



**An Open Source Development Platform
For Embedded Multi- and Many-Core Systems**

Harald Mackamul, Robert Bosch GmbH

SPONSORED BY THE



APP4MC – Application Platform Project for MultiCore

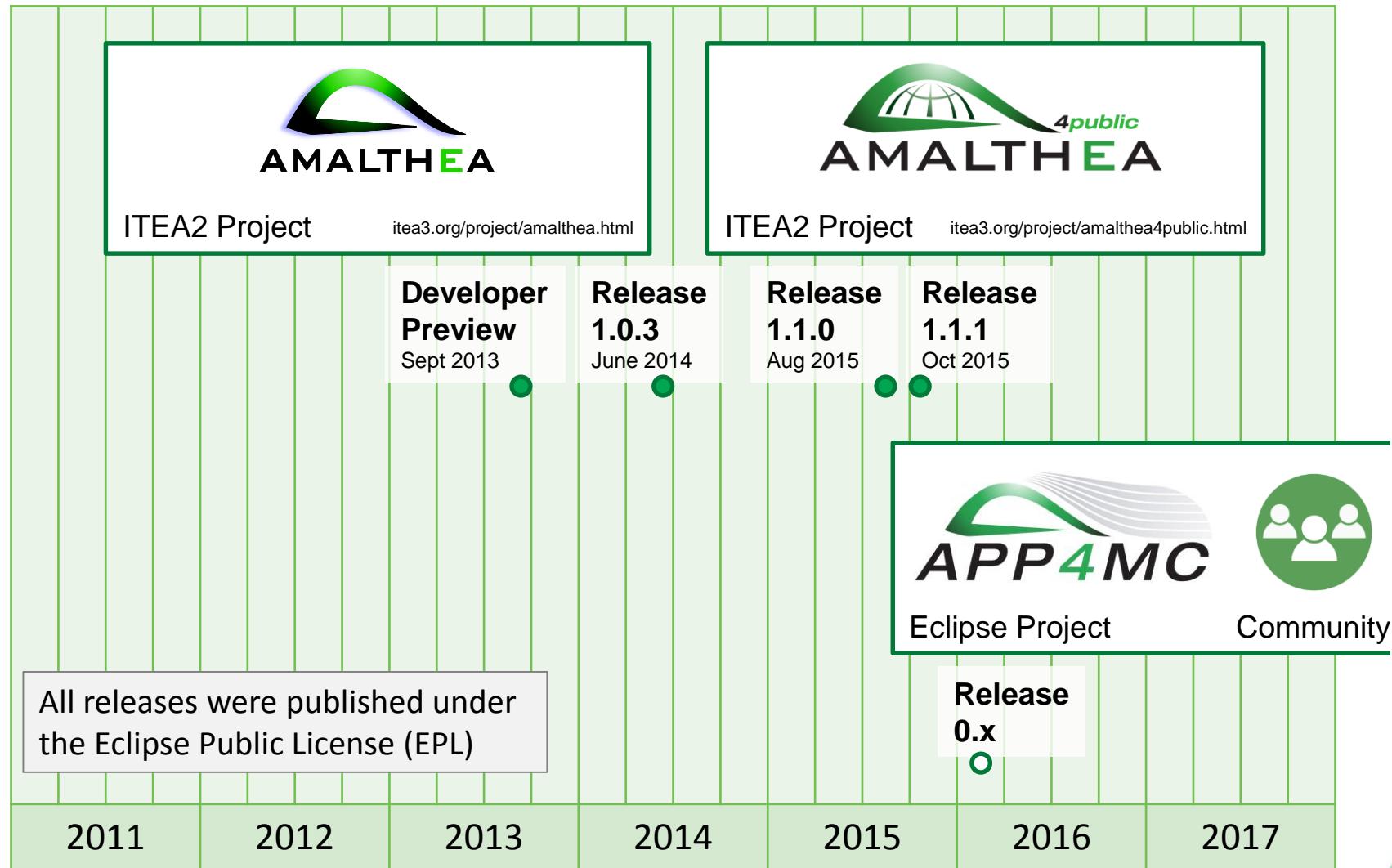
- AMALTHEA - Timeline and current project(s)
- Challenges for embedded multi- and many-core systems
- The AMALTHEA Platform
- Demo / Screenshots of current release
- APP4MC - Next steps

APP4MC – Application Platform Project for MultiCore

- **AMALTHEA - Timeline and current project(s)**
- Challenges for embedded multi- and many-core systems
- The AMALTHEA Platform
- Demo / Screenshots of current release
- APP4MC - Next steps

