cost-effective only for large systems:** The complexity of the SAMM (>100 classes) requires a substantial learning effort (approx. 1 week). Q-ImPRESS models are more difficult to understand than simple queuing networks because they are developed for evaluating evolution scenarios. Therefore, it might not be justified to apply Q-ImPRESS for single evolution scenarios or smaller systems. However, the effort might pay off when the models can be reused.
- **Limited expressiveness of SAMM:** Although being complex, SAMM still lacks constructs to model many interesting evolution scenarios. There is no direct support for virtualisation, OS changes, application server configurations, transmission protocols, event-based communication, real-time scheduling, or dynamic architectures. There is also only limited support for modelling the middleware. Finally, there is no support for modelling industry automation specific scenarios, e.g. the introduction of a new standard or the support for new controller devices.
- **Missing data collection support:** The hardest, most time-consuming, and error-prone activity in modelling is data collection (i.e. determining resource demands, failure probabilities, etc.). For these activities, Q-ImPRESS offers no method or tool support and requires manual work or third party product use from the user.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

9.2 Lessons learned

While developing and applying the method we learned the following lessons, which were not clear to us at the beginning of the project:

- Data collection is most time-consuming:** In the beginning of our case studies, we vastly underestimated the efforts required for determining QoS annotations (i.e. transition probabilities, resource demands, and failure probabilities). While there are standard tools for performance measurements, it is still time-consuming to set up a distributed, service-oriented system, derive performance requirements, create load drivers, monitor performance, statistically analyse the data, and instantiate the models. For reliability estimation, data collection is even harder as there are no established step-by-step processes but rather many approaches (e.g. reliability growth modelling, statistical testing, code metrics) with different pitfalls (e.g. missing statistical validity, limited scalability).
- Model creation is an iterative process:** In our experience, finding a suitable abstraction level for modelling is difficult and requires several iterations. This is especially pronounced if there is limited experience with the system under study. Although model creation is currently depicted as a single step in the Q-ImPRESS process model, it is actually an iterative process. Multiple information sources (e.g. architectural documents, developer interviews, measurements, estimations) have to be combined to create a meaningful model.
- Static code analysis and models of the runtime architecture:** As expectable, static code analysis predominantly reveals the static architecture of software systems. The same applies to the developed reverse engineering approach of Q-ImPRESS which is applying static analysis. The ABB case study, which was much larger than previously analysed Java systems, showed that for some systems, a system decomposition based on dynamic analyses could be well-supporting QoS analysis. A QoS model for such a complex system must necessarily heavily abstract from the source code. The structuring of the source code may only bear a limited resemblance of the high-level system at runtime. Therefore, the decomposition based on pure static analysis is of limited value when aiming at high-level runtime abstractions (i.e. reverse engineering only a hand full of components which each represent for example processes). Future work could extend Q-ImPRESS with dynamic analysis methods, e.g. performance monitoring or bug tracking system analyses tools. Dynamic analyses could provide an additional perspective on software systems.
- How to design an integrated meta-model:** Finding a fitting abstraction level for a QoS modelling language is hard. The Q-ImPRESS SAMM is complex but it still does not support many practically-relevant evolution scenarios. Some elements in SAMM are currently not used by the prediction tools (e.g., processor caches). Future meta-modelling should carefully consider the eventual analysis of any meta-model element. To reduce the complexity of the model and better support certain evolution scenarios, specialised model support for extended evolution modelling would be beneficial.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

9.3 Future work

Although Q-ImPRESS features an extensive meta-model and rich tool support, there are many exciting directions for future work. Some of them are derived from the experience with Q-ImPRESS, others were for example discussed while collecting the requirements for the project, but excluded from the scope of Q-ImPRESS:

