CRitical S**YST**em Engineering **A**cce**L**eration

Schlussbericht

Fraunhofer

Förderkennzeichen 01IS13001O

DOCUMENT INFORMATION

Projekt	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Förderkennzeichen	01IS13001O
Organisation	Fraunhofer
Bericht	Schlussbericht
Dokumenteneinstufung	Public
Version	1.0
Datum	31. Januar 2017
Kontakt	Dr. Martin Becker
Adresse	Fraunhofer Platz 1, 67663 Kaiserslautern
Telefon	0631-6800-2246
E-Mail	martin.becker@iese.fraunhofer.de

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS13001O gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Inhaltsverzeichnis

1 Kurze Zusammenfassung und Überblick

1.1 Aufgabenstellung

Das Forschungsprojekt CRYSTAL hat zum Ziel die führende Position Europas auf dem Gebiet des Entwurfs eingebetteter Systeme für sicherheitsrelevante Anwendungen zu stärken. Dabei von besonderer Bedeutung ist die Fokussierung auf die Sicherung der Qualität und die Senkung der Kosten. Grundsätzlich sollen eingebettete Systeme in kürzeren Entwicklungszeiten einen höheren Reifegrat erlangen können sowie durch Wiederverwendung von technologischen und methodischen Entwicklungsbausteinen die Kosten der Herstellung gesenkt werden. Um die Projektziele auch an eine große Anzahl relevanter Partner zu vermitteln wurden verschiedenen Anwendungsdomänen (Automotive, Flugzeugbau, Eisenbahn- und Medizintechnik), in denen eingebettete Systeme von großer Bedeutung sind, in den Mittelpunkt gestellt um eine kritische Masse zu erreichen.

Daher sollen die Projektziele durch Orientierung an Fallballspiele aus den unterschiedlichen Domänen und stark durch die Ausrichtung an den Anforderungen der industriellen Partner erreicht werden. Projektziele waren dabei:

- Aufbau eines allgemeinen Arbeitsgebiets zum Thema Interoperabilität für eingebettete Systeme um einen Standard für modellbasierte Systementwicklung mit Schwerpunkt Zertifizierung und Betriebssicherheit (Safety) zu initiieren
- Aufbau eines innovativen Ökosystems getrieben durch CRYSTAL mit dem Ziel die unterschiedlichen Erfordernisse aus den einzelnen Domänen zu harmonisieren und zu vereinheitlichen. Insbesondere auf den Gebieten des Safety Engineerings, Simulation und Variantenmanagements.

1.2 Voraussetzungen

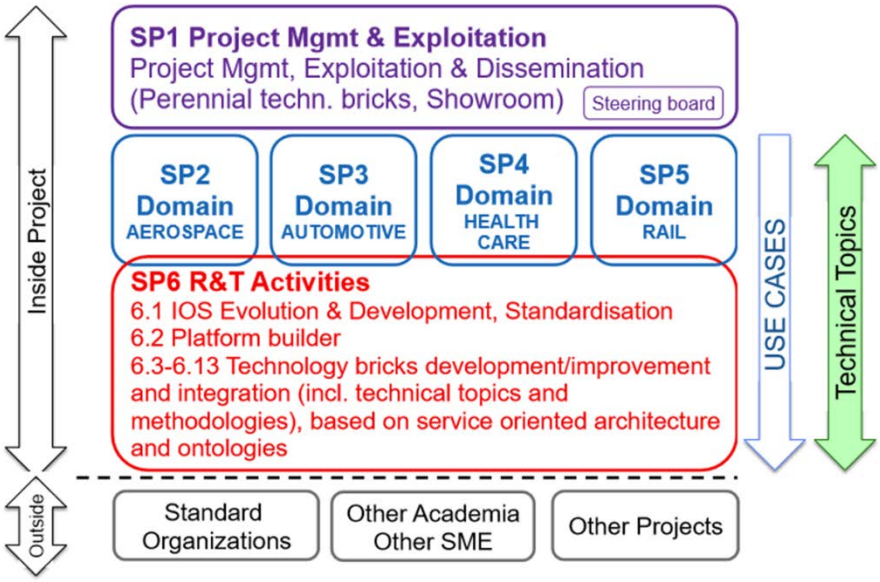
Das Projekt wurde als ein Verbundprojekt durchgeführt, dass verschiedene Partner mit unterschiedlichen und sich ergänzenden Kompetenzprofilen zusammengebracht hat. Alle Partner beschäftigen sich mit dem Entwurf eingebetteter sicherheitskritischer Anwendungen in ihrer spezifischen Rolle bspw. als Technologie-Anbieter.

Die wesentlichen für das Projekt relevanten inhaltlichen Voraussetzungen bestanden vor allem in den bereits teilweise erfolgten Standardisierungen betreffen Methoden, Werkzeugen und ihren Schnittstellen in den Domänen, so dass in CRYSTAL weitestgehend auf diesen Standards (etwa AUTOSAR[1] im Automobilbereich, FMI[2] als Cross-Domain Simulationstechnologie, …) aufgesetzt werden konnte. Darüber hinaus konnte auf den Ergebnissen der ARTEMIS Projekte CESAR (*Cost-Efficient Methods and Processes for Safety Relevant Embedded Systems*; 2009-2012), MBAT (*Combined Model-based Analysis and Testing of Embedded Systems*; 2011-2014), IFEST (*Industrial Framework for Embedded Systems Tools; 2010-2013*) aufgesetzt werden, in denen sowohl Teillösungen (sicherheitsgerichte Methoden für das Requirements-Engineering und Architekturentwicklung in CESAR, Methoden für die Integration von Test- und Analysemethoden in MBAT, Technologien für Werkzeugintegration in IFEST) erarbeitet wurden, aber auch weiterer Bedarf für die erforderliche Standardisierung von Engineering Methoden und Werkzeugen identifiziert wurden.