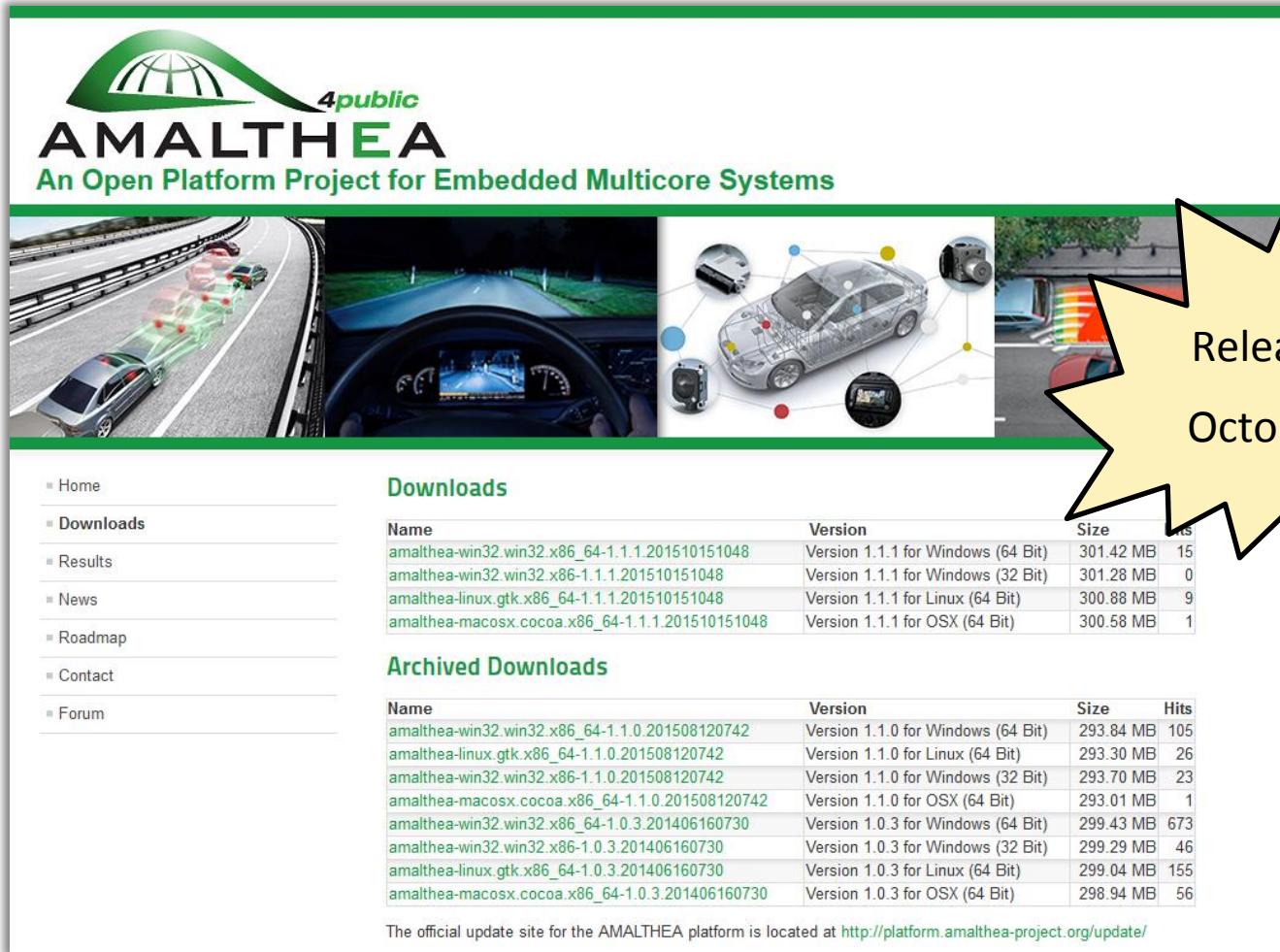
AMALTHEA

Timeline



AMALTHEA

Latest Open Source Releases



4public
AMALTHEA
An Open Platform Project for Embedded Multicore Systems

Home
Downloads
Results
News
Roadmap
Contact
Forum

Downloads

Name	Version	Size	Hits
amalthea-win32.win32.x86_64-1.1.1.201510151048	Version 1.1.1 for Windows (64 Bit)	301.42 MB	15
amalthea-win32.win32.x86-1.1.1.201510151048	Version 1.1.1 for Windows (32 Bit)	301.28 MB	0
amalthea-linux.gtk.x86_64-1.1.1.201510151048	Version 1.1.1 for Linux (64 Bit)	300.88 MB	9
amalthea-macosx.cocoa.x86_64-1.1.1.201510151048	Version 1.1.1 for OSX (64 Bit)	300.58 MB	1

