 THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.

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

Specification, Development and Assessment for AUTOSAR Tools & Components

D605.011

DOCUMENT INFORMATION

Project	CRYSTAL
Grant Agreement No.	ARTEMIS-2012-1-332830
Deliverable Title	Specification, Development and Assessment for AUTOSAR Tools & Components
Deliverable No.	D605.011
Dissemination Level	CO
Nature	R
Document Version	V1.00
Date	2014-03-31
Contact	Christian Reinisch
Organization	TTTech
Phone	+43 1 585 34 34 - 4207
E-Mail	christian.reinisch@tttech.com

AUTHORS TABLE

Name	Company	E-Mail
Christian Reinisch	TTTech	christian.reinisch@tttech.com
Daniel Sauter	ITKE	daniel.sauter@itk-engineering.de
Aleksander Lodwich	ITKE	aleksander.lodwich@itk-engineering.de
Cecilia Ekelin	VOLVO	cecilia.ekelin@volvo.com
Johan Ekberg	ARCC	johan.ekberg@arccore.com
Irfan Delbassez	ELEKTROBIT	irfan.delbassez@elektrobit.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
0.1	25.11.2013	Deliverable structure	All
0.5	23.12.2013	First inputs collected	All
0.6	17.01.2014	Further inputs received	All
0.8	03.02.2014	First preliminary document version	All
0.9	03.03.2014	Iterative refinements integrated	All
0.95	11.03.2014	Internal review version	All
0.99	21.03.2014	External review version	All
1.00	28.03.2014	Final version	All

CONTENT

Content of Figures

Content of Tables

1 Introduction

The main purpose of the deliverable is the documentation of the individual bricks developed in WP605.

1.1 Role of deliverable

This deliverable documents the Bricks that are developed in WP605. This deliverable is updated iteratively, i.e. three times during the project runtime, based on the corresponding milestones of the project. Therefore the Brick documentations in this document represent an evolutionary process of continuous development and enhancement of the CRYSTAL solutions.
The brick descriptions will thus be refined and enriched iteratively (e.g. the descriptions will be extended or chapters will be added) over the whole project duration and as such these follow-up deliverables will be complementary to any previous version of the document.

The bricks of WP605 represent methodology, software and hardware contributions that are developed by one partner or jointly together in a cross-partner collaboration. The bricks described in this document are:

- **Task 6.5.1:** Model based software development
- **Task 6.5.2:** AUTOSAR/EAST-ADL Interface
- **Task 6.5.3:** DSpace TargetLink
- **Task 6.5.4:** BSW builder
- **Task 6.5.5:** RTE builder
- **Task 6.5.6:** SWC builder
- **Task 6.5.7:** ARTOP
- **Task 6.5.8:** TTEthernet Design & Development Tools
- **Task 6.5.9:** SoS Infrastructure Middleware Bricks
- **Task 6.5.10:** SoS Infrastructure Wireless Interface
- **Task 6.5.11:** Changes to tresosStudio

The document reflects the state-of-the-art of the areas addressed by the technology bricks. In this first iteration, it describes the current results of the requirements elicitation phase of both, use cases and bricks. However, due to the ongoing work in use cases and also in research and technology that is driving the individual bricks, these requirements are open to be updated in the future.

This document thus has to be seen as a ‘living document’ where the contents are expected to evolve and possibly also change over time. These amendments will be captured and explained in the subsequent versions of this deliverable.

The scope of this deliverable is limited in terms of (not) explaining in detail the envisaged use and demonstration of the bricks in the use cases. For this information, we refer to the deliverables of

the use cases which will explicitly include this information in their descriptions. Please refer to Section 1.3 for an overview of bricks — use case relationships.

1.2 Relationship to other CRYSTAL Documents

This document has relationships to the following documents:

- The deliverables of the use cases where demonstrations of the WP605 bricks are planned.
- The follow-up versions of D605.011, i.e. deliverables D605.021 and D605.031.

1.3 Relationship between bricks and use cases

Table 1 shows the relationship between the tasks of WP605 and the use cases. For one task, the demonstration in a particular use case is still open to be defined. This task is marked with TBD. It is one goal of this document to announce and advertise these bricks to the CRYSTAL consortium and to encourage their uptake towards demonstration scenarios in interested use cases.

Note: Since WP605 focuses on the automotive domain, a demonstration of the bricks is most likely in the use cases of the automotive subprojects, i.e. UCs 3.x. Although not captured by WP605, an uptake of selected bricks in additional use cases shall not be excluded and is encouraged where appropriate.

	UC 3.1	UC 3.2	UC 3.3	UC 3.4	UC 3.5	UC 3.6	UC 3.7	UC 3.8	TBD
Task 6.5.1									X
Task 6.5.2	X								
Task 6.5.3				X					
Task 6.5.4	X								
Task 6.5.5	X								
Task 6.5.6	X								
Task 6.5.7	X								
Task 6.5.8				X					
Task 6.5.9				X					
Task 6.5.10				X					
Task 6.5.11						X			

Table 1: Relations between WP605 technology bricks and use cases

X … Technology Brick (Task) will be demonstrated in the Use Case(s)

1.4 Structure of this document

This document is structured in four main sections. Section 1 provides an introduction to the workpackage and outlines the relevance of its contents in the CRYSTAL project. It also provides a dedicated subsection 1.3 that highlights the relationships between the CRYSTAL use cases and the technology bricks. In Section 2, the different technology bricks of this workpackage are described in detail. Finally, in Sections 3 and 4 abbreviation and references are listed.

2 Technology bricks

In each section of this Chapter 2, one brick of workpackage 605 is described. These bricks correspond to the tasks identified in the description of work and thus include developments made in course of the CRYSTAL project. All subchapters follow the same structure which is based on the recommended outline as defined by SP6.

2.1 Model based software development

2.1.1 Description

Name:	Model based software development
Contact:	ITKE, Aleksander Lodwich, Aleksander.Lodwich@itk-engineering.de
Dependencies	n/a
License	n/a
Additional information	n/a

2.1.2 Manual

2.1.2.1 Motivation behind AUTOSAR

The development of computerized vehicle systems means a significant investment into the vertical systems architecture which must be protected as good as possible against invalidation while underlying electronic platforms evolve (ECU consolidation, energy consumption optimization, network layout optimization, platform cost optimization, etc.). In order to achieve this, a high level of reuse of value code must be intended and it seems possible because experience has shown that effects of platform changes can be absorbed by an intelligent middle-ware. As a consequence, the separation of the logical architecture and electronic platform is becoming more important. However, an OEM-specific solution is unattractive. Computerized components are supplied from different vendors and proper software mobility is only attained if branding barriers are avoided. This entails the establishment of a standardized set of concepts, agreed upon technical details and a readily available tool-chain to provide really exchangeable pieces of software. AUTOSAR was designed with these requirements in mind.
AUTOSAR (Automotive Open System Architecture) offers a data-flow oriented architecture approach which encapsulates greater real-time capable pieces of functionality in transportable SWCs (software components). These components can be assigned to different executing environments and can be replicated if necessary. The modularized approach to build systems from components improves collaboration between OEMs and TIER-X suppliers and harnesses some advantages of the object-oriented design paradigm (Data Capsulation).

The content of SWCs can be developed in many ways but the dominating source are behavioral models. Because the importance of traditional source code is diminished by autocoding, it is the functional models and their meta-models which must be reused. If AUTOSAR processes and tooling are setup properly then reuse of these models can be achieved.

2.1.2.2 Identifying Problems with Current State-of-the-Art

In the following discussion of AUTOSAR concepts and typical ways to work with it, some weaknesses of AUTOSAR shall be elaborated in order to understand further potentials of AUTOSAR employment by making advances in development tool integration.
From the perspective of model-based engineering the interest is focusing on AUTOSAR software components (SWCs). They contain the product specific logic. The software in the SWC is not in itself runnable. In order to run, it requires additional support from RTE (Real-Time Environment), BSW (Basic Software Layer) and a real-time OS.
The adaptation of the SWC to the individual ECU is achieved with an automatically generated RTE, which is a key advantage of using AUTOSAR. The RTE coordinates the SWCs by providing a plug/wallet analogy. An embedded system can be easily rewired this way and it is easier to provide various configurations. As long as proper wallets (ports) exist in the RTE, SWCs can be located anywhere, at least in theory. This flexibility bears the potential for new business models (vehicle ad-hoc functionality) but such topics are currently something for the future.
AUTOSAR is also concerned with defining device drivers and resources in a standardized way but these aspects of the BSW are of little relevance to model-based software development. The purpose of AUTOSAR is to minimize the influence of individual platform details on the design of dynamic functions by such means as generated interfaces, drivers and abstracted hardware configuration.
In practice, AUTOSAR fails to deliver the promises for various reasons:

a) Development tools do not properly support the AUTOSAR life-cycle
(internal engineering cycles become severely elongated)
b) The vertical position of AUTOSAR-based software isolation is still too low in context of the overall system architecture
(too many platform details penetrate middle-ground and hit SWCs)
c) Costs of hardware still dominate development costs of software.
(Platforms do not really afford middle-ground overhead)

The brick will concentrate on fixing problems marked as a) and b).

2.1.2.2.1 Problem a

The point describes the fact that most tool-chains do not properly support the necessary flow of information in an AUTOSAR project. A typical flow of required information is depicted in Figure 1. The figurative workflow is abstract.