[1] Automotive Open System Architecture – http://www.autosar.org
[2] Functional Mock-up Interface – http://www.fmi-standard.org

1.3 Planung und Ablauf des Vorhabens

Um die Projektziele zu erreichen wurde CRYSTAL in Unterprojekte aufgeteilt. Eine schematische Darstellung dieser Aufteilung findet sich der nachfolgenden Abbildung.



Die Unterprojekte (SP) 2 bis 5 spiegeln die unterschiedlichen Domänen (Flugzeugbau, Automobil, Medizintechnik und Eisenbahntechnik) wider. Diese Domänenprojekte lieferten typische Benutzerszenarien und Fallbeispiele aus den jeweiligen Bereichen. Aus diesen Beispielen und Szenaren wurden dann die Anforderungen für die Forschungs- und Entwicklungsarbeiten abgeleitet, welche im SP6 organisiert wurde. Das Unterprojekt 6 (SP6) ist domänenübergreifend angelegt und hatte die Entwicklung einer Interoperabilitätslösung sowie die Entwicklung von allgemeinen Bausteinen, die in den Domänen-Unterprojekten wiederverwendet werden.

Das Projektmanagement wurde im Unterprojekt 1 organisiert. Die Arbeiten von Fraunhofer fanden in den Unterprojekten 2, 3 und 6 (SP2, SP3 und SP6) statt.

Die Fraunhofer Gesellschaft war im CRYSTAL-Projekt mit zwei Instituten vertreten: dem Fraunhofer Fraunhofer-Institut für Offene Kommunikationssysteme (FOKUS) und dem Fraunhofer-Institut für Experimentelles Software Engineering (IESE).

Die Institute haben im Projekt folgende Schwerpunktthemen bearbeitet:

Schwerpunktthema	Institut
Interoperability Specification (IOS)	FOKUS
Heterogene Simulation	IESE
Model-based Requirements Engineering	IESE
Safety Engineering	IESE
Tool Integration	FOKUS
Variability Management	IESE

1.4 Wissenschaftlicher und technischer Stand vor Beginn des Projektes

Vor Beginn des Projektes zeichnete sich der wissenschaftliche und technische Stand in folgenden Bereichen wie folgt aus:

- Die modellbasierte Entwicklung von eingebetteten Systemen unter Berücksichtigung von Zertifizierungsaspekten erforderte einen Erkenntnisgewinn auf dem Gebiet der durchgängigen Nachverfolgbarkeit (Traceability) von unterschiedlichen Objekten des Entwicklungsprozesses. Insbesondere die einheitliche Nachverfolgbarkeit über Werkzeuggrenzen hinweg stellte ein technisches Problem dar.
- Die Kopplung verschiedener Simulationsaspekte stellt eine technische und akademische Herausforderung dar. Von Bedeutung ist auch hier wieder die nahtlose Nachverfolgbarkeit solche Entwicklungsartefakte im gesamten Entwicklungsprozess.
- Im Bereich des Variabilitätsmanagements existierte kein allgemeingültiger und universeller Ansatz. Im Bereich der eingebetteten Systeme gab es eine Reihe von proprietärer Lösungen, die oft nur im Nachgang in den Entwicklungsprozess eingebunden wurden. Insbesondere ist hier auch eine nahtlose Nachverfolgbarkeit über Werkzeuggrenzen hinweg ein technisches Problem.

1.5 Zusammenarbeit mit anderen Stellen

Bei CRYSTAL handelt es sich um ein ARTEMIS-Projekt, welches europäisch aufgestellt ist. Die Projektziele wurden vom ganzen Konsortium bearbeitet. Das Konsortium setzt sich aus 71 verschiedenen Partnern aus 10 verschiedenen Ländern zusammen. Weitere Kooperationen fanden im Bereich der Standardisierung statt, vor allem mit den einschlägigen ARTEMIS, H2020 und ITEA Projekten.

2 Detaillierte Beschreibung

2.1 Projekt Ergebnisse

Die Beschreibung der Projektergebnisse wurde direkt aus den Projekt-Berichten übernommen und erfolgt daher in englischer Sprache.

2.1.1 Interoperability Specification (FOKUS)

Based on the use case description we identified several general interoperability needs / requirements which might be a relevant input for an interoperability specification (IOS). The aim is that the IOS and its concepts eases the instantiation of the Systems Engineering Environment, respectively also the Reference Technology Platform (RTP) that fulfils the use case needs and in particular the interoperability aspects of the use cases.

The following list summarizes the main interoperability requirements, which are relevant for the automotive use cases. The aim is that the IOS and RTP solution cover these requirements on a certain level in order to ease the realization of the use cases based on the CRYSTAL approach.

- Traceability - The linking of all engineering artefacts is one of the major interoperability requirements of the UC. We identified several aspects, which shall be addressed by the IOS traceability approach. The traceability solution shall allow creating typed links between elements. Traces shall be bi-directional and navigable in order to apply appropriate analysis methods, such as requirement coverage. Furthermore, the traceability information shall be consistent and available at all phases of the development process. It shall be possible to import legacy traceability information to the new CRYSTAL approach.
- Tool and Data Interoperability - Huge amounts of data will be produced during the development process by many heterogeneous engineering tools. These tools shall exchange the produced information seamlessly. Exchanged information shall be manageable in a consistent way within the different tools.
- Versioning Support - The huge amount of produced data within the use cases demands a proper mechanism to handle and to manage this data. A suitable solution is to version all development artefacts. This includes the synchronization and merging of diverse development branches. Therefore, the UC is requiring an IOS approach that supports version management of development artefacts produced and exchanged by different tools within the use case tool-chains.
- Multi-User Support and User Collaboration - One major challenge in the powertrain engineering process is that many users need to collaborate across locations and engineering disciplines. Therefore, the use case requires an IOS solution, which supports multiple users in distributed environments. Hence, the UC requires a mechanism to synchronize users and to ease user collaboration
- Distributed Development - Obviously, the CRYSTAL tool chains are distributed systems with typical challenges of distributed systems. However, the use cases require an IOS approach that supports the maintainability, consistency as well as availability of necessary engineering information.
- Automation Support - Within the CRYSTAL use cases, several engineering methods are applied which could potentially be automated, such as requirements coverage analysis. Therefore, the use cases require an IOS approach that supports the automation of engineering methods as services. These services shall be easily integrated into the engineering process.

