AMALTHEA

(ITEA 2 – 09013)

Model Based Open Source Development Environment for Automotive Multi-Core Systems

---

Deliverable: D 4.4

Report on model and tool exchange

Work Package: 4

Continuous tool chain platform

Task: 4.4

Implementation/customization of tool chain elements

Document type:	Deliverable	**Classification:**	**Public**
Document version:	Final	**Contract Start Date:**	01.07.2011
Document Preparation Date:	31.04.2014	**Duration:**	30.04.2014

ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

Contents

Executive Summary iv

1 Introduction 1

2 Technical Overview 2
2.1 Basic Information . . . 2
2.2 Installation . . . 3
2.3 Examples and Help . . . 3

3 The AMALTHEA Model 5

4 Working on the AMALTHEA Model 7
4.1 Overview . . . 7
4.2 Workflows . . . 7
4.3 AMALTHEA sample . . . 8
4.4 Industrial UseCase . . . 9

5 Event Traces 11
5.1 Overview . . . 11
5.2 Trace Formats . . . 11
5.2.1 BTF . . . 11
5.2.2 HTF . . . 12

6 Extendibility of the AMALTHEA Tool Platform 14
6.1 Timing Architects Toolsuite . . . 14
6.1.1 Interoperability . . . 14
6.1.2 Functionality . . . 14
6.2 ETAS Ascet . . . 15

List of Figures

2.1 AMALTHEA Tool Platform 2
2.2 AMALTHEA Tool Platform - Splashscreen 3
2.3 AMALTHEA Tool Platform - Examples Wizard 4
2.4 AMALTHEA Tool Platform - Help 4

3.1 AMALTHEA Model in relation to existing tools and standards 5
3.2 Model editor 6

4.1 AMALTHEA Model and the information flow 7
4.2 Sample Workflow Screenshot 8
4.3 Sample Workflow Overview 9
4.4 Industrial Workflow 10

5.1 OSEK task states and transitions 11
5.2 Extended task states and transitions 12
5.3 HTF trace format conversions 13

6.1 AMALTHEA extendibility - Timing Architects Toolsuite 15

Executive Summary

AMALTHEA is a ITEA 2 funded project that is developing an open and expandable tool platform for automotive embedded-system engineering based on model-driven methodology. Key features include the support for multicore systems combined with AUTOSAR compatibility and product-line engineering. The resulting AMALTHEA Tool Platform is distributed under an Eclipse Public License.

This document is the final deliverable of Work Package 4 "Continuous tool chain platform" of AMALTHEA. It is produced within Task 4.4 "Implementation/customization of tool chain elements".

Content. This deliverable D 4.4 "Report on model and tool exchange" provides a very short description of the AMALTHEA Tool Platform and shows how to use and extend the platform. The document is classified as "Public". It mainly serves as a summary and contains references to more detailed information that is already published as part of the open source contributions of the AMALTHEA project.

Intended readership. The target audience of this document are the project sponsors and project members of the AMALTHEA project. Moreover, this document addresses also *project-external readers who are interested in the way how the AMALTHEA models can be used as exchange format and how custom tools can be integrated.* For this purpose, this document also includes some introductional material about AMALTHEA and can be read without referring to other documents.

Overview. The document is structured in the following main chapters:

Chapter 1 "Introduction" provides a short introduction to the results of WP4 and the relation to other parts of the AMALTHEA project.

Chapter 2 "Technical Overview" provides an overview on the Eclipse based AMALTHEA implementation, the prerequisites and the different installation possibilities.

Chapter 3 "The AMALTHEA Model" introduces the high level modeling of embedded multicore systems that contains several sub models to describe hardware, software behavior, timing constraints, etc.

Chapter 4 "Working on the AMALTHEA Model" provides a brief overview on the possibilities to execute customized tool chains. An example shows how existing tools can be reused and additional tools can be integrated.

Chapter 5 "Event Traces" introduces an extended description of task states that allows to characterize the software execution on multicore systems. It also introduces two event trace format specifications (BTF and HTF) that are results of the AMALTHEA project.

Chapter 6 "Extendibility of the AMALTHEA Tool Platform" describes already existing tools that provide import/export interfaces for the AMALTHEA models. These (commercial or open source) tools provide additional editing, visualization or simulation features.

1 Introduction

AMALTHEA is a ITEA 2 funded project that is developing an open and expandable toolchain for automotive embedded-system engineering based on model-driven methodology. Key features include the support for multicore systems combined with AUTOSAR compatibility and product-line engineering. The resulting AMALTHEA Tool Platform is distributed under an Eclipse Public License.

The contributions to the AMALTHEA development approach were distributed over the different work packages. Work Package 1 was concerned with requirements engineering, Work Package 2 provided a tool for variability modeling. These activities and some results of Work Package 3 (e. g. architectural and behavioral modeling) were described in other deliverables. Deliverable D3.4 "Prototypical Implementation of Selected Concepts" gives a good overview and is also publicly available.

This document shows how to use and extend the AMALTHEA Tool Platform. It mainly serves as a summary and contains references to more detailed information that is already published as part of the open source contributions of the AMALTHEA project.

The main focus is the basic infrastructure of the platform that was contributed by Work Package 4:

- a comprehensive data model (see Chapter 3)
- a generic execution environment / workflows (see Chapter 4)
- an extended state model for event traces (see Chapter 5)

The document also contains open source results from other work packages, e. g. the algorithms for partitioning and mapping of multicore systems as part of the workflows or the "Hardware Trace Format".

The last chapter shows that the tool vendors in the AMALTHEA project are using the AMALTHEA model as an exchange format. Timing Architects already integrated the import and export capabilities in their commercial tools.