The idealistic workflow assumes that interfaces of software components will be defined a priori by the act of a vehicle's E/E architect. In this case model developers merely design by contract. An

additional assumption is that testing the integrated models can happen late on the ECU. The justification behind this is that exact signal specifications and signal simulations are representative of the designed system's environment. All these assumptions are often not true for real-life development processes.

Much more the following seems to be true: It is the functions designer and not the architect who defines the necessary communication between SWCs and this role is often an expert on peculiar product operation characteristics. In this context, model development is often rather a continuous process than a finite sequence of activities included in such engineering paradigms as the V-model (or the document dependency tree from Figure 1). The development paradigms and artefact flow schemas are just means for communicating responsibilities and implicitly allocating activities to different parties. The specification of required information is the output of the iterative model improvement where the true source of information is not corresponding to the document flow or process scheme in use.

As a consequence, the elaboration of available RTE interfaces will require far reaching interactions between project stakeholders which consume more time than deemed tolerable for function design cycles. This also adds costs because managing the connectivity on the AUTOSAR device extends the efforts to manage connectivity between ECUs. For example, what would be amendable is a bazaar concept for component developers. If developers can quickly provide ports and detect undesired dependencies on their outports then development cycles could become shorter. However, such a bazaar concept will require tool support not yet available. IOS can help to create such bazaars for interface information and to verify their feasibility across several different technologies of logic isolation by supporting taxonomy- or interval-based matching of interface parameters which are exposed by several instances of involved tools even before they enter a central point of reference. Designing such interfaces and setup of such bazaars can become a matter of this brick.

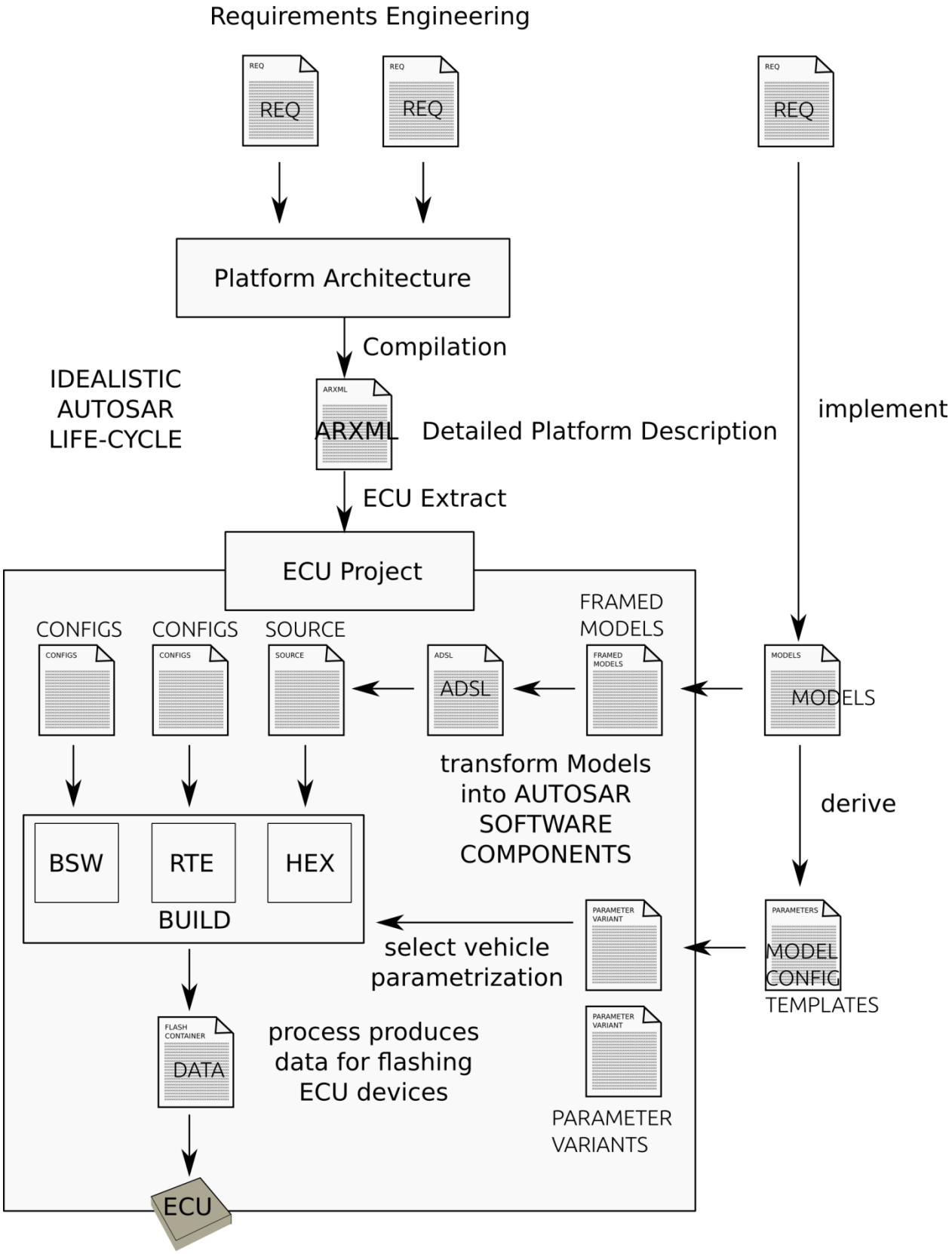

Figure 1: Idealistic workflow of AUTOSAR devices without design cycles

AUTOSAR also introduces shared responsibilities on the path of information flow which are not always clear how to treat. The source of the problem is the high abstraction level of the model code. The model developers work in highly specialized model designing environments and are generally interested in interface specifics only in a very generic way. For example, a model may require a velocity for input as a physical quantity in km/h. Unless the designer is forced to be concerned, this is the level of abstraction that he needs to get his job done. If the solution is distributed among different ECU devices then the developer has to start to worry about network technology, about which signals can be provided and which can be consumed. This introduces a new set of requirements to the model. On the input side the model developer must be able to express a minimum requirement of signal rate, resolution or securitization. On the output side the model developer must be able to respect the strictest requirements for the emitted signal. Achieving all this is a shared responsibility which causes unnecessary development cycles in real projects.

Even though many system vendors distribute their efforts among model departments and integration departments in order to handle the obvious need to handle projects at different levels of abstraction the problem is not solved. Integration departments often do not have the knowledge to adapt abstract models to network signal requirements correctly. Even if developers help to specify some signal properties, still not all signal properties can be contributed by them. Because of this, the documents governing the full interface specification are not accessible for writing to developers. This entails slow information integration processes from responsible staff. Developers often simply e-mail their requirements and contributions. These processes are sometimes designed ad-hoc and lack good tool support. Companies either attack this problem by adding new tools (e.g. PreeVision) or by writing in-house tools to cope with the problem. In both cases the tools tend to be isolated. IOS can help to overcome these problems if developers can gain access to relevant objects and their features through expert views which can access and manipulate relevant objects in great detail.

Additional problems of similar kind arise if bus technology is introduced to AUTOSAR projects. In context of systems producing emergent behaviors it is not efficient to consider parameters of the Virtual Functional Bus (VFB) and the corresponding networking technology in isolation. There are many parameters for the RTE ports which have to be defined and there are many bus-specific parameters which have to be compatible. It is also not practical to maintain different variants of Excel sheets in order to model all the signal flow constellations and the involved parameters. The tools used to configure the interface layers are created in disjoint contexts and at different points of history but they provide important pieces of information. If IOS helps to create and operate views for consistent specification of complete communication paths despite distributed databases then the work of developers would become greatly simplified and chances of error would be minimized.

Another problem related to weak tool-chains is testing of AUTOSAR systems because far reaching readiness of the whole system is required. An ECU will probably not run properly if the RTE is not fully configured and most or all SWCs have to be in place, compiled and ready for execution. Developers who want to test their AUTOSAR component in some intermediate state need to find an individual solution. As the whole environment evolves they find themselves designing and maintaining different kinds of auxiliary frameworks for testing which have to be maintained to actual ECU AUTOSAR architecture. A great increase of work efficiency could be achieved if the tool-chains allowed co-simulation of complete ECUs with transparent support of AUTOSAR components at different readiness levels. What is being looked for is very similar to Vector's

concept of Network Remainder Simulation but for AUTOSAR. This way maintenance of redundant intermediate AUTOSAR frameworks could be avoided (cf. Figure 2).

2.1.2.2.2 Problem b

This point is concerned with the difficulty to re-use AUTOSAR components in follow-up projects. The intended level of re-use for AUTOSAR components is still binary. At this level it is very difficult to handle the observed E/E platform variety even if the RTE can be rewired or drivers can be exchanged. However, from the point of the product developer it is not the AUTOSAR component which is valuable but its logical content.

Therefore, the effects of this problem could be reduced if the effort to produce new project-specific model-source could be based on meta-models. Tools for handling variability and high level modeling tools are involved in such setup. If properly interoperable, some AUTOSAR ECU design could be done prior to fully developing AUTOSAR components. Such a grave modification of AUTOSAR workflow will require a tool-chain like the one described above which can manage the communication boundary allocation for the whole system and support a consequent aspect-based design. Possible scenarios would be involving tools like pure::variants, Simulink, Enterprise Architect (SysML) or arKItect Developer.