- Analysis Support – Several CRYSTAL use cases make use of a high number of engineering methods dealing with analysis, such as hazard and risk analysis or failure mode and effects analysis. Hence, the IOS approach shall support these analyses by providing appropriate methods in order to automate, if possible, and to configure the analysis in a useful way. For instance, if specific input data for an analysis is requested, that data shall be transported transparently from the user to the tools which will perform the analysis.
- Simulation Support - Simulation is also an important engineering method to get assistance in several dimensions like calibration or testing in order to reduce the development time and costs. Therefore, the use cases are requesting support for simulation activities within the IOS principle. Simulation parameter and results shall be shared within the use case tool-chain.

2.1.2 Heterogene Simulation (IESE)

2.1.2.1 FERAL simulation Web Frontend

To enable the rapid use of various implementation technologies, a web based version of the Fraunhofer FERAL framework has been developed. Together with the simulation model search engine, which enables the storage, search, and management of simulation models, the FERAL web frontend supports the development of simulation scenarios that include functional mockup units.

Fraunhofer FERAL has been developed as an Eclipse application. For Crystal, a web based frontend has been developed that may be used for the rapid development of simulation scenarios. The main purpose is to adapt eclipse based scenarios, e.g. by adding functional mockup units or changing parameters. The development of simulation scenarios from scratch is possible, but less comfortable since the web environment lacks the ability of code completion and code assistance that is provided by the eclipse framework.

The FERAL web frontend is realized as a multi user environment that supports an individual workspace for every user. Each user has access to his own scenarios and to a collection of global simulation scenarios that are shared with all other users of the server. Scenarios may be replicated and created based on scenario templates. This way, the use of common scenario frameworks is possible. When a user opens a scenario, the screen changes to the normal scenario desktop. It enables users to edit files that belong to the scenario template. A standard text editor with syntax highlighting for the groovy language is provided for this purpose. Users may upload functional mockup units into their scenarios. Executed scenarios write results in a console window. When the user closes the web application or logs out, the simulation is not terminated. The FERAL simulation server is capable to execute multiple simulations concurrently and uses all available processors for this purpose. Administrators may configure the maximum number of concurrently executed simulation scenarios.

2.1.2.2 Simulative evaluation of heterogeneous models

Nowadays real products are to be assembled from a broad spectrum of parts obeying a complex set of laws each. On a system and subsystem level these parts of a product mutually interoperate in extensive ways. Models representing such parts (and a combination of such parts respectively) can be used for the simulation of their interplay within controlled system environments thus simulating the behaviour of the product to be evaluated. To this end simulating environments take a closed loop control based on events, time, and states to operate an arbitrary and often complex combination of simulation (sub) models.

In homogeneous environments this is realized by implementing the whole model and all its parts in one modelling language, but for different reasons this approach does not fit the needs:

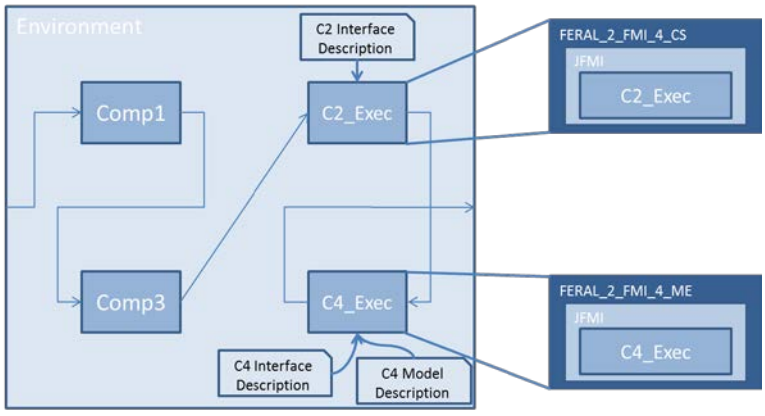
- Parts of the model implementations to be evaluated exist already but may be the intellectual property of another party. Therefore, they can’t be used until the “wheel is re-invented”.

- Already existing models, freely available but implemented in a different language cannot be used and therefore cause double work.
- Limitations of the used simulation environment force for re-modeling of available (sub) models.

Since products' parts are generally built by different producers each and every of the issues mentioned above may be valid for a simulation scenario, making evaluation unnecessarily expensive and time consuming or even impossible. Therefore, a simulation environment is needed that supports the deployment of heterogeneous models for heterogeneous simulation tools.

We have solved the challenge of a simulative evaluation of heterogeneous models by deploying model units like functional mock-up units (FMU) within a simulation framework that takes care of the interplay of the different FMUs deployed. By using functional mock-up interfaces (FMI) for co-simulation and model exchange it was be possible to build complex systems consisting of model parts based on a wide variety of simulation tools that already support FMI. While the simulation framework controls the process of deploying the contained heterogeneous models in a closed loop manner based on events, time and states part deployments are delegated to the FMUs that either execute their own simulation (co-simulation) or provide all data for the deployment of their underlying model within the specified simulation environment (model-exchange). As simulation framework we propose the usage of FERAL which provides the necessary domain and component types that are needed for a successful integration of FMUs. FERAL builds up simulation scenarios by combining and nesting simulation and directing components.

In a first step we had to adapt to FMU on a code basis as FERAL is Java-based and FMUs are C-based. To reach this goal we had to deploy JFMI (a Java wrapper for the FMI by the Ptolemy project). In a next step we have built wrappers for FMI for Co-Simulation as well as for FMI for Model Exchange to translate the semantics of the FERAL framework to those of FMI.