2 Technical Overview

2.1 Basic Information

The Eclipse Automotive Industry Working Group[1] (Auto-IWG) is a group of automotive companies and tool developers that cooperate to provide an Eclipse automotive distribution. This distribution is based on the regular Eclipse distribution and adds various tools like C/C++ development tools (CDT), XML editors, Eclipse Modeling Framework (EMF), and Xtext/Xtend on top of it.

As the AMALTHEA project originally comes from the automotive domain, the AMALTHEA Tool Platform is based on the Auto-IWG distribution. AMALTHEA further adds its own models and tools to this distribution as depicted in Figure 2.1. Further third-party tools are integrated into the AMALTHEA Tool Platform like Franca Interface Definition Language (IDL), Eclipse Requirements Framework (RMF), Eclipse Damos, and YAKINDU Statechart Tools (SCT).

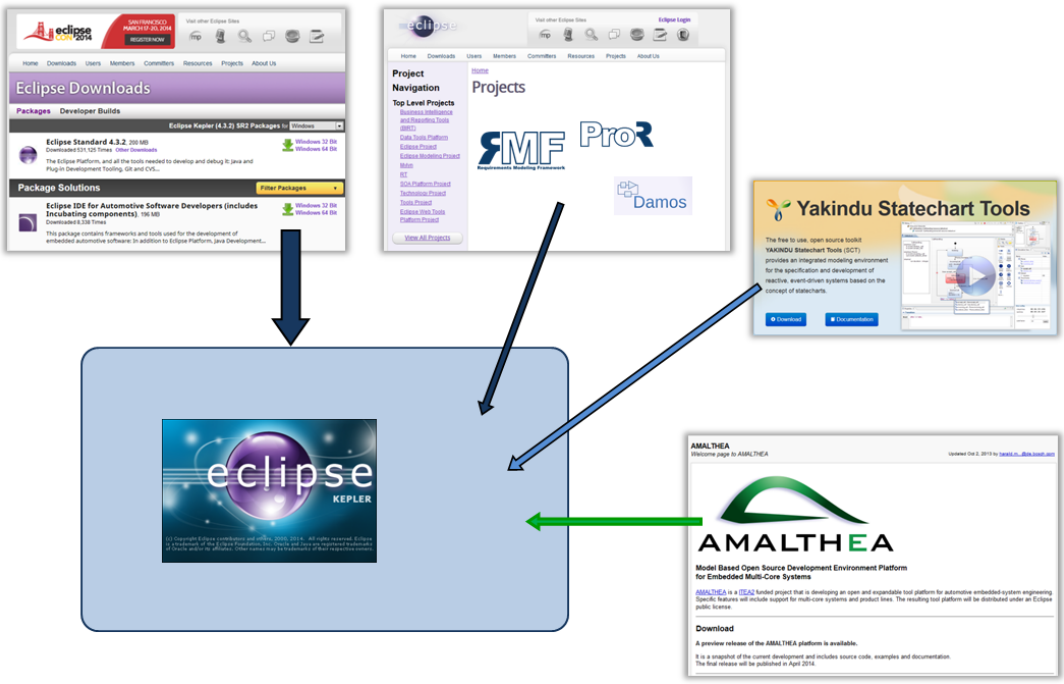

Figure 2.1: AMALTHEA Tool Platform

[1] http://www.eclipse.org/org/workinggroups/autowg.php

2.2 Installation

There are two possibilities to install the AMALTHEA Tool Platform. The first one is the recommended way. On the AMALTHEA Project website[2] complete AMALTHEA Tool Platform distributions for various operating systems can be found. These distributions just need to be downloaded, unzipped, and run.

Figure 2.2: AMALTHEA Tool Platform - Splashscreen

Another way to install the AMALTHEA Tool Platform is to integrate it into an existing Eclipse distribution that might be already in use. On the same AMALTHEA Project website, an Eclipse update site can be found that contains all AMALTHEA models and tools. Third-party tools are linked from this update site. The update site can be used in the ordinary way to add the AMALTHEA Tool Platform feature to an existing Eclipse installation. Such an installation lacks the AMALTHEA branding but is still fully functional.

2.3 Examples and Help

The most detailed information about the AMALTHEA Tool Platform is maintained as part of the open source distribution. The Eclipse Development Environment contains Wizards to create new examples at runtime. A screenshot of the provided AMALTHEA examples is shown in Figure 2.3.

The documentation is integrated as part of the Eclipse help system. It contains a User Guide, a Developer Guide and a detailed model description. The help contents and one of the model diagrams are depicted in Figure 2.4.