- **Integration of memory and caching effects:** the meta-model of Q-ImPRESS is already prepared to allow the specification of memory and cache sizes, but the performance and reliability analysis tools cannot yet take these values into account due to missing analysis or simulation techniques.
- **Smart editors for parameter dependencies:** parameter dependencies from the PCM must currently be specified using plain strings, whose syntax is not checked. Editors from the PCM workbench could be incorporated into Q-ImPRESS.
- **Make parsing by SISSy robust against Microsoft code:** in our experiments with C++ code implemented using Microsoft technologies, such as MFC, COM, and ATL, the integrated CDT parser of SISSy was not able to process all parts of the code and produced many error messages. To further improve the amount of analysable code, the underlying CDT parser should be replaced by a more robust alternative. During the development of SISSy, mainly Java code was analysed due to its wide-spread use in academia. More successful tests on (open-source) C++ system and case study reports are necessary before the tool can be recommended for practical use in industrial settings.
- **Make clustering by SoMoX more transparent:** configuring the weights of SoMoX to achieve a reasonable clustering of the classes in source into higher-level components is still difficult and requires expertise knowledge due to the complex interplay of the different metrics. More research is needed to find good weights for the metrics and more experience from analysing open source software systems should be documented.
- **More graphical editors:** graphical model editors for the usage model, target environment model, hardware model, service allocation and QoS annotation model could be implemented.
- **Methods for data collection:** As Q-ImPRESS offers only limited method and tool support for data collection, it would be conceivable to create technology-specific versions of Q-ImPRESS, for example to support data collection in .NET or Java environments.
- **Method for migrating existing models to Q-ImPRESS:** Model transformations from UML to Q-ImPRESS SAMM would be desirable to reuse existing architectural models.

10 Partners

The Q-ImPRESS consortium is a well-balanced mix of experts in research, validation and dissemination related activities.

Regarding research and development, the consortium could count on four of the Europe's leading research groups in the area of quality prediction (FZI, CUNI, PMI, MDU).

The large industrial partners (ABB, ENT) provided two highly relevant demonstrators used to validate the Q-ImPRESS platform.

A SME (ITE) developed a valid testbed in the enterprise domain, while the other SME (SFT) focused on the documentation of the workflow method and platform tools usage.

Figure 3 outlines the partner's involvement in the project.

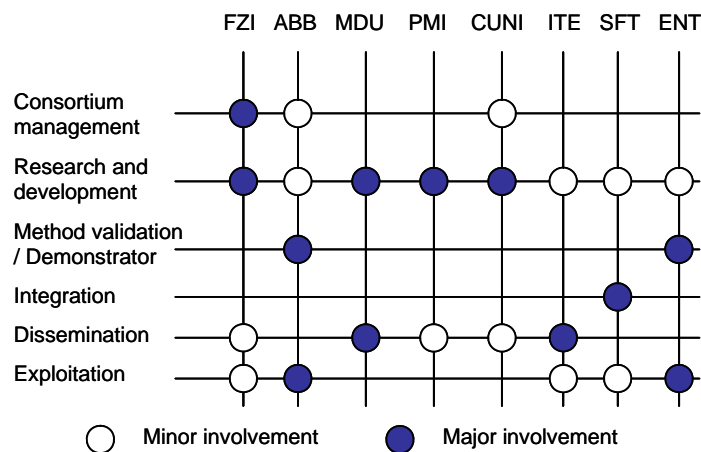



Figure 3: Partners involvement in the project

Forschungszentrum Informatik (FZI) an der Universität Karlsruhe

The FZI Research Centre for Information Technology in Karlsruhe is an interdisciplinary non-profit research centre funded partly by the economics ministry of Baden-Württemberg in Germany, partly by cooperative research projects and partly by direct contracts with the industry. The mission of FZI is to facilitate technology transfer of innovative solutions in information technologies and to provide a bridge between academia and industry.

ABB AG Forschungszentrum Deutschland (ABB)

The ABB Corporate Research Centre in Ladenburg (ABB AG Forschungszentrum Deutschland), with about 100 researchers, is one of the international ABB Group's major research centres, which totally employ around 650 researchers. Closely networked with six other research centres in Europe, US, and Asia, this is where fundamental concepts are created for the ABB companies' innovative, market-driven complete-system solutions.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

Mälardalen University (MDU)

Mälardalen Real-Time Research Centre (MRTC) is one of two research directions at the Department of Computer Science and Electronics at Mälardalen University (MDU) in Västerås, Sweden. MRTC is the leading research profile at MDU, and the leading group in Sweden in Embedded Systems related research.