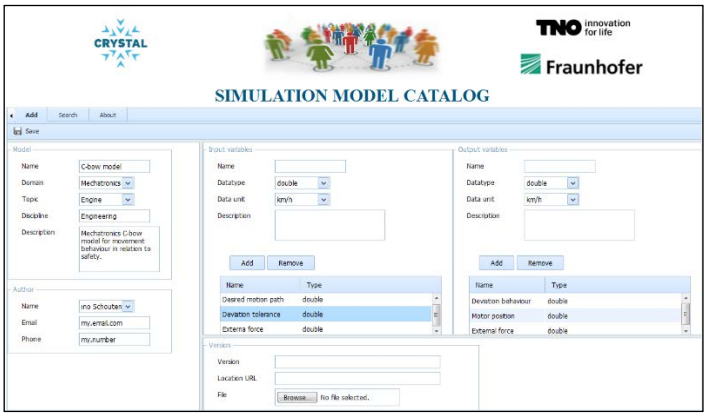
Signals handled within the FMI context, that is inputs, outputs, parameters, and status information as well as derivatives of inputs, and outputs w.r.t. time can be set and retrieved by the FERAL framework via the FMI wrapper. FERAL also counts for co-simulation algorithms and communication technology for distributed scenarios since FMI does not define these concepts. Using FMUs in FERAL not only opens the framework for a variety of models already described by differential, algebraic and discrete equations, but supports also the integration of executable models from different tools supporting FMI and thus keeps intellectual properties secure and prepares distributed evaluation scenarios.

2.1.2.3 Simulation search tool based on metadata

In order to support the development of an integrated simulation based on existing simulation modules a simulation search tools based on metadata has been developed. The simulation model catalog of CRYSTAL is a set of descriptions of (FMI based) simulation models with a reference to where the

model is actually stored. The simulation model catalog provides functions to create, read, update, delete (CRUD) model descriptions, and to search in model descriptions. Each model description includes name, domain and topic of the model, and the description of the input and output variables of the model, all conform the metadata definition.

The user interface of the simulation model catalog runs in web bowser, enabling easy access from virtually any place.



To enable the integration of the FMU search engine with the FERAL simulation environment, a second frontend has been developmed by Fraunhofer for the FMU search engine backend. This second frontend was developed using the GWT environment, the same toolkit that was used for the FERAL web frontend to prepare the integration of the FMU search engine and the simulation environment.

2.1.3 Model-based Requirements Engineering (IESE)

The work on Model-based Requirements Engineering was mainly conducted within work packages 3.4 and 6.4. The activities can be roughly divided into two parts: the definition of a SysML-based approach of modelling legal constraints, as well as the investigation of the possibility of an OSLC-based extension of the Artisan Studio tool and its prototypical implementation.

2.1.3.1 Modelling legal constraints

Model-based requirements engineering methods provide means to master the challenges that arise from increased size and complexity of (natural language) requirement specifications. While various techniques and tools are available, industry acceptance is still limited because of methodical uncertainties.

The IESE contribution addresses these uncertainties by presenting a systematic, SysML-based approach to model legal artefacts as laws or standards. The method is assessed in the context of automotive emission legislation.

As a result of the domain analysis, the origin of legal requirements has been described. In general, legal requirements define constraints for the system under development and/or the development process. Additionally, it has been identified that traceability plays an important role when regulatory approval is an engineering goal. Regarding requirements engineering in the automotive domain, it has been described that a tool-independent modelling approach might help to cope with the challenges resulting from size and complexity growth of requirements specifications.

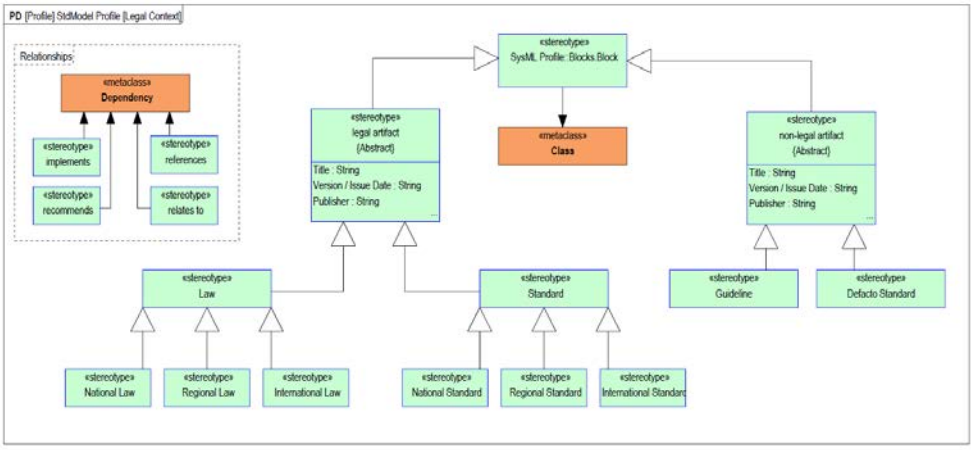
Some insights into SysML were given: important criteria for industrial acceptance are mentioned, the specific language constructs targeting requirements engineering are presented, and the UML extension mechanism “profile” is explained conceptually.

The three main tasks were:

- Develop a systematic approach to model legal artefacts using SysML

- Identify missing concepts and add them as part of a SysML profile
- Assess the developed approach in the context of the WLTP standard

The overall approach was structured in three main steps: “Model Legal Context”, “Model Legal Artefact”, and “Reuse Model”. Each main step was further decomposed into several sub steps, whereas each sub step was characterized and systematic step-by-step instructions were given. To foster comprehensibility, each sub step was illustrated with a practical example from the WLTP applying the developed SysML extension “StdModel Profile”.