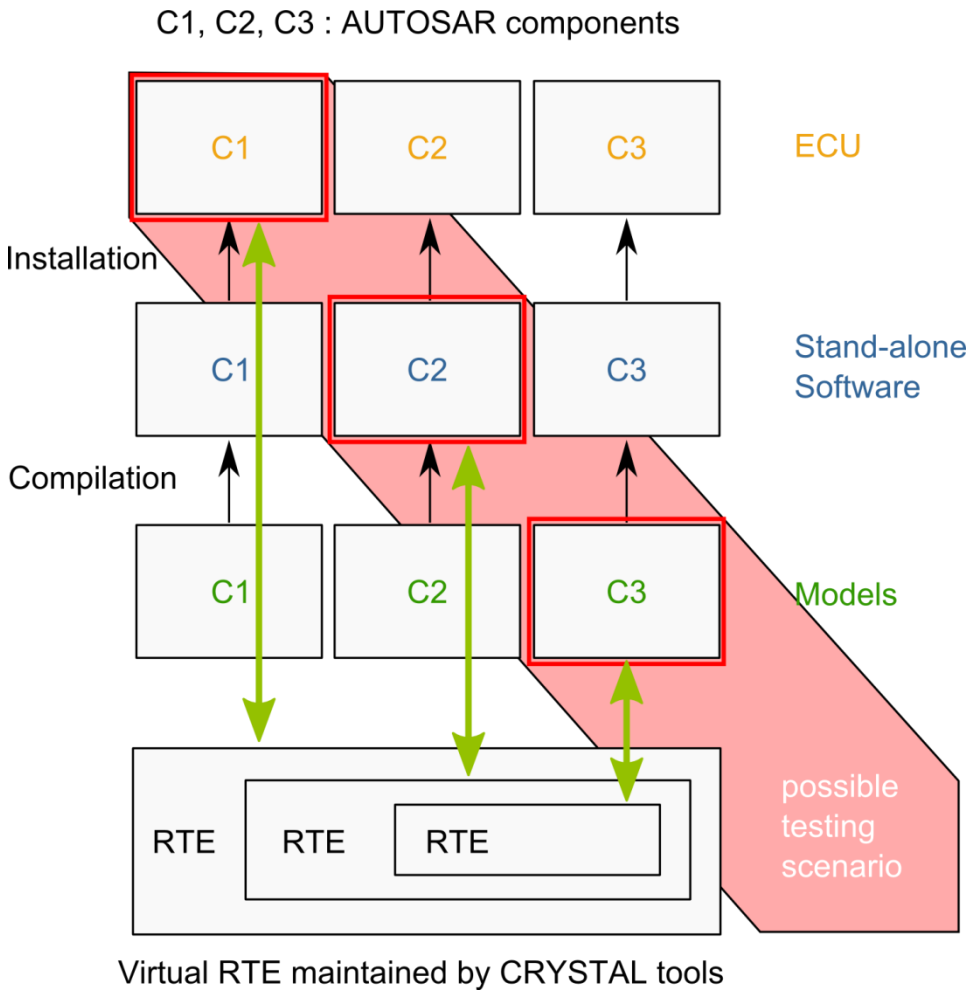

Figure 2: A jointly managed virtual environment for AUTOSAR components can greatly simplify testing.

2.1.2.2.3 Problem c

Protection of investment into SWCs could be achieved by smarter middle-ware (or rich middle-ware). Rich middle-ware is capable of live adaptation of pieces of software (late adaptation) and it allows for better re-use of AUTOSAR components even with contemporary tool-chains. However, rich middle-ware also takes a higher toll on system resources. TFor economic reasons, rich middle-ware is not feasible for many AUTOSAR projects because the required additional performance is considered too expensive if the hardware is purchased by the millions. The RTE offers some functionality of this kind like the ability to perform linear transformations for ports with different value representations. More complex transformations can be imagined including semantic reasoning. However, in projects where costs of software engineering are more substantial rich middle-ware seems to be a possible contribution to SWC re-use. In the worst case such a rich middle-ware could be a virtual machine which simulates a different ECU. This example shows that rich middle-ware will often involve additional levels of expensive functional abstraction. However, the question how to design and handle such additional abstractions and the resulting additional interfaces with contemporary tooling is out of scope of this brick.

2.1.3 Use Case coverage and application

In general, the work conducted in this brick is considered to highlight the general benefits of the CRYSTAL approach and to adopt these benefits to AUTOSAR developments. The brick is therefore offered to all interested use cases, in particular from the automotive domain.

Specific use cases for demonstration of selected parts of this work will be identified in the next phase of the project. Then, also more concrete statements about addressing use case specific needs will be made.

2.1.4 General Improvement

The definition of Technical Items (TI) will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.1.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.2 AUTOSAR/EAST-ADL interface

2.2.1 Description

Name:	AUTOSAR/EAST-ADL interface
Contact:	VOLVO, Cecilia Ekelin, cecilia.ekelin@volvo.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.2.2 Manual

This task will specify, develop and assess a partial common interface specification for tools dealing with AUTOSAR and EAST-ADL information.
AUTOSAR and EAST-ADL define two complementary and compatible metamodels for capturing design information for automotive embedded systems. They moreover define XML-based data exchange formats based on the metamodels. This brick will standardize the way AUTOSAR and EAST-ADL tools connect with the IOS, given the common metamodels on which their data is based on.

State-of-the-art:
An overview of the EAST-ADL model is shown in Figure 3.
Currently the data exchange between EAST-ADL and AUTOSAR is based on export and import of XML-files. This prevents the tool environment from providing a global, coherent view of the system to the developers. Data should instead be seamlessly linked and/or exchanged in order to maintain consistency. This requires common methods for providing links and data exchange. For EAST-ADL and AUTOSAR this specifically means that common links that connect the two metamodels need to be defined, e.g. linking of AUTOSAR software components to EAST-ADL design functions. These link definitions also form the basis for automated data exchange since they may indicate semantic equivalences between the metamodels. The links are however not enough since they do not define the semantics of the individual metamodels. These semantics are at the moment not necessarily specified clearly enough to allow a well-defined one-to-one model transformation. The semantics within a metamodel may even in some cases be in-complete (e.g. triggering) which is a major hurdle for tool developers and prevents modeling and/or analysis tools from being built. Thus, there is a need to review and possibly detail the semantics of both AUTOSAR and EAST-ADL, especially in their relationships.

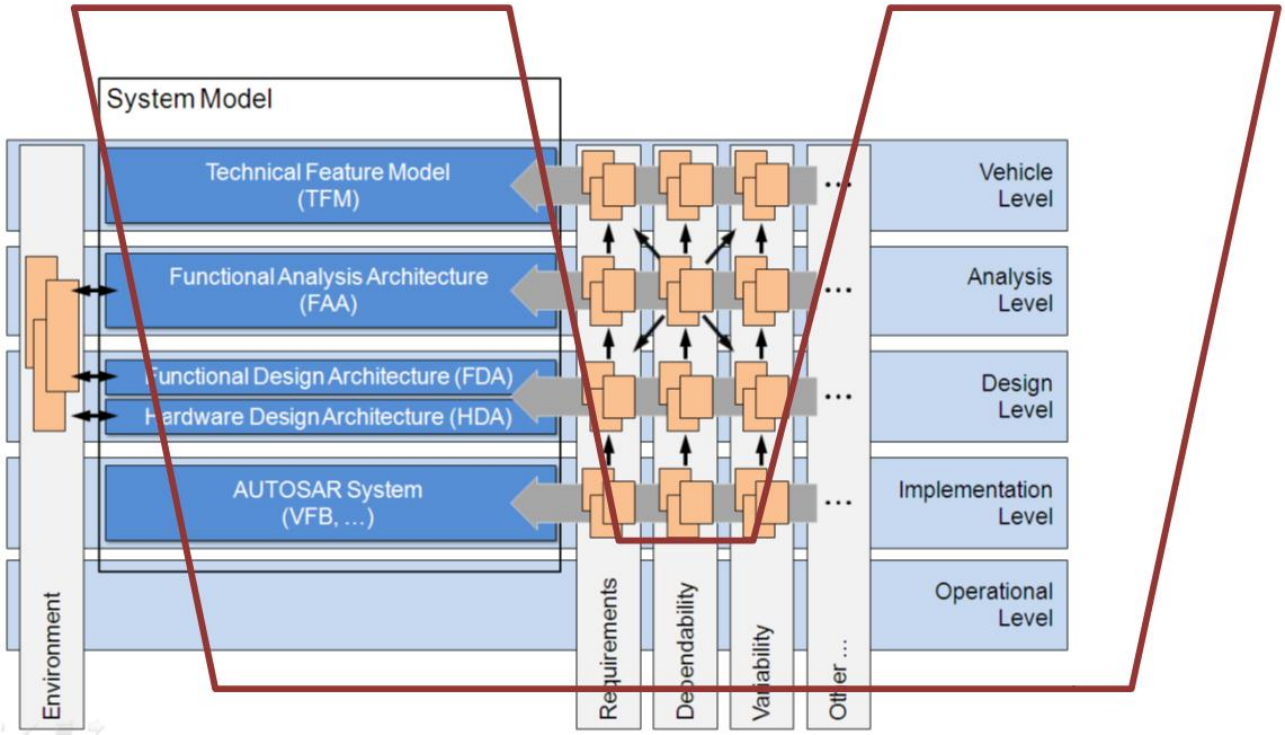

Figure 3: Applying modeling in V-Model. Example from the automotive domain [Chen, 2013]

Planned CRYSTAL work:

The mentioned shortcomings will be addressed in the implementation of UC 3.1. Since this use case contains data linking and exchange between EAST-ADL (like) models and AUTOSAR models, the steps necessary to carry out the use case will be implemented.