Archived Downloads

Name	Version	Size	Hits
amalthea-win32.win32.x86_64-1.1.0.201508120742	Version 1.1.0 for Windows (64 Bit)	293.84 MB	105
amalthea-linux.gtk.x86_64-1.1.0.201508120742	Version 1.1.0 for Linux (64 Bit)	293.30 MB	26
amalthea-win32.win32.x86-1.1.0.201508120742	Version 1.1.0 for Windows (32 Bit)	293.70 MB	23
amalthea-macosx.cocoa.x86_64-1.1.0.201508120742	Version 1.1.0 for OSX (64 Bit)	293.01 MB	1
amalthea-win32.win32.x86_64-1.0.3.201406160730	Version 1.0.3 for Windows (64 Bit)	299.43 MB	673
amalthea-win32.win32.x86-1.0.3.201406160730	Version 1.0.3 for Windows (32 Bit)	299.29 MB	46
amalthea-linux.gtk.x86_64-1.0.3.201406160730	Version 1.0.3 for Linux (64 Bit)	299.04 MB	155
amalthea-macosx.cocoa.x86_64-1.0.3.201406160730	Version 1.0.3 for OSX (64 Bit)	298.94 MB	56

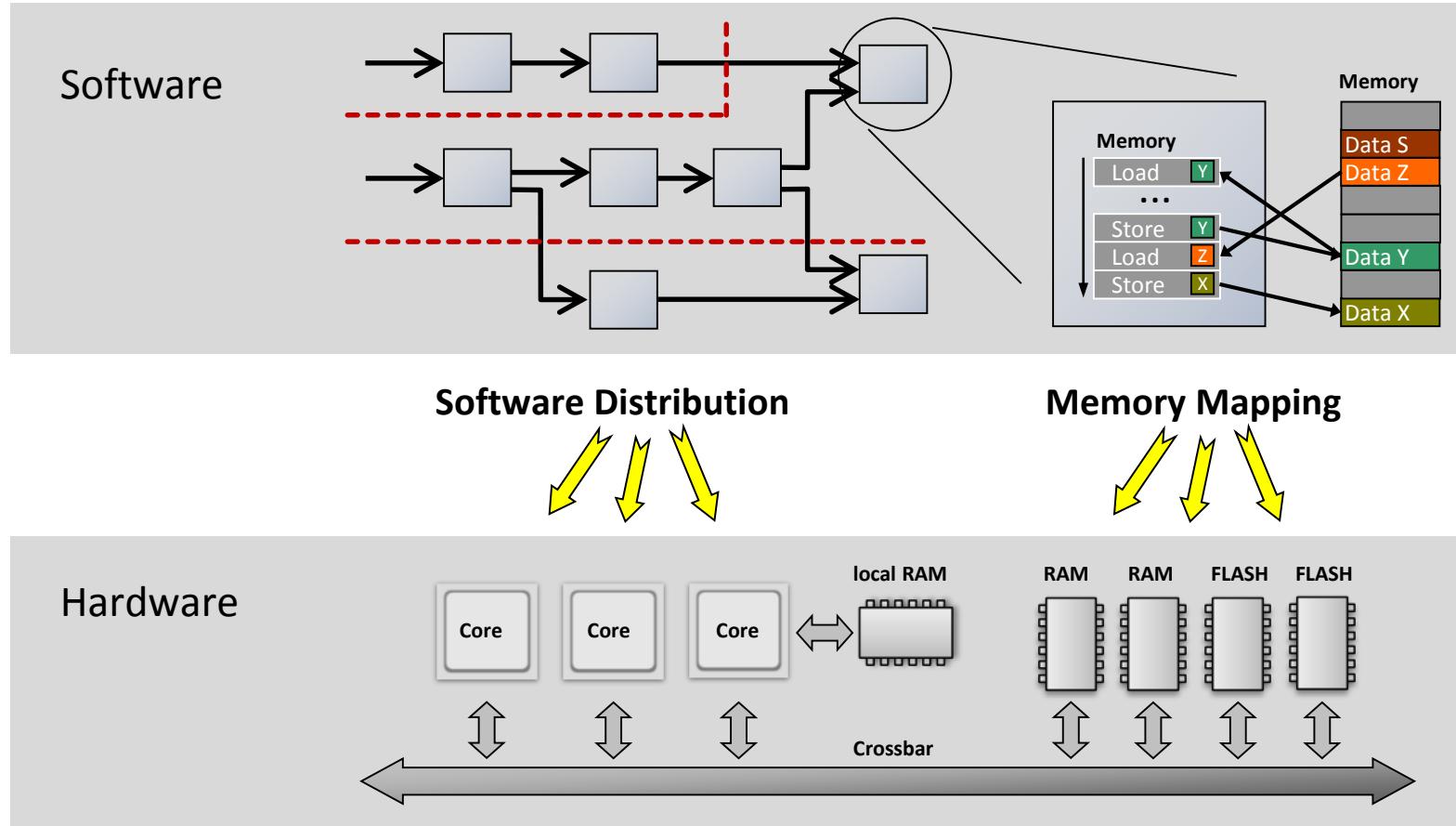
The official update site for the AMALTHEA platform is located at <http://platform.amalthea-project.org/update/>