As the main result of this contribution, it has been shown that the developed approach is suited to model legal artefacts similar to the WLTP. Key achievements of this contribution are:

- A systematic transformation process that transfers the (textual) information from the legal artefact to the model was introduced.
- The traceability challenge (cf. regulatory approval) was addressed by putting special attention to it in each individual step.
- Aspects that are relevant in the context of “modelling for reuse” were addressed. Typically, a model of a legal artefact is developed for reuse in several projects in the same or a similar development context. Some important aspects that are relevant when reusing a model of a legal artefact are described. The model is located inside a model library, i.e. a special type of package that is designated for reuse. Typically, the model library is imported to a project-specific model.
- A SysML extension “StdModel Profile” supplements the developed approach. This Profile” is an extension to SysML for the purpose of modelling legal artefacts as standards or laws.

2.1.3.2 Artisan Studio OSLC Adapter

The implemented OSLC adapter (Requirements Consumer) makes it possible to link (OSLC) requirements to artefacts within Artisan Studio (e.g. SysML-Blocks). Since the Artisan Studios UI is not modifiable sufficiently, it is not possible to display and link the requirements provided by an OSLC requirements provider right inside the tool. Therefore, it was necessary to implement a website, which serves as an interface between the OSLC requirements provider and Artisan Studio.

The Artisan Studio project structure can be displayed on this website. The different artefacts can be selected and linked to requirements. The OSLC link information is stored inside Artisan Studio, from where it will be read at every website refresh. Due to the lack of customizable Artisan Studio UI, this approach is not very user-friendly, but it basically demonstrates the possibility of linking RM artefacts with SysML artefacts.

2.1.4 Safety Engineering (IESE)

From the methodological perspective in the CRYSTAL project a modular and integrated meta-model for safety engineering had to be developed and this goal has been achieved.

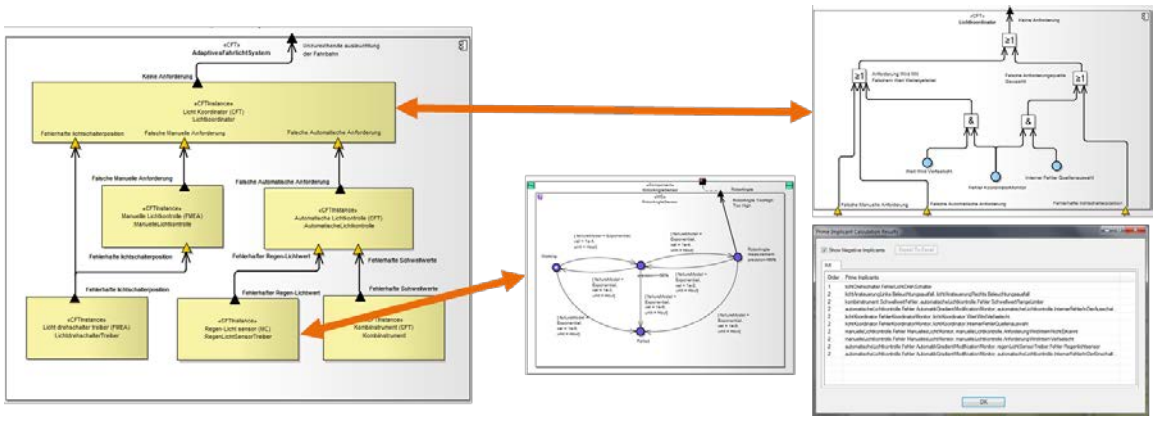
The meta-model integrates several methods for the documentation of safety relevant information, typically generated during execution of a development project. Examples of these methods are, Fault tree analysis, Failure Modes and Effects Analysis, Markov chains on the area of safety analysis and Goal Structured Notation (GSN) and Safety Concepts Trees (SCT) in the context of safety argumentation. The goal behind the integration approach is to formalize the traces existing between these techniques which implicitly appear during the realization of projects. This shall facilitate on the one hand the data exchange between tools, since by having common meta-model the discrepancies between the information representation is minimized. On the other hand, the integrated meta–model facilitates maintenance and reuse since due to the enhanced traceability between artifacts it is easier to find out where changes have been performed.

Three primary aspects have been pursued during CRYSTAL with respect to the tool implementation.

- Decoupling modelling features from analysis features. (interoperability related)
- Integration of the Fault Tree Plus computation engine in the tool chain. (interoperability related)
- Enhance safety modelling capabilities.(general)

In order to achieve the first two, the tool was restructured in a layered architecture. Now the modelling features fit in the front-end concept and the analysis features in the back-end concept. Between these two layers a third layer, called the exchange layer, plays an important role in keeping things decoupled. The exchange layer is responsible for model transformations based on a standard meta-model, as well as to establish the communication from/to front-ends and back-ends. Thanks to this concept, it is now much easier to bring together front-ends and back-ends in a flexible way. As a proof of concept, we have partially migrated our modelling features from Magic Draw to Enterprise Architect. In the first tool, modelling features are provided by means of Java Plugins, while in the latter through C-Sharp Addins. The exchange layer is able to communicate with both and any other front-end or back-end since it has been implemented under a client-server- approach, in which the communication is based on the standard meta-model. Thanks to the same mechanisms, the integration of new back-ends into the tool chain has been facilitated, so we could also achieve the integration of Fault Tree Plus as a computing engine.

The third goal was achieved in parallel to the other two. In the original state of the tool, only CFTs were offered to perform safety analysis. However, in the industry it is common to see other techniques in use, e.g. Failure Effects and Modes Analysis (FMEA) and Markov chains. We enhanced our modelling features by including these techniques. Moreover, the same integration and traceability concept used in C2FT was adapted for these techniques, allowing now the user of the tool to perform heterogeneous safety analysis. The concept is really promising, therefore we did not stopped at the failure cause analysis, but also applied the same concepts to the construction of safety concepts. This now allows to create safety concepts in a modular perspective and in the same hierarchical structure as existing in the architecture models and the associated failure models. Currently other techniques for hazard and risk assessment as well as for the construction of safety cases are being investigated.