[2] http://amalthea-project.org/

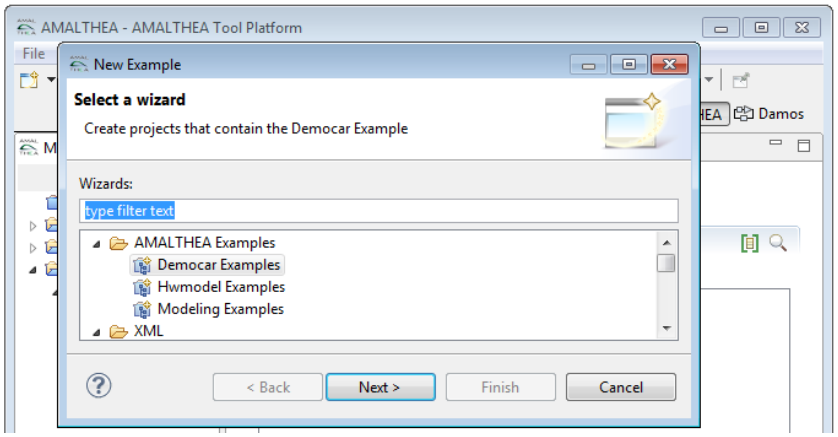

Figure 2.3: AMALTHEA Tool Platform - Examples Wizard

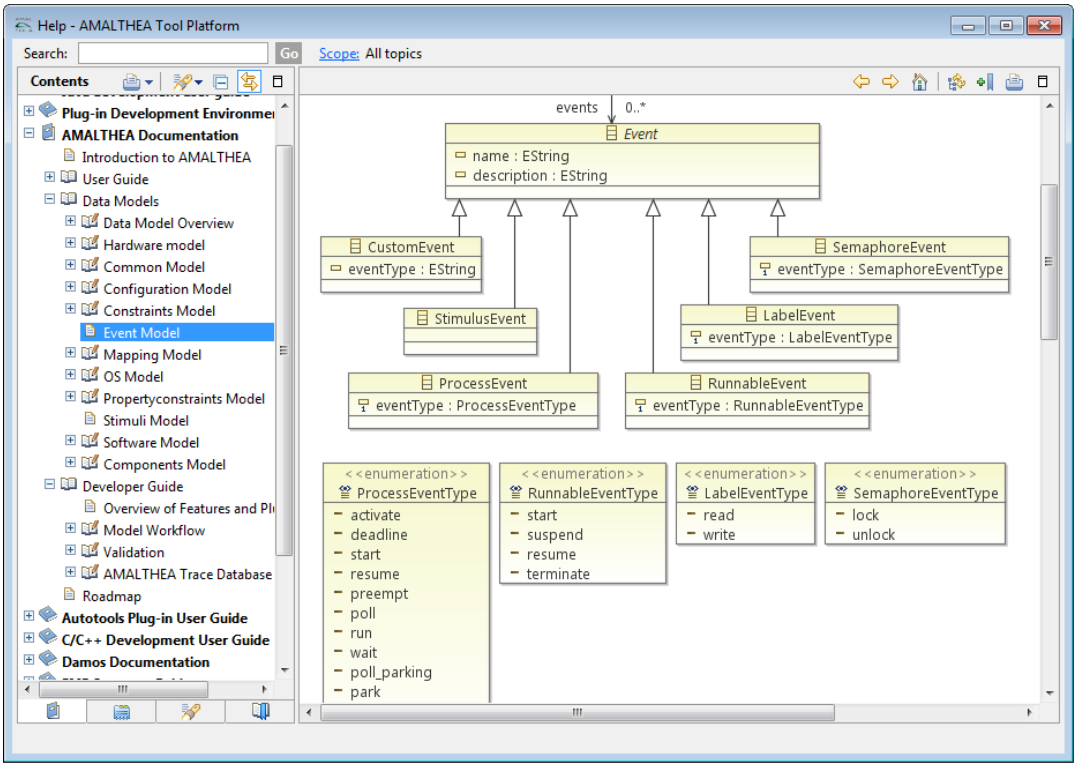

Figure 2.4: AMALTHEA Tool Platform - Help

3 The AMALTHEA Model

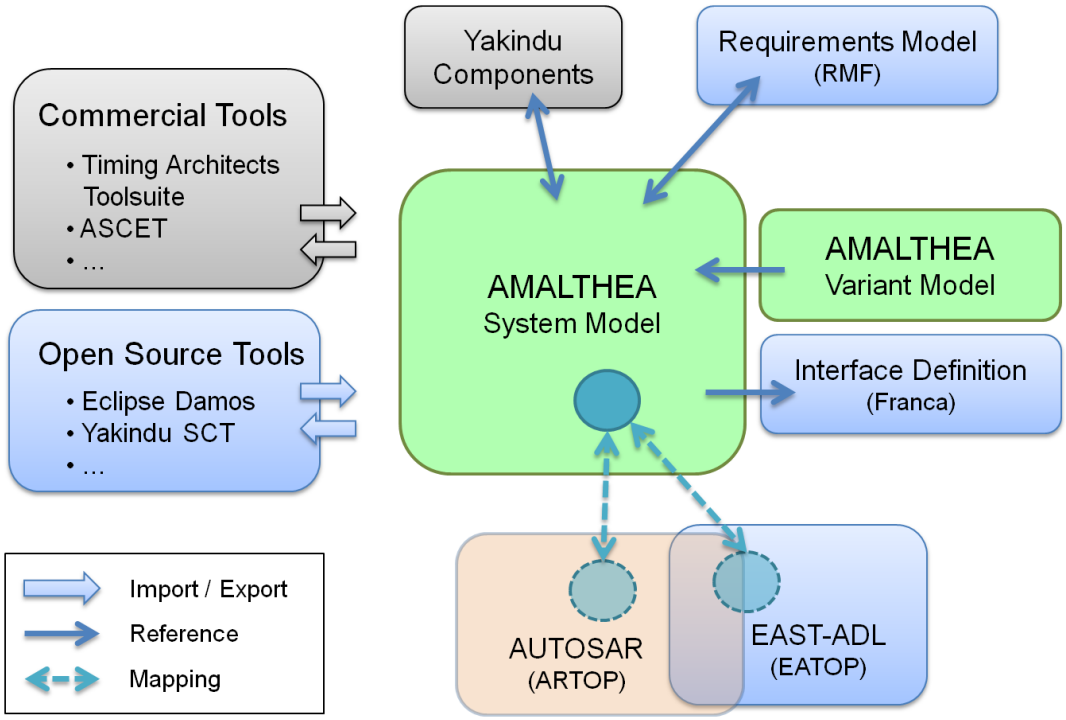

Figure 3.1: AMALTHEA Model in relation to existing tools and standards

One of the main results in the AMALTHEA project was to create a meta model representing the software structure and additional requirements like timing information. This model is implemented using Eclipse EMF [3] and Xcore [4].

The AMALTHEA model itself is split into several parts, each representing one main aspect of the model. These parts are described in more detail in Table 3.1.

Additional information about structure and the contained elements can be found in the Eclipse AMALTHEA platform in the provided help system (in the main toolbar under Help -> Help Contents). There you can find one section named “AMALTHEA Documentation”, which should be the first place to get further information about AMALTHEA.