This includes:

1. establishing how to link EAST-ADL models with AUTOSAR models
2. establishing how to transform an EAST-ADL model to an AUTOSAR model (model generation)
3. identifying and fixing semantic weaknesses in the EAST-ADL and AUTOSAR metamodels.

Of course, due to resource limitations, a complete coverage of EAST-ADL and AUTOSAR will not be feasible. However, hopefully the partial results will represent absolute knowledge which makes them a building block for future work as well as for the IOS.

2.2.3 Use Case coverage and application

This brick is going to be demonstrated in use case 3.1. There, the approach developed in this brick will be used for function development of heavy vehicles in an integrated software engineering environment.

More specific use case needs will be identified in the next phase of the project.

2.2.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.2.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.3 Dspace TargetLink

2.3.1 Description

Name:	DSpace TargetLink
Contact:	ITKE, Aleksander Lodwich, Aleksander.Lodwich@itk-engineering.de
Dependencies	n/a
License	n/a
Additional information	n/a

2.3.2 Manual

TargetLink is a Simulink extension from dSpace which attempts to simplify the transition between concept phase and hardware implementation during a project. A special set of blocks allows to model dynamical and logical functions which can later run on dSpace hardware (such as MicroAutoBox). This approach allows for shortening design cycles for single components within partially developed systems significantly.

The blockset of TargetLink offers dialogs for additional parameters which are required to configure concrete communication channels and computation modes. This information can be iteratively refined in a development process until enough information is acquired for generating source code.

The approach taken by TargetLink is close to achieving aspect-based engineering but it is vendor oriented. TargetLink defines all important technical features as model properties and provides tool-based support for managing project related information according to responsibilities. Data dictionaries are used to transport produced and consumed information in a coordinated manner. This feature includes AUTOSAR relevant properties.

dSpace provides a set of tools attuned to each other. The challenge of this task is to develop strategies to manage workflow outside of the TargetLink toolset. For example new functionality is often introduced by developers who add parameters to Matlab workspace. As the project matures the information is transferred to dd-files or to DCM-files or both. Similar to use case 3.4a it is very important to have a way to treat the project data logically and to keep track of all reconciliations between the representations.

2.3.3 Use Case coverage and application

The brick shall be demonstrated within Use Case 3.4a Test case definition interlinked with model based requirement engineering / variant management.

There, it will address the following topics:

Model-related data should be represented by abstract objects and the tool-chain should provide means to retrieve specific representations of data according to local needs from them. In order to

achieve this, these abstract objects hold relationships to various data representations (files) and contain a history of reconciliation and transformation activities between them.

With additional knowledge of data formats, a logical comparison between the representations can be expressed as an abstract distance (estimate amount of inconsistency). This is a compromise between being able to generate files from a single generic database (difficult to achieve) and a loose collection of versioned files (status quo).

More specific use case needs will be identified in the next phase of the project.

2.3.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.3.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.4 BSW builder

2.4.1 Description

Name:	BSW builder
Contact:	ArcCore, Johan Ekberg, johan.ekberg@arccore.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.4.2 Manual

The BSW builder is the tool to edit the Autosar communication stacks and platform and to generate the BSW configuration files. The tool is built on the Autosar standard and modules are identified with their Autosar names. The tool configures the standard and vendor specific parameters that are possible to use in the BSW. The parameters that can be used to configure the BSW are dependent on the ECU configuration, the available BSW modules and the current capabilities in the Autosar version. The BSW editor displays the available capabilities to set up the ECU platform with a clean user interface.

The BSW Builder is a graphical development tool used to configure and generate Autosar basic software configurations. It is built as an extension to Arccore Arctic Studio [AS, n.d.] and is based on the Autosar methodology and Autosar XML exchange format. As the Arctic Studio is built up on Artop, the IOS plugins implemented in ARTOP can be used and extended into the BSW builder. Artop is further described in chapter 2.7.

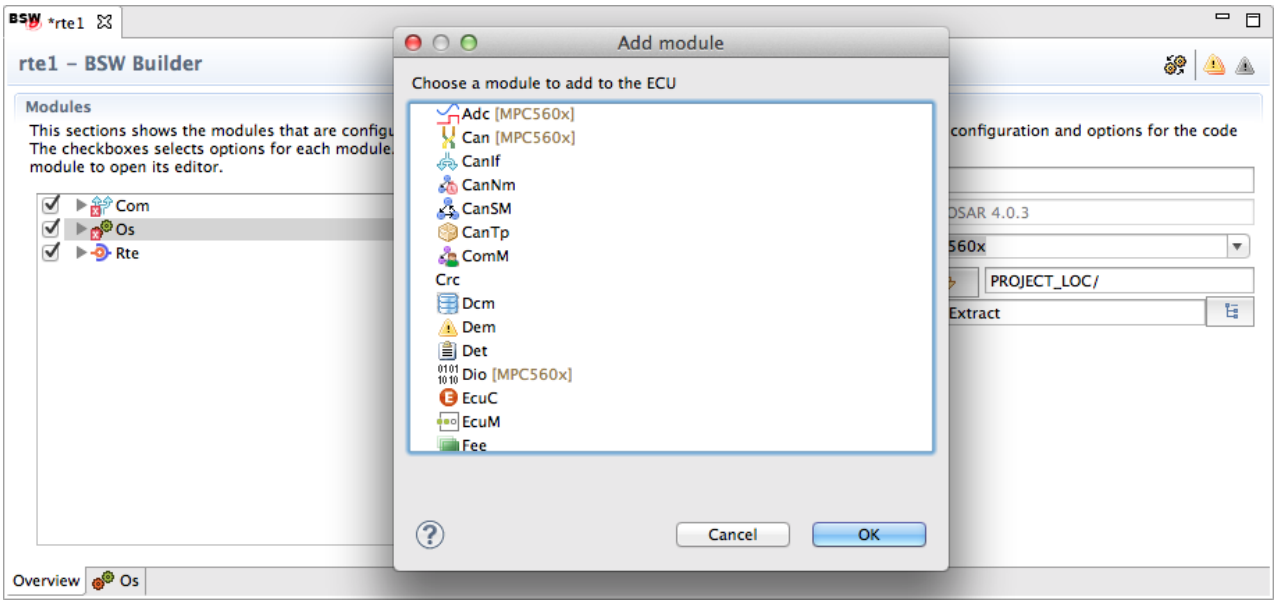

Figure 4: Screenshot of the BSW builder

Today most Autosar products are built as single user systems where the XML exchange format is the central part to exchange information between users and tools. To take the next step and let the tools support work in a multi-user environment, good algorithms for comparing and merging need to be developed. In addition it is essential that the user can get and share configured parameters with other tools to achieve a complete development system. For instance certain parameters can be specified already at the requirement level and should be available when starting to configure the BSW.

2.4.3 Use Case coverage and application

This brick will be integrated in UC3.1 Function development for heavy vehicles. Among other topics, it will address the linking between requirements and their use in BSW builder for BSW configuration.

Within CRYSTAL, BSW Builder will be adapted to get configuration parameters from other tools like requirements tools within the IOS infrastructure. In addition the BSW Builder will need to be extended with a good diff and merge technology so that changes of BSW parameters within the IOS infrastructure can be easily incorporated into new versions of configurations.

More specific use case needs will be identified in the next phase of the project.

2.4.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.4.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:					

2.5 RTE builder

2.5.1 Description

Name:	RTE builder
Contact:	ArcCore, Johan Ekberg, johan.ekberg@arccore.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.5.2 Manual

The RTE editor is used to instantiate Autosar SWC prototypes on an ECU and to generate the Autosar Runtime Environment which glues the SWC prototypes together with the BSW platform. It is also used to map runnables, which are executable entities, to events. With the configured information the RTE editor will set up the run configuration and priority of the complete ECU software.

The RTE Builder is a code generator and graphical configuration tool for the Autosar RTE. It is built as an extension to Arccore's Arctic Studio Tool Suite and is based on the Autosar methodology and Autosar XML exchange format.