Following Project innovations can be considered satisfied

- Reuse of engineering artefacts: The current version of the C2FT - Tool integrates better concepts for reuse at component level. Architecture models, safety concepts and safety analysis by means of CFTs are now fully modularized.
- Traceability between model elements and other artefacts, Improving the quality of requirements specifications. Traceability was one of the most important drivers on the modifications done to the underlying meta-model of the tool. Thanks to the development of component integrated methodologies for safety analysis, traceability has been made more transparent. This has been reflected in the tool by means of special connectors offered as modelling elements. Furthermore, traces are also created during other modelling activities with the help of automation mechanisms. For example, by linking architectural elements being referenced during the specification of a requirement. Mechanisms that allow identifying broken traces were also implemented, helping modellers to keep things consistent.
- Impact analysis to support change management (Only partially satisfied). Impact analysis has not been yet implemented as part of the C2FT – Tool (It was also not originally planned). However, we can consider the tool ready for it, since the implementation is based on a standardized meta-model, in which the interrelationships between safety artefacts are clearly defined. These relationships are created automatically during modelling through traces and are kept consistent during modifications of the models. The next step, in this regard is to create proper algorithms that allow evaluating how these relationships could be affected under “what-if” modelling scenarios.

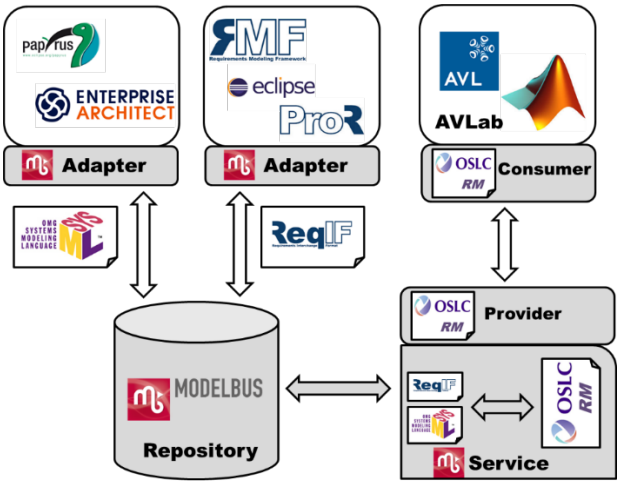
2.1.5 Tool Integration (FOKUS)

Based on the IOS and the Reference Technology Platform we have realized several use case scenarios by using the ModelBus Framework. Tools are integrated by an adapter pattern. An adapter eases the access to tool specific functionality and tool data. The CRYSTAL IOS defines specific interfaces that tools can offer. We have implemented these IOS interfaces for several engineering tools such as all Eclipse-based tools and Enterprise Architect. The ModelBus was used in one of the automotive use cases.

Basically, the IOS implies a Service Oriented Architecture (SOA) built upon the REST paradigm for distributed systems. The main concept in a RESTful architecture is the Resource which has a unique URI. A Resource can also contain further Resources (ResourceCollections). In principle, a REST resource contains all necessary information that resource provider and consumer require to exchange and to interpret resource content. OSLC defines minimalistic specifications for REST resources representation in order to ease the integration of lifecycle data and tools. For instance, OSLC community provides a specification for Requirement Management in which the format of

REST resources is defined for sharing requirement information across different tools. However, there are also other approaches to standardize requirement representation such as OMG's SysML or Requirement Interchange Format (ReqIF).

We've used Papyrus and Enterprise Architect for SysML based requirements and ProR as a ReqIF-based requirement authoring tool to implement an automotive related use case. The ModelBus framework integrates these tools. ModelBus provides Model Repository paradigm, which allows the storage and management of different lifecycle artefacts, and in particular models. Each entity that is managed by ModelBus. The Repository can be accessed via standardized interface and protocols such as REST, WSDL or HTTP. Thereby, a ModelBus Service realizes the transformation and mapping between SysML, ReqIF and OSLC RM. This ModelBus Service provides all requirement related data (SysML, ReqIF) as OSLC RM resources.



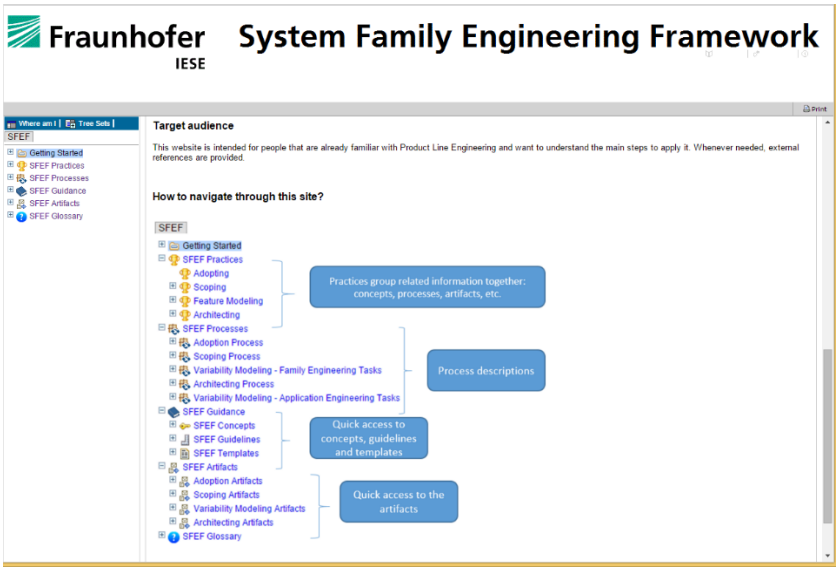
The demonstrator shows the feasibility of tool integration based on the CRYSTAL IOS approach. In particular it shows the traceability across different engineering tools.

2.1.6 Variability Management (IESE)

2.1.6.1 System Family Engineering Framework (SFEF)

System Family Engineering Framework (SFEF) provides a customizable method, respective techniques, guidelines and best practices to engineer families of high-integrity (including safety critical) systems, which can be followed in different application domains.

Specific features of the framework comprise the planning and specification of high integrity families, modularization approaches, and orthogonal variability management across several system levels. The framework is based on several existing approaches, in particular Fraunhofer PuLSE, SEI Framework and the second generation product line approach.