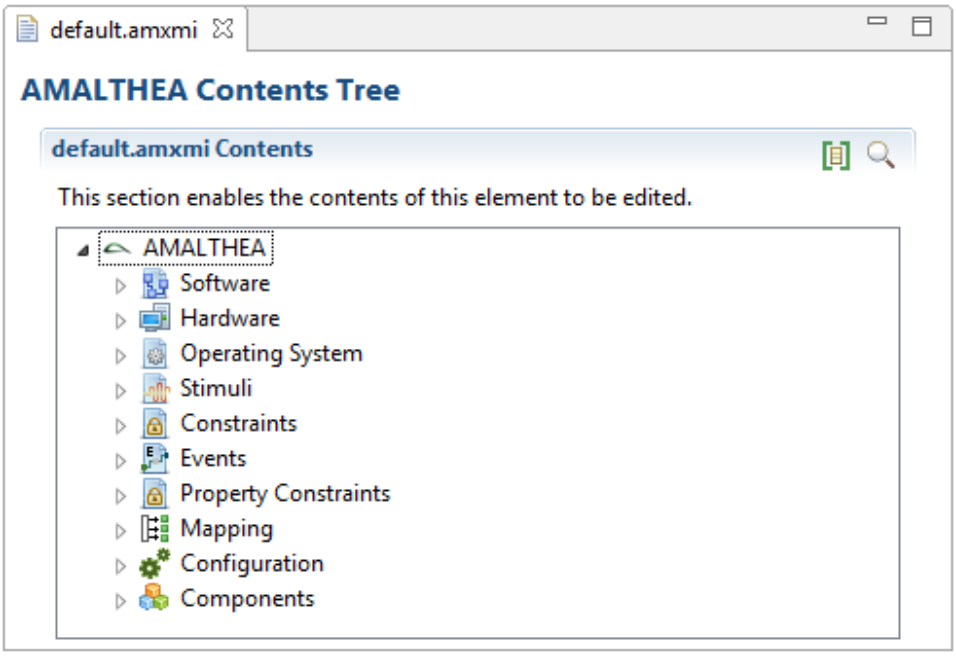

Figure 3.2: Model editor

Name	Description
Central	Container for all other model parts to store them in one file
Common	Provides basic object definitions for other model to reuse
Components	Definition of Components and a hierarchy of them
Config	Contains definitions and configurations relevant for simulation or hardware tracing
Constraints	Provides definitions for EventChains, TimingConstraints, AffinityConstraints and RunnableSequencingConstraints
Events	Event definitions to reuse at EventChains
Hardware	Abstract structure definition of the target Hardware
Mapping	Mapping definitions of Software elements to Hardware components
Operating System	Abstract information of the OS, like Scheduler and Scheduling Algorithms
PropertyConstraints	Constraints of Software to Hardware mappings
Stimuli	Stimulus definitions for activations
Software	Structure of software components like Tasks, Runnables and Tasks

Table 3.1: Model Overview

4 Working on the AMALTHEA Model

4.1 Overview

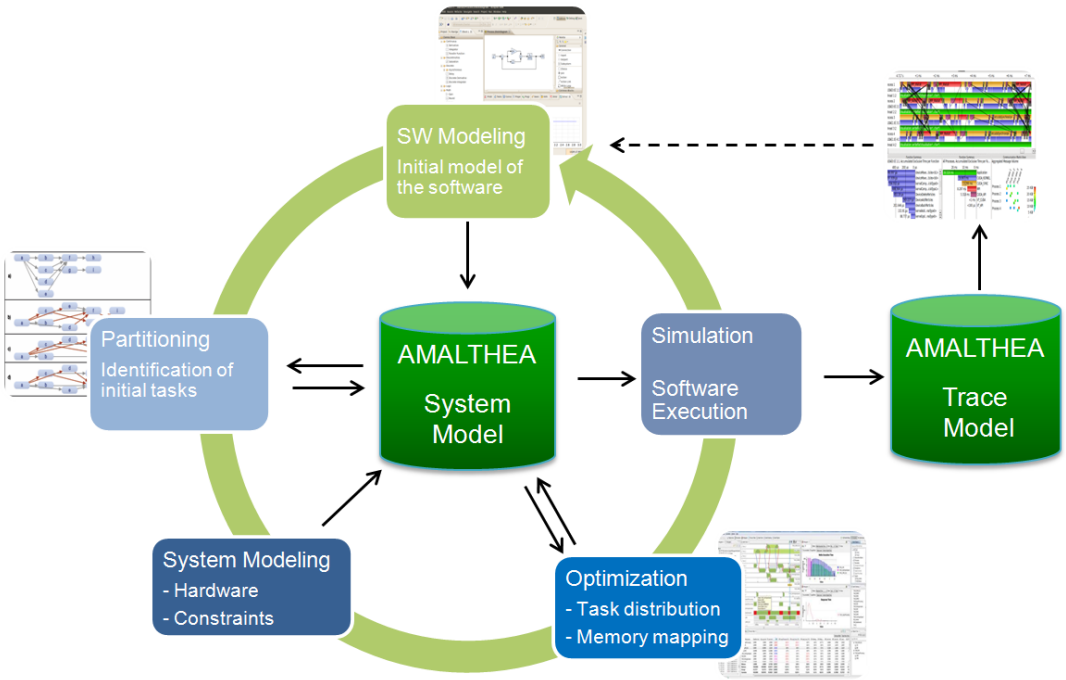

Figure 4.1: AMALTHEA Model and the information flow

4.2 Workflows

The AMALTHEA toolchain provides the possibility to automate working steps or the execution of algorithms as workflows. The sequence of steps in a workflow can be defined and executed in the AMALTHEA runtime environment. This offers the possibility to add user-specific workflows that reuse workflow elements. The purpose of this is to make it easier for the user to work with recurring steps in a convenient way (e. g. if only some parameters are changing).