Politecnico di Milano (PMI)

The "Politecnico di Milano" Technical University was established in 1863 by a group of scholars and entrepreneurs belonging to prominent Milanese families. The Politecnico di Milano is now ranked as one of the most outstanding European universities in Engineering, Architecture and Industrial Design, and in many disciplines is regarded as a leading research institution worldwide.

Univerzita Karlova v Praze (CUNI)

Charles University in Prague (CUNI) participates in the project through its Distributed Systems Research Group (DSRG). The entire university, founded in 1348, consists of 17 faculties and teaches over 42000 students. DSRG is a part of the School of Computer Science within the Faculty of Mathematics and Physics.

Itemis GmbH & Co. KG (ITE)


Itemis GmbH & Co. KG is an independent IT consultant with subsidiaries in Lünen (near Dortmund), Bonn, Hilden (near Düsseldorf) and Pforzheim (near Karlsruhe). Itemis was founded 2003 and today it has approximately 60 employees. The main focus of the company is the introduction of modern Software Development Processes, Model Driven Architecture (MDA) and Open Source based tool chains.

Softeco Sismat S.p.A. (SFT)

Softeco Sismat SpA is an independent industrial SME established in Genoa, Italy, in 1979. The company is a major supplier of software and automation systems for industry and research partners operating in complex environments with advanced IT methodologies, tools, hardware and software standards.

Ericsson Nikola Tesla d.d. (ENT)

Ericsson Nikola Tesla d.d. is a Croatian company and the largest specialised provider of modern telecommunications products, solutions and services in Central and Eastern Europe. The company has been in business for fifty eight years, and today, twelve years after Ericsson became its major shareholder (49.7%), is an expert centre in the field of information and communications technology and a significant constituent part of Ericsson in the Central European market unit as well as the global Ericsson research and development community.

| | | |
|--|--|-------------------------|
|  | Publishable Project Summary Report: D9.7 | |
| | Version: 1.0 | Last change: 31.01.2011 |

11 Contacts

In the following, general contact information for the Q-ImPrESS project as well as the contact information of all consortium members is provided. Further information about the Q-ImPrESS project is also available at the official project website (<http://www.q-impress.eu>).

Q-ImPrESS Project General Contact Information

E-Mail: contact@q-impress.eu

Q-ImPrESS Consortium Members

FZI Forschungszentrum Informatik

Dr.-Ing. Mircea Trifu
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe – Germany
E-Mail: mircea.trifu@fzi.de
Telephone: +49 721 9654 624

itemis GmbH

Andreas Graf
Meitnerstr. 10
D-70563 Stuttgart – Germany
E-Mail: andreas.graf@itemis.de
Telephone: +49 711 342191 0

ABB AG

Roland Weiss
Wallstadter Str. 59
D-68526 Ladenburg – Germany
E-Mail: roland.weiss@de.abb.com
Telephone: +49 6203 71 6211

Mälardalen University Sweden

Ivica Crnković
Högskoleplan 1
72220 Västerås – Sweden
E-Mail: ivica.crnkovic@mdh.se
Telephone: +46 21 10 31 83

Charles University Prague

Petr Tuma
Malostranske namesti 25
118 00 Prague 1 – Czech Republic
E-mail: petr.tuma@d3s.mff.cuni.cz
Telephone: +420 2 2191 4267

Politecnico di Milano

Raffaella Mirandola
Dipartimento di Elettronica e Informazione
Via Golgi 42
20133 Milano – Italy
E-Mail: mirandola@elet.polimi.it
Telephone: +39 02 2399 3523

Ericsson Nikola Tesla d. d.

Darko Huljenic
Krapinska 45
10002 Zagreb – Croatia
E-mail: darko.huljenic@ericsson.com
Telephone: +385 1 365 4734

Softeco Sismat S.p.A.

Marco Masetti
Via De Marini, 1
16149 Genova – Italy
E-Mail: marco.masetti@softeco.it
Telephone: +39 010 60261