The framework covers the following aspects: Scoping, PL Adoption, Variability Modeling, Configuration Management / Lifecycle management issues. The framework is not tool-specific, but it is tool-supported. That means, the activities described in the process do not rely on a specific tool, but we provide guidelines on how to adopt a particular process using a specific tool.

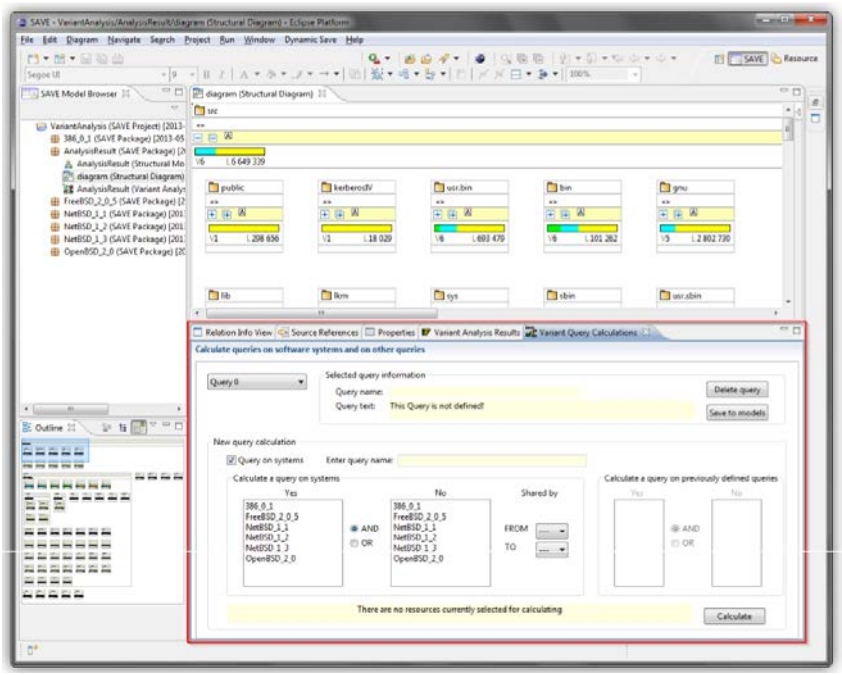
The SFEF product scoping process is well supported by the Product Feature Matrix (PFM) tool. The product family scope can easily identified and communicated between all stakeholders within the project team and to the customers. The allocation of each feature to concrete products enables a common understanding of the required commonality and variability the product family. This commonality and variability information is essential for the product family architecture development because Reusability, Modifiability and Expendability are significant architectural drivers of a product family's architecture. Each feature is described in the product feature matrix. If there is a customer visible difference between product variants the feature needs to be decomposed. If there is no variability, the feature shall be kept on a top level and shall not be detailed e.g. for completion purpose. This ensures that the PFM can be easily maintained and communicated also in environments were the product family scope is always in evolution. The assessment of the feature provides a common understanding on the importance and readiness of each feature. This is an essential information for the development schedule of the features.

Feature Name	Products				F_Characteristics						Architectural Components						
	Variant 1	Variant 2	Variant 3	Variant 4	Level	Parent Row	Usage Probability	Generality Probability	Core-Base Factor	Effort Savings Potential	Component 1	Component 2	Component 3	Component 4	Component 5	Component 6	Component 7
XYZ system					0	0	0%	0%	0%	0%							
Capability Features					1	3	0%	0%	0%	0%							
Domain 1					2	4	0%	0%	0%	0%							
SubDomain 1.1					3	5	0%	75%	75%	-75%							
Feature A	✓	✓	✓	✓	4	6	100%	50%	125%	375%	0,5						
Alternative A1.1	✓	✓	✓	✓	5	7	100%	100%	100%	300%							
Alternative A1.2	✗	✗	✗	✗	5	7	0%	0%	0%	0%							
Feature B	✓		✓		4	6	50%	25%	81%	81%	0,5						
Alternative B1.1	✓		✓		5	10	50%	50%	88%	88%							
Alternative B1.2	✗	✗	✗	✗	5	10	0%	0%	0%	0%							
										ESP	228%	300%	200%	44%	88%	97%	196%
										CBF	103%	100%	103%	70%	88%	97%	98%
										GP	38%	100%	75%	38%	50%	88%	64%
										UP	75%	100%	75%	38%	50%	50%	74%

We have also provided a plugin to import the Product Feature Matrix Template into the variability management tool pure::variants. This avoids manually porting each feature to the variability management tool. A synchronization between template and the tool is currently not possible. As a consequence the master of the variability information needs carefully decided.

2.1.6.2 Variant Analysis

Variant Analysis is an approach and tooling to identify commonality and variability in the engineering artifacts (in particular, textual artifacts) of existing system variants in an efficient and effective way. It compares several artefact variants in parallel and supports an interactive commonality-variability-analysis on multiple levels of abstraction. In this way, it enables the user to assess the reuse potential in existing artefacts and decide on the optimal strategy for introducing variation management approaches.



Variant Analysis uses a hierarchical set similarity model which enables very efficient similarity analysis and supports easily understandable result presentation. Compared to other similarity analysis approaches, Variant Analysis is particularly efficient for large software systems and a high number of analyzed software variants. In the project Variant Analysis has been extended to be support also the analysis of models. SimuLink has been used as the modeling language to this end.

2.2 Ressourcen

Das geplante Budget und die geplanten Aufwände Partner liegen im geplanten Bereich und es gibt keine nennenswerten Abweichungen. Die folgende Tabelle zeigt eine Aufschlüsselung der Personal-, Reise- und sonstige Kosten:

Kostenart	Ist [€]	Plan [€]
Personal	1.196.519,92	1.178.798,00
Reisen	27.204,35	39.000,00
Sonstige	19.342,65	19.212,00
Total	***1.243.066,925***	***1.237.010,00***

Die im Projekt geleisteten Arbeiten wurden mit folgender Ressourcenverteilung erbracht.