To achieve this, AMALTHEA uses an already existent framework to define these steps and their relevant parameters. This framework is the Modeling Workflow Engine 2 (MWE2) [1], which is orignal provided and implemented in the context of the Xtext [2] framework.

The principle of this workflow concept is to separate each working step in different implementations and to gather them in one defined workflow. This makes it flexible enough for different requirements and to build your own workflow based on available steps. AMALTHEA provides

one sample in Section 4.3 which is a gathered workflow of different steps to perform the actions from the AMALTHEA sample in a more automated way.

The central concept of the MWE2 is also to use the provided Context object in the implementation. This Context object consists in general of a HashMap with key/value pairs. As values the processed root model objects can be stored to make them available for the other defined steps. The different steps can access them using the proper key, which can be defined in the workflow definition. The sample in Section 4.3 is showing this principle in more detail.

In addition, AMALTHEA provides some already available steps to integrate in your own workflow definition. These elements are available in the Eclipse plugin *org.itea2.amalthea.workflow.base*. The following overview explains these elements:

- Model Reader: The implementation in *org.itea2.amalthea.workflow.util.AmaltheaReader* provides the ability to read different AMALTHEA model files, gather them to one model in the memory and store this result to the workflow context.
- Model Writer: The implementation in *org.itea2.amalthea.workflow.util.AmaltheaWriter* is responsible to write a current model available in a given slot of the workflow context to a defined file. This can be either one file containing all model elements or split files with one model part in one file.
- Add Schedule Points: The implementation *org.itea2.amalthea.workflow.util.AddSchedulePoints* adds to all Tasks, which are marked as cooperative in the preemption attribute, so called SchedulePoints to their call graph between all included elements. This has the effect to provide information for a simulation that at this step the current task can be interrupted.

4.3 AMALTHEA sample

The AMALTHEA workflow sample can be accessed through the provided Democar samples and is located in the mapping example project *org.itea2.amalthea.example.democar.mapping* in the folder *workflow*. There you can find a file named *sample_ workflow.mwe2*, which contains a configuration ready to run.

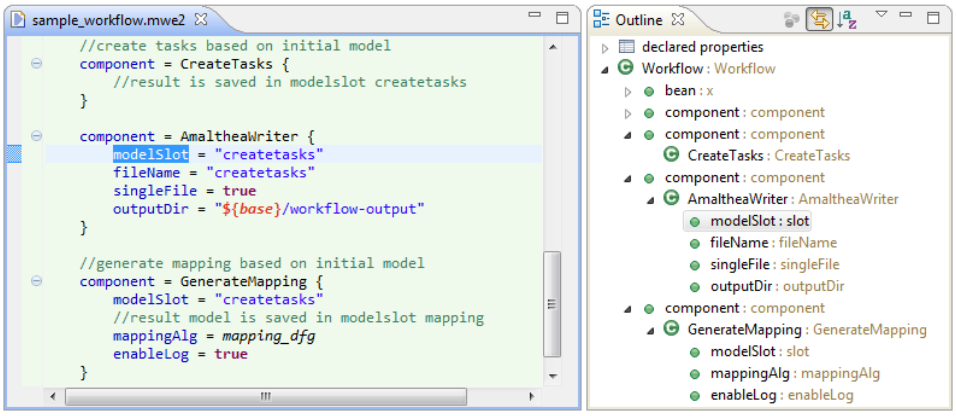

Figure 4.2: Sample Workflow Screenshot

Figure 4.3 shows a graphical overview of the steps on the left side and the elements stored in the context of the right side (the name of the context elements is the corresponding key to access the elements).

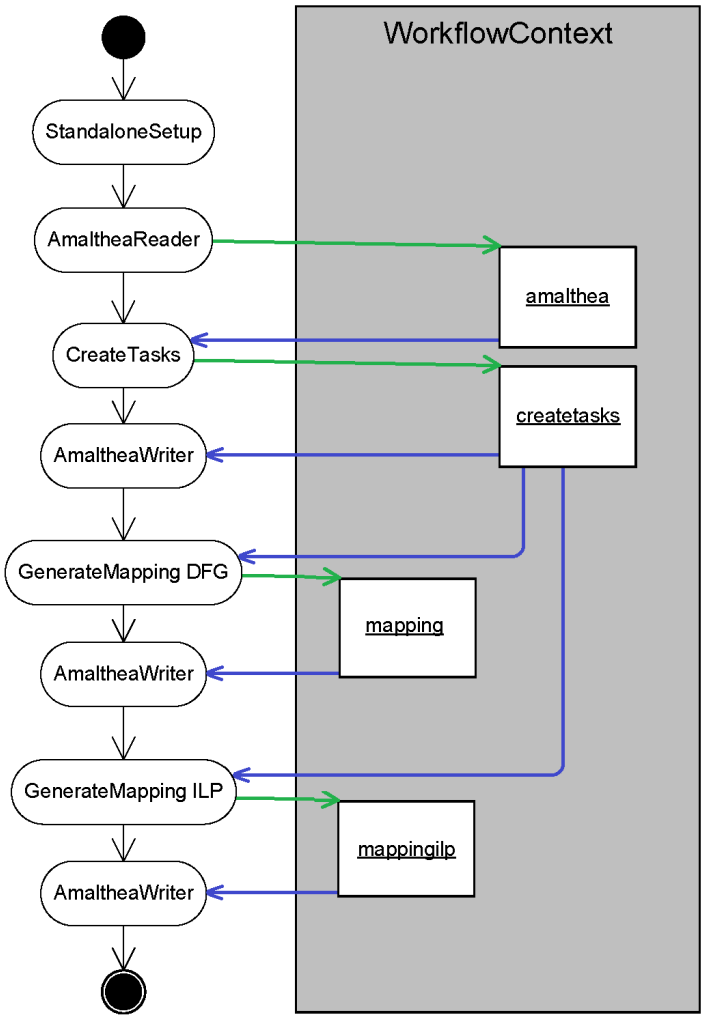

Figure 4.3: Sample Workflow Overview

As shown in the picture, in this workflow configuration every step stores its result in an own slot, which is afterwards accessed by its successor. Using this kind of configuration it is very flexible to access a specific result and to store it in a file, which is also shown in the sample workflow.