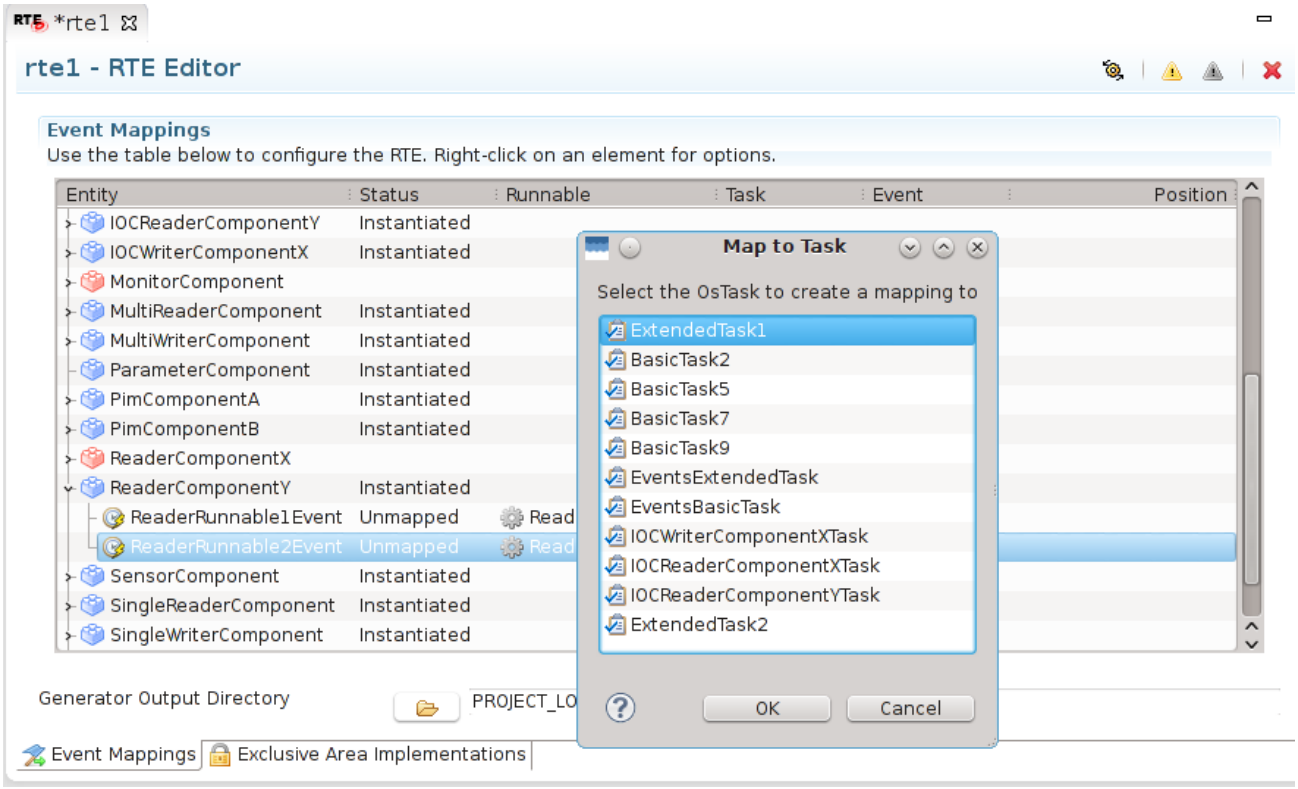

Figure 5: Screenshot of the RTE Editor

The user of the RTE Builder will get new configurations from multiple up-stream tools during a project. It is very important for the user of RTE Builder to understand what changes have been made by the up-streams tools before incorporating the changes into the RTE Builder configuration. The RTE Builder therefore needs tools to perform comparing and merging.

2.5.3 Use Case coverage and application

This brick will be integrated in UC3.1 Function development for heavy vehicles. Among other topics, it will address the linking between requirement tools and/or system design and communication tools with RTEBuilder.

Within CRYSTAL RTE Builder will be adapted so an ECU-Extract can be imported from other tools within the partnership. For example this can be delivered by requirements tools and/or system design and communication tools. Investigations and prototype implementations of compare algorithms and GUI visualization of differences will also be addressed.

More specific use case needs will be identified in the next phase of the project.

2.5.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.5.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.6 SWC builder

2.6.1 Description

Name:	SWC builder
Contact:	ArcCore, Johan Ekberg, johan.ekberg@arccore.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.6.2 Manual

The SWC Builder is a solution for creating Autosar system models consisting of software components, compositions and behavior. It is built on the Artext textual language from the Artop open source project [ARTOP, n.d.]. As an alternative to graphical representations commonly used in modeling software a textual language is used because textual languages have proven to scale successfully to large projects. They are also well suited for distributed development environments, as they can fully take advantage of the well-established, mature tools for textual formats (e.g.source, version and change control). Most importantly, a textual representation is easily readable, and understandable, which improves collaboration in the AUTOSAR model development.

SWC Builder is an Autosar authoring tool used to describe software components and their relations. It is built as an extension to Arccore's Arctic Studio Tool Suite and is based on the Autosar methodology and Autosar XML exchange format.

To work in collaboration with others within the scope of SWC Builder it is important for the user that imports new configurations to locate differences in the software component description and also to be able to view new connections between components.

2.6.3 Use Case coverage and application

This brick will be integrated in UC3.1 "Function development for heavy vehicles". Among other topics, it will allow tracing of software components specifications in comparison to earlier versions.

Within CRYSTAL, SWC Builder will be adapted to the IOS infrastructure so it will be possible to fetch new specifications for software components, compare them with earlier versions and view the differences detected. It should also be possible to publish software component information to tools further down the tool-chain of the Autosar methodology.

More specific use case needs will be identified in the next phase of the project.

2.6.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.6.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.7 ARTOP

2.7.1 Description

Name:	ARTOP
Contact:	ArcCore, Johan Ekberg, johan.ekberg@arccore.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.7.2 Manual

Artop is the leading AUTOSAR framework built for the eclipse environment. It is built on top of the eclipse EMF Framework [EMF, 2014] and adapted to Autosar within the Artop Open Source community [ARTOP, n.d.].

Today many of the OEMs, Tier1, Tier2 and tools suppliers use Artop to build their Autosar tool chain. This makes the framework a cornerstone in future tool-chain development projects for Autosar.

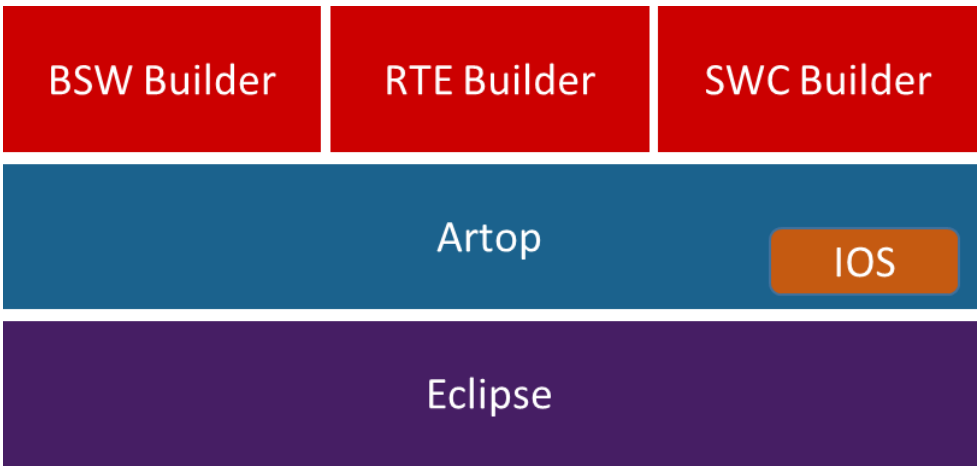

Figure 6: Artop tool Hierarchy

Artop uses the Autosar XML exchange format to let users exchange information within a project. This makes the framework, when used today, a single user framework where files are exchanged and manually merged by the user instead of leveraging on multi-users/multi-tools scenarios as specified in CRYSTAL.

2.7.3 Use Case coverage and application

During the CRYSTAL project the goal is to adapt the Artop framework so it can be used in the IOS infrastructure. The IOS support shall be added to support a framework to import higher level requirements into the tools built upon Artop, and to be able to publish information to tools that are used in other products in the development life cycle. This will give current users of Artop the possibility to also be part of the project results achieved within CRYSTAL.

More specific use case needs will be identified in the next phase of the project.

2.7.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.7.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.8 TTEthernet Design & Development Tools

2.8.1 Description

Name:	TTEthernet Design & Development Tools
Contact:	TTTech, Christian Reinisch, christian.reinisch@tttech.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.8.2 Manual

TTEthernet (SAE AS6802) [TTE, 2011] is a scalable, open real-time Ethernet platform used for safety-related applications primarily in transportation industries and industrial automation. TTEthernet extends classic Ethernet functionalities to provide more flexibility, modularity and scalability in Ethernet-based systems. It is compatible to IEEE 802.3 Ethernet and integrates transparently with Ethernet network components.

TTEthernet based networks enable the seamless communication of all kinds of applications via Ethernet. Conventional PCs, web and office devices, multimedia systems, real-time systems and safety-critical systems are to use the same network. One single network that is completely compatible with the IEEE Ethernet 802.3 standards is suited for data transmission among different applications with various requirements, e.g. satisfying different criticality requirements and fail-safe or even fail-operational behavior. Figure 7 gives an overview of the different communication types of TTEthernet. For CRYSTAL, in particular the time-triggered traffic is of most relevance since it supports the requirements of the automotive domain best.

Time-triggered (TT) traffic has two important pre-requisites: the need for a global notion of time in the network and the availability of schedules that organize the communication in the time domain, i.e. providing time partitioning on the network. For these reasons, switches in TTEthernet take over the central role of organizing the data communication. TT messages are routed in the switch according to a predefined schedule with as little delay as possible. Precise planning at the time of system design precludes resource conflicts at runtime. TT messages have the highest priority level. If the planned transmission time of one of these messages arrives, this message is immediately transmitted. Due to the predefined transmission of the message the switch ensures that the medium is free at the time of transmission and delays are precluded.