Thema	Ist [PM]	Plan [PM]
Interoperability Specification (IOS)	16,46	18,00
Heterogene Simulation	19,80	18,00
Model-based Requirements Engineering	20,44	21,00
Safety Engineering	19,26	18,00
Tool Integration	19,09	18,00
Variability Management	22,20	20,00
Projekt Management	1,00	1,00
Total	***118,25***	***114,00***

Die Arbeiten im Projekt erforderten einen deutlich höheren Implementierungsanteil und Implementierungsaufwand als ursprünglich geplant. Um dies mit den zur Verfügung stehenden Projektmitteln zu bewerkstelligen wurden die Implementierungsarbeiten nach Möglichkeit von den wissenschaftlichen Mitarbeitern an geeignete Hilfswissenschaftler übertragen, die dann durch wissenschaftliche Mitarbeiter angeleitet wurden. Dadurch verringert sich der erbrachte Aufwand bei den wiss. Mitarbeitern, während die erbrachte Arbeitsleistung tatsächlich über dem ursprünglichen Plan liegt. In Summe konnte so mit den verfügbaren Projektmitteln eine höhere Arbeitsleistung erbracht werden, als ursprünglich geplant war.

2.3 Notwendigkeit und Angemessenheit der Arbeiten

Die in den verschiedenen Themenschwerpunkten durchgeführten Arbeiten waren alle notwendig und führten wie oben geschildert zu konkreten Ergebnissen.

Insbesondere die enge Zusammenarbeit mit den verschiedenen Technologieanbietern und Anwendungspartnern hat sich bei der Erarbeitung der Methoden- und Werkzeugbausteine als sehr hilfreich erwiesen.

Wo erforderlich, wurden Aufgabenstellungen an den konkreten Bedarf der Use Cases angepasst.

2.4 Vorteile

Insgesamt kann festgestellt werden, dass CRYSTAL für Fraunhofer ein sehr erfolgreiches Projekt war. Die initial erhofften Ziele und Vorteile in den verschiedenen Schwerpunktthemen konnten voll erzielt werden. Nachfolgend werden zentrale Vorteile kurz zusammengefasst.

2.4.1 Heterogene Simulation (IESE)

Arbeiten an der Kopplung funktionaler Modelle haben beim Fraunhofer IESE bereits zu Aufträgen aus der Automobilbranche geführt. Man betrachtet neben der Kopplung weniger Modelle auch immer stärker Aspekte der Simulation vollständiger Fahrzeuge und der Automatisierung von simulationsbasierten Tests, die eine automatisierte Konfiguration und Kopplung von Simulatoren erfordern. Auch in der Domäne der Produktionssysteme werden Simulationsmodelle im Rahmen von Industrie 4.0 immer relevanter. Hier bilden die CRYSTAL-Ergebnisse zum Beispiel die Grundlage für die Simulation von virtualisierte Produktionsstraßen.

2.4.2 Model-based Requirements Engineering (IESE)

Mit den Arbeiten konnte gezeigt werden, wie die Formalisierung von Vorschriften basierend auf SysML gelingen kann. Die gewonnenen Erfahrungen helfen IESE ein modell-getriebenes Systems

Engineering extensiver einzuführen, in dem der Fokus vom System nun auch auf mitgeltende Dokumente ausgedehnt werden kann.

2.4.3 Safety Engineering (IESE)

Durch die Evaluierung der entwickelten Safety Engineering Methoden und Werkzeuge in diversen Applikationskontexten konnte eine domänenübergreifende Version des integrierten Meta-Modells sowie eine robustere Tool-Unterstützung (weiter) entwickelt werden. Auf längere Sicht kann das Meta-Modell die Grundlage für die Definition einer OSLC-Domäne für Safety bilden. Dies könnte den Datenaustausch zwischen Modellierungswerkzeugen bei der Entwicklung von sicherheitskritischen Systemen signifikant erleichtern. Dank des entwickelten integrierten Meta-Modells gelingt die Integration von Safety Artefakten und Analysen im Fraunhofer I-Safe-Werkzeug nun viel effizienter. Zusätzlich half die Evaluierung, das Werkzeug zu verbessern und die vorliegenden Beispielmodelle erleichtern die Vermarktung des Werkzeug-Prototypen.

2.4.4 Variability Management (IESE)

Die Arbeiten am System Family Engineering Framework (Scoping, Feature-Modellierung, EXConfig) und der Varianten Analyse konnten inzwischen bereits mehrfach in Kooperationsprojekten mit der Industrie erfolgreich eingesetzt und transferiert werden. Bei im Projekt erzeugten Beispiele erweisen sich hier als sehr hilfreiche Referenzen. Zu der notwendigen Generalisierung der Variantenmanagement-Ansätze in Software und Hardwarediszipliner konnten durch das Projekt wichtige erste Beiträge geleistet werden.

2.5 Publikationen

Folgende Veröffentlichungen der Projektergebnisse sind bereits erfolgt:

- Kuhn Thomas, „Frühzeitiges Absichern von Entwurfsentscheidungen durch Virtuelle Deployments“, in Proceedings of Informatik 2014.
- Käßmeyer Michael, Velasco Santiago, Schurius Markus, “Evaluation of a systematic approach in variant management for safety-critical systems development”, Embedded and Ubiquitous Computing conference, Porto – Portugal, 21.10.2015
- Duszynski, Slawomir. Analyzing Similarity of Cloned Software Variants Using Hierarchical Set Models. Dissertation am FB Informatik der TU Kaiserslautern, Fraunhofer IRB Verlag, 2015.
- Bo Zhang, Martin Becker, “Supporting Product Configuration in Application Engineering Using EXConfig”, In Proceedings of the 20th International Software Product Line Conference (SPLC 2016)
- Bo Zhang, Slawomir Duszynski, and Martin Becker, “Variability Mechanisms and Lessons Learned in Practice”, in Proceedings of the 1st International Workshop on Variability and Complexity in Software Design (VACE 2016)