4.4 Industrial UseCase

The Industrial UseCase describes a current implementation to build an AMALTHEA model based on different sources in an automated way. To build such a model for a target platform, it is necessary to concat data from different data sources. Figure 4.4 shows this way from a high level view.

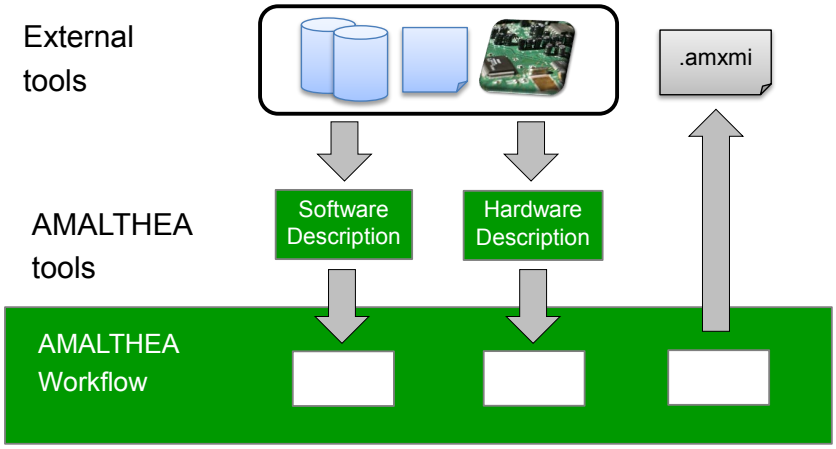

Figure 4.4: Industrial Workflow

As a first step it is needed to have all the data, containing structure of the software like Tasks, Runnables and Labels. Another important point is runtime information, which can come from a measurement on real hardware. Last but not least the information and structure of the Hardware must be available.

All these data is read by a defined workflow, which contains different steps to read all the data and building up the target AMALTHEA model. The result of this workflow is a model containing all information, which can then be further analyzed, either using additional workflow steps or a 3rd party tool like the Timing Architects Toolsuite using the available import mechanisms and then simulate it.

5 Event Traces

5.1 Overview

Automotive applications are characterized by hard real-time requirements. In that domain the ideas of the OSEK operating system and its task definition are widely used. The OSEK/VDX[1] portal provides further information.

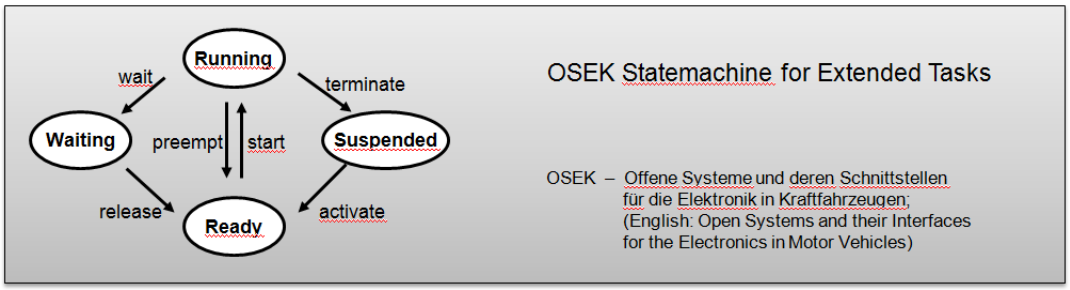

Figure 5.1: OSEK task states and transitions

To better characterize the behavior of a multicore system we extended the task states and transitions. Events that indicate the change between these extended task states are part of the new trace format specifications (see Figure 5.2).

5.2 Trace Formats

In the AMALTHEA project we defined two trace formats: BTF and HTF. Both format specifications are published under an open format license. The royalty-free terms ensure that the format remains accessible to everyone free of charge. The formats are documented in all its details and are free for all to implement. The license also requires that extensions have to be integrated and published under the same free license.

The specifications are published on the Eclipse Auto IWG (Industry Working Group) website[2]. The PDF versions of the documents are availabe for download.

5.2.1 BTF

The Best Trace Format (BTF) is based on the Better Trace Format, initially defined by Continental Automotive GmbH, and was extended in the context of the AMALTHEA project. It allows analyzing the behavior of a system in a chronologically correct manner in order to apply timing, performance, or reliability evaluations. In general, it assumes a signal processing system, where one component of the system notifies another component of the system. These