Schedules in TTEthernet are generated using a dedicated tool chain, where each tool solves a particular task of the configuration. The overall TTEthernet configuration tool chain is depicted in Figure 9. It consists of the following main parts:

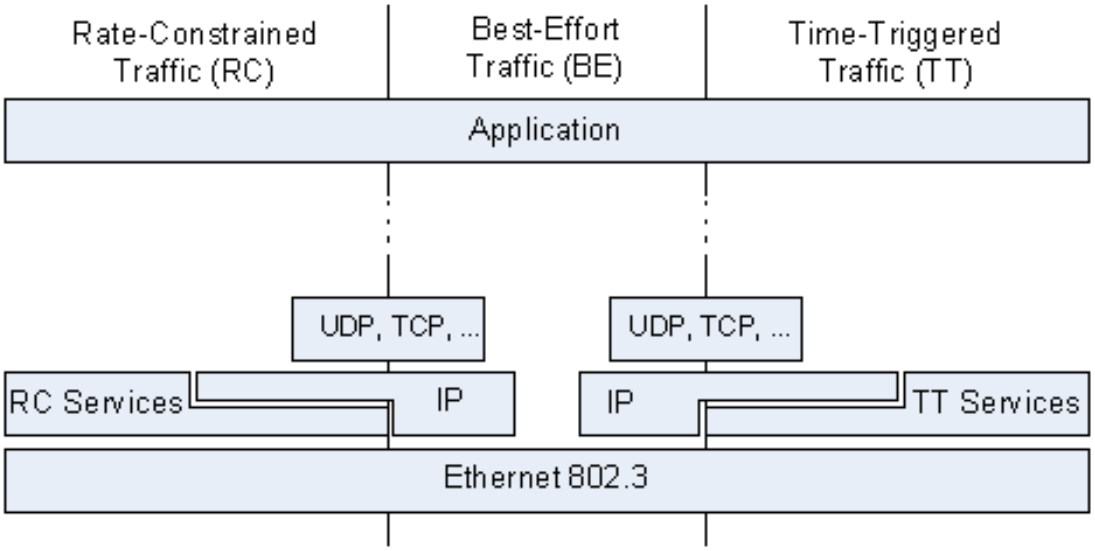

Figure 7: TTEthernet traffic types and relation to other protocols

- **TTEPlan:** TTEPlan is the TTEthernet network planning tool. Based on input provided to the tool, TTEPlan creates the whole network configuration databases.
- **TTEBuild:** TTEBuild allows converting XML-based device configuration database files into binary configuration images required by the TTE Switches and the TTE End Systems.
- **TTELoad:** TTELoad is an application suitable to configure a TTE Switch based on TTEthernet switch IP that also supports bootstrap configurations of TTE Switches.
- **TTEView:** This TTEthernet frame dissector for Wireshark[1] 1.x is a plug-in to Wireshark which supports the recording and analysis of over 300 Ethernet and internet protocols including TTEthernet.

An overview of this tool chain showing input and output files is shown in Figure 8Figure 9. TTEPlan can be used to configure a network from scratch, or to migrate an existing configuration to a network description file. The configuration output of the tool chain is a schedule that can be downloaded or otherwise communicated to the TTEthernet network components. It defines the time-slots during which communication on the network will occur including a separation along the different communication types co-existing in the network. An example configuration output is shown in Figure 8.

[1] http://www.wireshark.org

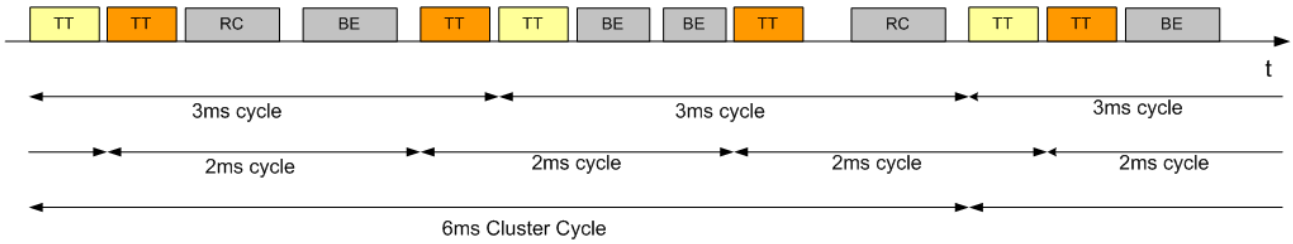

Figure 8: Example schedule as output of the TTE tool chain

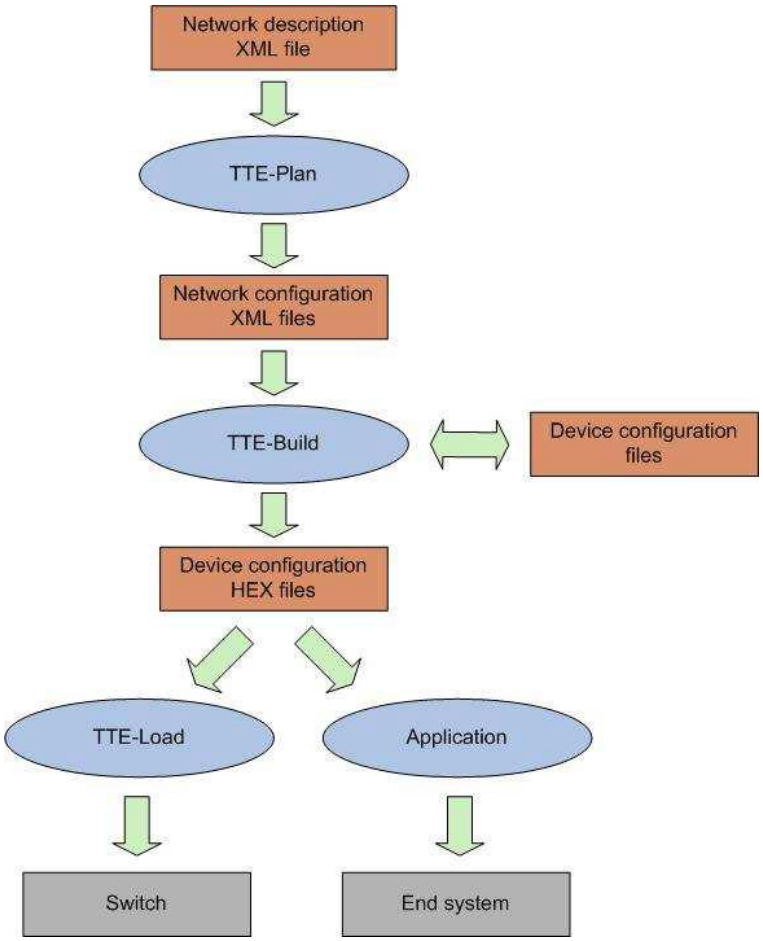

Figure 9: TTEthernet configuration toolchain

2.8.3 Use Case coverage and application

Within the CRYSTAL project it is the goal to extend the tool chain and relevant single tools to also allow for the configuration of larger engineering environments such as System-of-systems scenarios. The design and development tools of TTEthernet shall enable an SoS platform configuration that, for example, follows a time-triggered architecture approach supported by TTEthernet.

The developments of this brick are closely related to the other bricks of TTTech that target the development of SoS infrastructure parts. The brick is going to be demonstrated in use case 3.4 “Test case definition interlinked with model based requirement engineering. / Variant management integration”. There it shall support the configuration of the SoS platform that will be developed to demonstrate the challenges of the work described in Chapters 0 and 0 of this document.

More specific use case needs will be identified in the next phase of the project.

2.8.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.8.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.9 SoS Infrastructure Middleware Bricks

2.9.1 Description

Name:	SoS Infrastructure Middleware Bricks
Contact:	TTTech, Christian Reinisch, christian.reinisch@tttech.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.9.2 Manual

The main content of this bricks is the design and development of a middleware for the flexibly configurable System-of-Systems platform. The middleware is a set of software components that is needed to enable the data communication on the SoS platform and to also support communication to the 'outside world', using for example wired or wireless links as described in Chapter 2.10 of this document. Particular emphasis will be put on the consideration of safety-related requirements, so that the developed software bricks ensure the dependable communication that is required by various use cases such as UC 3.4.

Issues that need to be addressed in the software are, among others, the full support of mixed-criticality applications to be executed on the heterogeneous platform, thereby ensuring freedom-from-interference and other important safety factors such as timing control and failure mitigation. Figure 10 shows a possible concept of the SoS platform for which the middleware software components will be developed. It can be seen that safety-critical and non-safety critical system parts shall co-exist on a single platform, which demands specific protection mechanisms to be put in place.

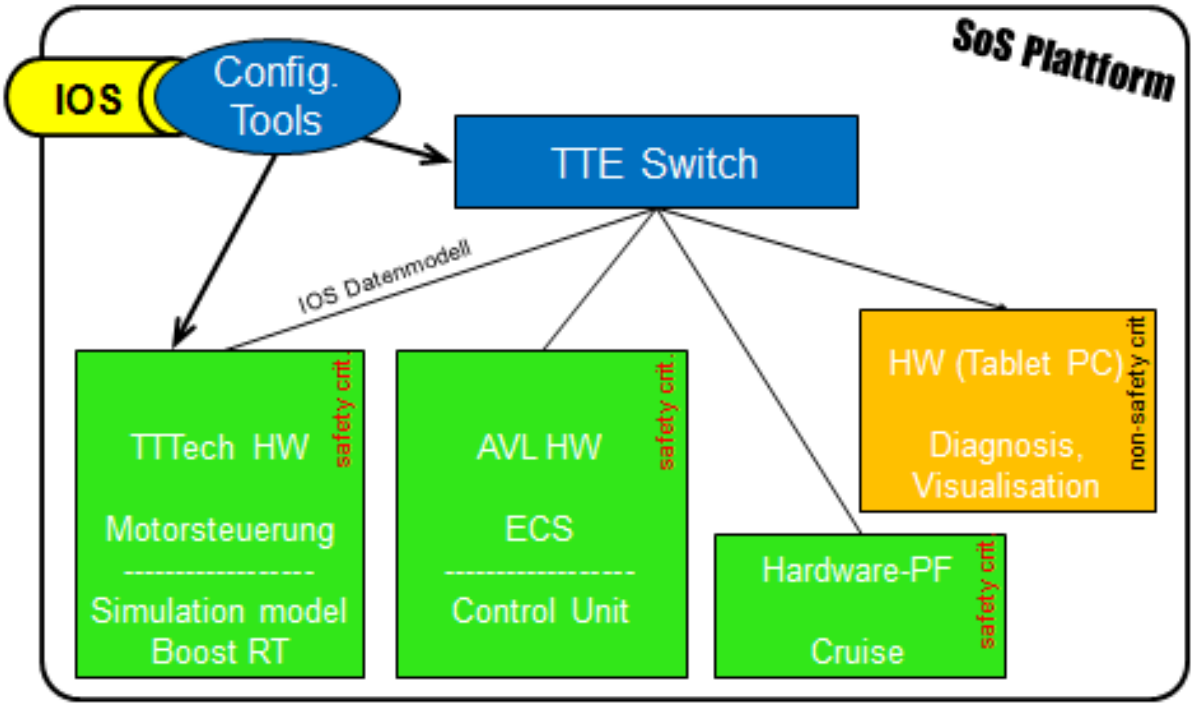

Figure 10: Concept of the versatile SoS platform

2.9.3 Use Case coverage and application

The work conducted in this brick will be used and demonstrated in use case 3.4 “Test case definition interlinked with model based requirement engineering / Variant management integration”. It shall allow the usage of a simulation model – imported from the AVL Data Backbone – within an embedded verification platform that will be built on the basis of the SoS Platform. The middleware will be instantiated / configured so that the individual requirements of the building blocks, e.g. the different Electronic Control Units (ECUs) can be fulfilled.

More specific use case needs will be identified in the next phase of the project.

2.9.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.9.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.10 SoS Infrastructure Wireless Interface

2.10.1 Description

Name:	SoS Infrastructure Wireless Interface
Contact:	TTTech, Christian Reinisch, christian.reinisch@tttech.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.10.2 Manual

Today, most platforms for safety critical applications are confined to a discrete construction where communication is mostly realized over dedicated interconnects on the hardware. In the setting of a system of systems platform such as the one described in Chapter 10 or in the automotive use case 3.4, the system building blocks might also be interconnected using dedicated networks such as TTEthernet. These specialized networks ensure that the initial requirements e.g. in terms of safety parameters, are preserved even when an extension and integration via a network takes places (i.e. a System of Systems platform is constructed).

In recent years, a clear trend towards wireless communication can be observed. Such wireless networks are traditionally unreliable and thus suited well for (uncritical) office and entertainment scenarios but of limited applicability in cases where dependable behavior is required, as it is the case e.g. in the automotive domain. However, especially in the automotive area they could realize vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) connections and eventually even connect vehicles to the Internet of Things. Hence, such an interface could allow realizing concepts such as an Apps Store for automotive purposes connecting automotive OEMs to the car and its owner.

The challenge in this brick is therefore twofold:

- Develop a wireless extension that meets requirements of deterministic communication as it is required in safety-critical domains such as automotive.
- Provide a configuration means that can be used during the SoS platform setup and/or operation to define links and their properties. This work needs to take into consideration how the SoS platform is configured as well as provide means to define specific communication properties of the wireless link

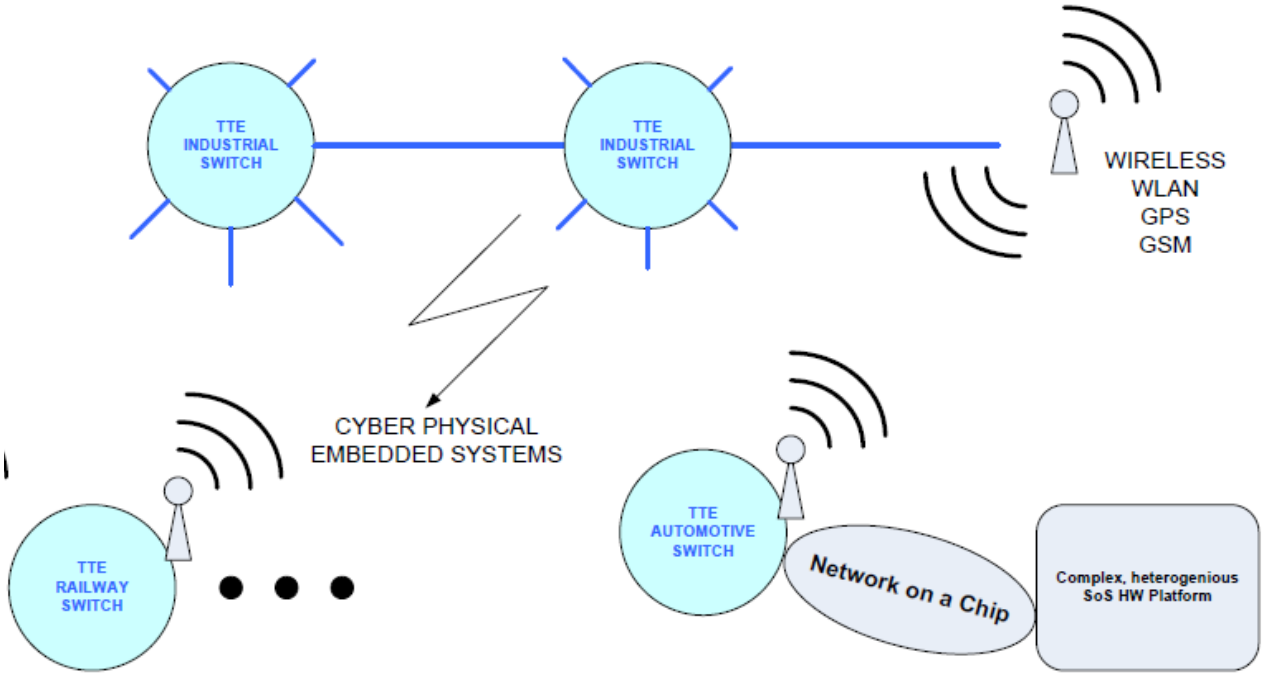

Figure 11: Example for the wireless communication within the SoS infrastructure

To fulfill these goals, an adaptation or extension of the configuration tools (for wired networks configuration) might also be required. Therefore, this work will be done in alignment with the goals from Chapter 9, as potentially even an extension of the overall tool chain will be required.

2.10.3 Use Case coverage and application

Within the CRYSTAL project it is the goal to extend the System of systems platform with a wireless link that allows for the communication of the system-of-systems platform with other infrastructure components like main stations, visualization endpoints, monitoring and maintenance infrastructure etc.

The developments of this brick closely relate to the other bricks of TTTech that target the development of SoS infrastructure parts and TTEthernet tooling. The brick is going to be demonstrated in use case 3.4 “Test case definition interlinked with model based requirement engineering / Variant management integration”. There it shall be exploited in a way that it supports use case requirements such as the provision of wireless monitoring or even debug interfaces to the SoS platform.

More specific use case needs will be identified in the next phase of the project.

2.10.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.10.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:		**n/a**			

2.11 Changes to tresosStudio

2.11.1 Description

Name:	Changes to tresosStudio
Contact:	Elektrobit, Irfan Delbassez, irfan.delbassez@elektrobit.com
Dependencies	n/a
License	n/a
Additional information	n/a

2.11.2 Manual

For the configuration of the Elektrobit AUTOSAR target stack (AutoCore) tresosStudio [Tresos, n.d.] is used. Depending on design decisions of WP3.6 adoptions of the tooling are necessary and will be realized by Task 6.5.11. Also adoptions to the IOC infrastructure are needed and will be done within this task.

EB tresos Studio is intended to be used for configuring and generating ECU Basic SoftWare (BSW) compliant to the AUTOSAR standard. EB tresos Studio can import system description data from AUTOSAR system description files to do partially the configuration of each BSW which compose our EB tresos Studio project. An EB tresos Studio project can host an ECU configuration consisting of module configurations in the form of XML files or a system description model in the form of a database or both.