[1] http://osek-vdx.org/
[2] http://wiki.eclipse.org/Auto_IWG#Publications

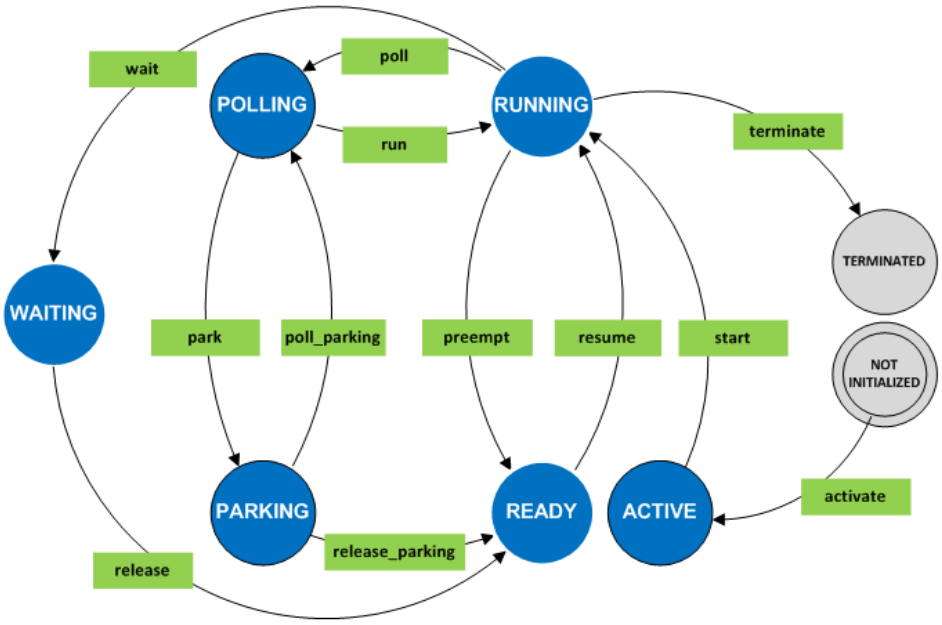

Figure 5.2: Extended task states and transitions

notifications are realized by events, stored in the BTF file. In comparison with compact trace formats from debugger traces, a BTF log of an event includes the entire information, namely: which component interacts with which component by an event. A BTF-file consists of two parts: a header-section and a data-section. The header-section contains meta-information on objects of the trace and optional comments. The data-section contains the trace-data of the simulation or measurement with optional comment-lines. Each line represents one event of the traced system in CSV format. The columns of the event-line describe the time, entities, and event. The BTF specification is published on the Eclipse Auto IWG (Industry Working Group) website.

5.2.2 HTF

The Hardware Trace Format (HTF) is a result of the AMALTHEA project. It is a compact trace format for the tracing of events on embedded systems, including multicore systems. Its binary representation shortens the execution time of the tracing instructions and improves the storage efficiency of the tracing data on the target system, compared to other trace formats. On the host system, the target traces can be saved in an ASCII **.htf* file without the need for any extra conversion. Still, it is compatible to other trace formats (e. g. BTF, OT1[3]), so a conversion into these formats is possible.

HTF files are divided into three sections. The first part is the header where meta data can be found. Exemplary parameters are the creation date, a description and the target system name. In the second section, reference tables are defined. These tables represent the association between the recorded events and the AMALTHEA software components. Every table entry consists of a specific hex value and a textual description. Subsequently, reference tables allow

[3]https://gliwa.com/ot1

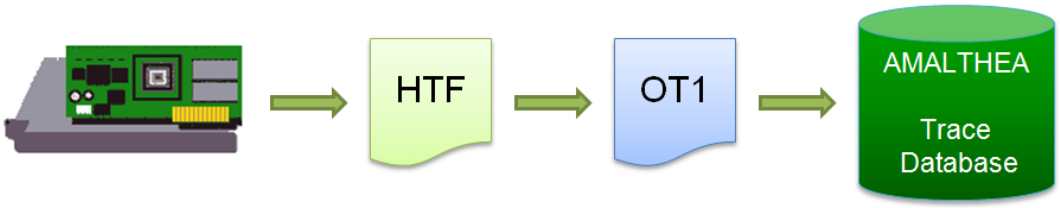

Figure 5.3: HTF trace format conversions

the interpretation of the binary trace data. The last section of the HTF file contains the recorded trace datasets. Each HTF trace dataset consists of a timestamp, an entity, and an event. When tracing single core systems, this part has one data section, whereas the trace of multicore systems can produce additional data sections. The HTF specification is published on the Eclipse Auto IWG (Industry Working Group) website. More detailed information on the HTF is given in the deliverable 3.4 (Prototypical Implementation of Selected Concepts).

6 Extendibility of the AMALTHEA Tool Platform

The AMALTHEA Tool Platform can also benefit from so called external tools, which are not build upon the AMALTHEA Tool Platform, in multiple ways. For that reason, this section focuses on the possibility of the AMALTHEA Tool Platform to interconnect with such external tools and describes a variety of benefits. All that is necessary for an external tool to communicate with the AMALTHEA Tool Platform and thus to be able to exchange information is a well defined interface. Since the AMALTHEA Tool Platform is Open Source, the required information is freely available and well documented.

6.1 Timing Architects Toolsuite

For this consideration, the proprietary discrete-event simulation, the Timing Architects Toolsuite [5] is taken into account. Figure 6.1 gives an overview on some of the benefits that can be achieved by interconnecting the AMALTHEA Tool Platform with the Timing Architects Toolsuite.

6.1.1 Interoperability

The first benefit that can be achieved by interconnecting the AMALTHEA Tool Platform with the Timing Architects Toolsuite is the gain of additional interoperability. Interoperability in this case means that it is possible to exchange information with different modeling languages and formats.

The Timing Architects Toolsuite for example, has its own model for internal data processing that contains all the information about the system under development. In addition to that it also provides an interface to AUTOSAR. So by building an interface from the AMALTHEA Tool Platform to the Timing Architects Toolsuite it is then possible to interchange information not only with the proprietary data model but also with the AUTOSAR model by using their AUTOSAR interface.

6.1.2 Functionality

In addition to the gained interoperability, the AMALTHEA Tool Platform can also benefit from additional functionality. A few examples are given in the following:

- Editing
  Although the AMALTHEA Tool Platform provides a basic functionality for editing a model, additional and more professional functionality can be gained by interconnecting to the Timing Architects Toolsuite.