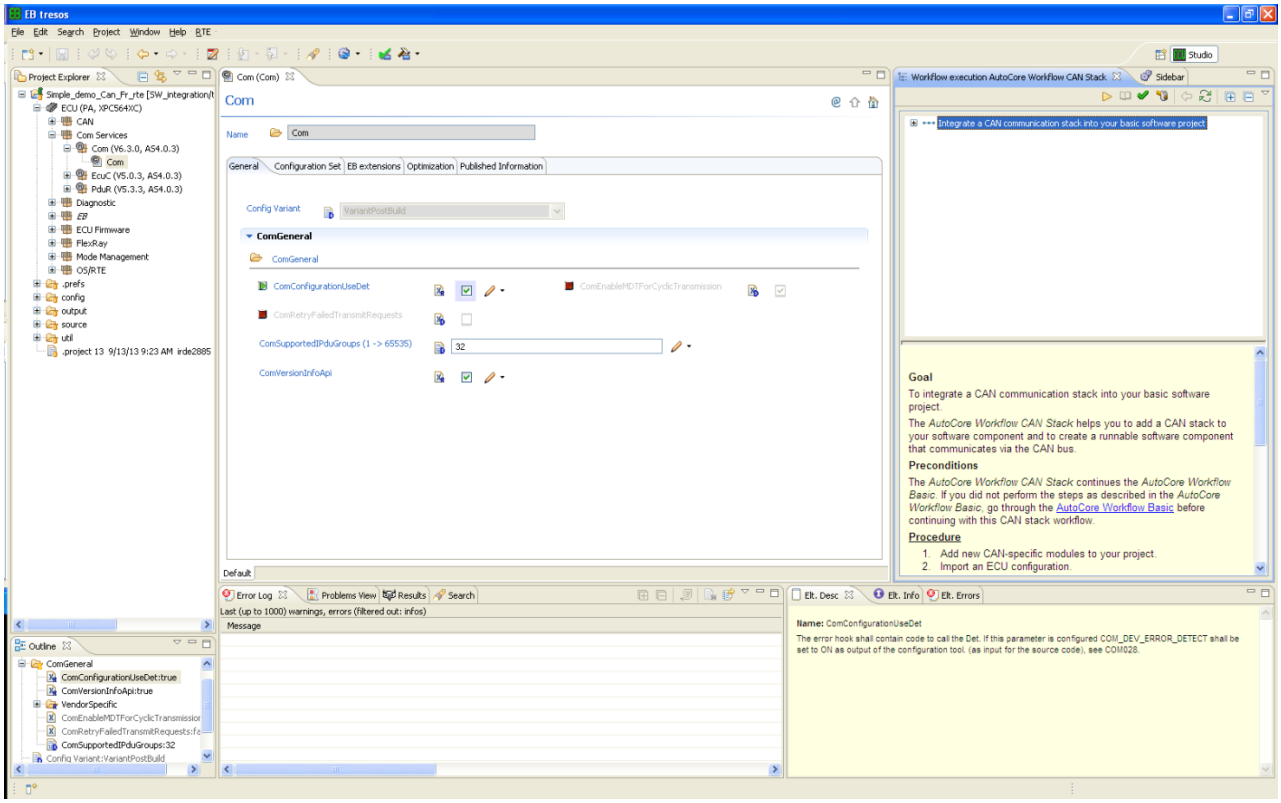

Figure 12: Main window of Tresos Studio

The tool environment is open to replace individual AutoCore modules with company-specific implementations or to add further modules, such as complex device drivers or standard modules of the ECU application layer.

Currently in order to design a complete Autosar ECU, an additional tool for creating or editing AUTOSAR Software Component Descriptions (called Authoring tool) is required. The developments in CRYSTAL could allow EB Studio to eventually support this feature.

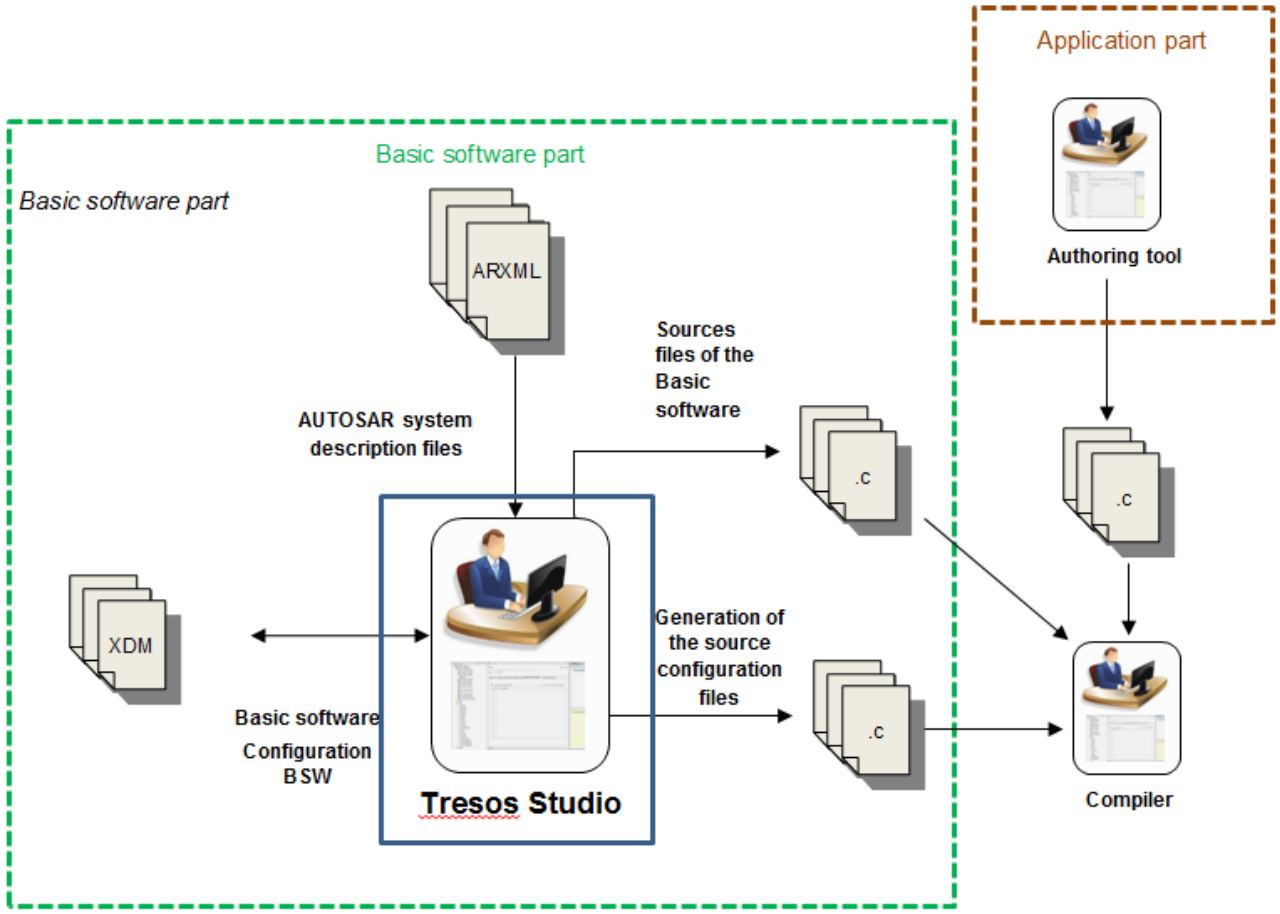

Figure 13: Autosar configuration generation process with Tresos Studio

2.11.3 Use Case coverage and application

The developments of this task will be demonstrated in CRYSTAL use case 3.6 OS MultiCore Compatible AUTOSAR & Safety Mechanisms for ISO26262 compliance.

More specific use case needs will be identified in the next phase of the project.

2.11.4 General Improvement

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

2.11.5 Integration and Interoperability

The definition of Technical Items will be part of the next phase of the project.

TI NAME:					
TI_ID	CRYSTAL_TI_x	**Kind of TI**	T, M, MM, G	**Contact email**	TBD
Description: TBD					
Link to internal working documents:	**n/a**				

3 Terms, Abbreviations and Definitions

CRYSTAL	**CR**itical S**YST**em Engnieering **A**cce**L**eration
CAN	Control Area Network
CO	Confidential, only for members of the consortium (including the JU).
D	Demonstrator
E/E	Electric/Electronic
ECS	Engine Control System
ECU	Electronic Control Unit
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
IOS	Interoperability Specification
O	Other
OEM	Original Equipment Manufacturer
OS	Operating System
OSLC	Open Services for Lifecycle Collaboration
P	Prototype
PP	Restricted to other program participants (including the JU).
PU	Public
R	Report
RE	Restricted to a group specified by the consortium (including the JU).
SoS	System of Systems
SP	Subproject
SQL	Structured Query Language
TTE	TTEthernet
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VFB	Virtual Functional Bus
WLAN	Wireless Local Area Network
WP	Work Package
XML	Extensible Markup Language

Table 2: Terms, Abbreviations and Definitions

4 References

[AS, n.d.]	*Arctic Studio – the development tools,* http://www.arccore.com/products/arctic-studio/
[ARTOP, n.d.]	*Artop AUTOSAR Tool Platform User Group,* http://www.artop.org
[Chen, 2013]	Chen, D. et al.; *Advances in Automotive System Modeling: EAST-ADL*; Automotive EE-times, http://www.automotive-eetimes.com/en/advances-in-automotive-system-modeling-east-adl-part-1.html
[EMF, 2014]	*Eclipse Modeling Framework Project (EMF),* https://www.eclipse.org/modeling/emf/
[Tresos, n.d.]	*Tresos Studio*, http://automotive.elektrobit.com/home/ecu-software/autosar/eb-tresos-product-line/eb-tresos-studio.html
[TTE, 2011]	SAE Standard AS6802: *Time-Triggered Ethernet*, http://standards.sae.org/as6802/

Table 3: List of References