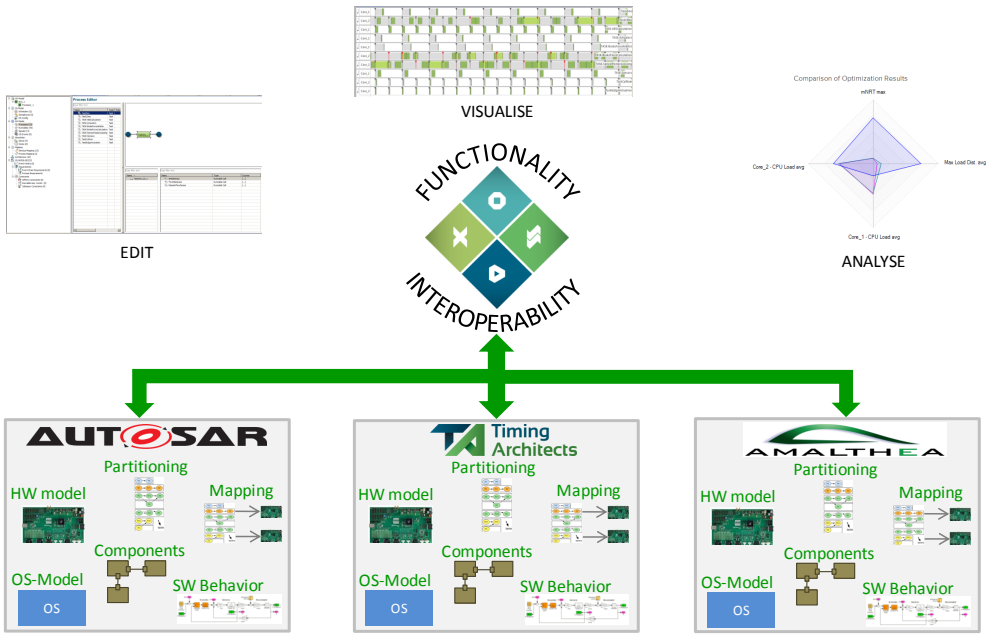

Figure 6.1: AMALTHEA extendibility - Timing Architects Toolsuite

- Visualization
  Currently, just a visualization of the individual model elements is available. Professional tools however provide more extensive functionality. such that for example the data dependencies between functions are displayed.

- Analysis
  Last but not least, the AMALTHEA Tool Platform does not provide thus far any possibility to analyse the system under development based on the model. An external tool like the Timing Architects Toolsuite however makes this possible. Thereby a system can be simulated based on a model.

As it can be seen, the AMALTHEA Tool Platform can highly benefit from so external tools like the Timing Architects Toolsuite. And since it is Open Source, the hurdle to create such a required interconnection is very low and easy to implement.

6.2 ETAS Ascet

The AMALTHEA Tool Platform enables the hardware/software co-design. Therefor it provides data structures and services which are to be instrumented to create hardware and software profiles. Based on this profiles AMALTHEA enables to perform services such as e. g. requirements collection and analysis, scheduling, optimization of deployment scenarios, etc. The ETAS Ascet

tool enables to create software descriptions which are conform to the AUTOSAR standard. As a contribution to the AMALTHEA tool platform ETAS developed an AMALTHEA adapter which enables to integrate software models developed in ASCET into the AMALTHEA platform and to use ASCET software models in the context and together with the integrated AMALTHEA services.

For the ASCET/AMALTHEA adapter ETAS decided for an innovative integration technology based on OSLC (Open Services for Lifecycle Collaboration). The intention of this specification is to couple different lifecycle tools from different vendors in an open approach. This is done by implementing a tool specific adapter to provide an interface for OSLC. In general there are different topics, which are covered by corresponding OSLC standardization working groups, so called domains. The EU project CESAR[1] (Cost-efficient methods and processes for safety relevant embedded systems) and the follow-up project CHRYSTAL have the goal to provide a tool chain for different use cases and topic areas. Therefore there are a lot of tools existent for the different steps of the development life cycle. Every characteristic of the tool chain has the problem to make an integration to the other chosen existent tools. CESAR has specified an Interoperability Specification, which covers this topic. The intention is to have a guideline on which technologies the tool chain is build on. One technology CESAR is using and specify is OSLC to couple the chosen tools. The reason for this is that there is a lean coupling of the tools by following the linked data approach. For the ASCET/AMALTHEA adapter implementation ETAS followed the recommendation of CESAR/CHRYSTAL. The open API of the AMALTHEA allowed to implement the OSLC adapter.

The import of ASCET software models into the AMALTHEA tool chain allows to perform complex scenarios for software/hardware development. Once imported the software description can be analyzed regarding its deployability, hardware/software profiles can be generated, and scheduling algorithms can be derived with inherent optimality criteria (see Section 6.1). Such software development services in the AMALTHEA environment can be easily extended by involving external tools and services. For this intend the OSLC technology—as utilized for the ASCET/AMALTHEA adapter—can be adapted to a broad range of services and tools to be integrated into the AMALTHEA environment. E. g. specific ETAS tools for AUTOSAR software stack development such as ETAS ISOLAR-A for authoring software component interface definitions and ETAS ISOLAR-EVE for real-time execution of software in a virtual environment can be integrated in the AMALTHEA platform. Such tools can assist and complement the software/hardware profile development in the AMALTHEA environment to perform an efficient development following the AMALTHEA methodology.

[1] http://www.cesarproject.eu

Bibliography

[1] CONTRIBUTORS, Various open-source: *MWE2*. April 2014. – `http://www.eclipse.org/Xtext/documentation.html#MWE2`

[2] CONTRIBUTORS, Various open-source: *Xtext*. April 2014. – `http://www.eclipse.org/Xtext/`

[3] ECLIPSE: *Eclipse Modeling Framework Project (EMF)*. December 2012. – URL `http://www.eclipse.org/modeling/emf/?project=emf#emf`

[4] ECLIPSE: *Xcore*. December 2012. – URL `http://wiki.eclipse.org/Xcore`

[5] TIMING-ARCHITECTS: *TA Toolsuite Version 13.09.0. TA Academic & Research License Program.* [Online]. Available: http://www.timing-architects.com