<http://www.amalthea-project.org>

APP4MC – Application Platform Project for MultiCore

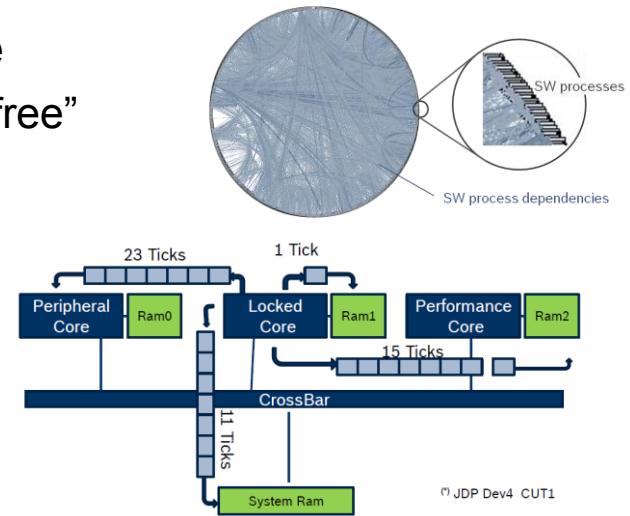
- AMALTHEA - Timeline and current project(s)
- **Challenges for embedded multi- and many-core systems**
- The AMALTHEA Platform
- Demo / Screenshots of current release
- APP4MC - Next steps

Challenges of Embedded Multi-Core



Challenges of Embedded Multi-Core

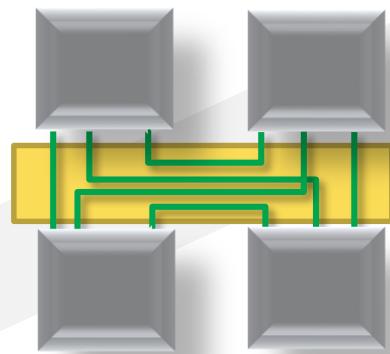
- **Typical Paradigm of Single-Core Software**
 - Blackboard Architecture: Memory access is for “free”
 - Integration challenge: scheduling of computation
- **Paradigm Change for Multi-/Many-Core**
 - Cross-Core Communication is expensive
 - Synchronization leads to high overheads
 - Memory location matters
 - Integration challenge: scheduling of computation and communication
- Sophisticated new tooling required for task distribution, memory location optimization and performance analysis



Cross-core communication is a new resource bottleneck

Multi-Core

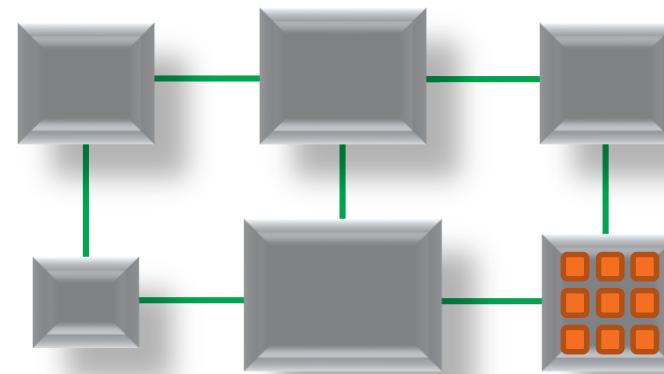
- Small number of homogenous cores with shared memory
- Mostly symmetric connectivity (e.g. crossbar)
- Limited impact on SW distribution



Increasing
Computing Power

Many-Core

- Larger number of **heterogeneous cores** with distributed memories
- Increasingly **heterogeneous connectivity** (Non-uniform Memory Access)
- High impact on SW distribution

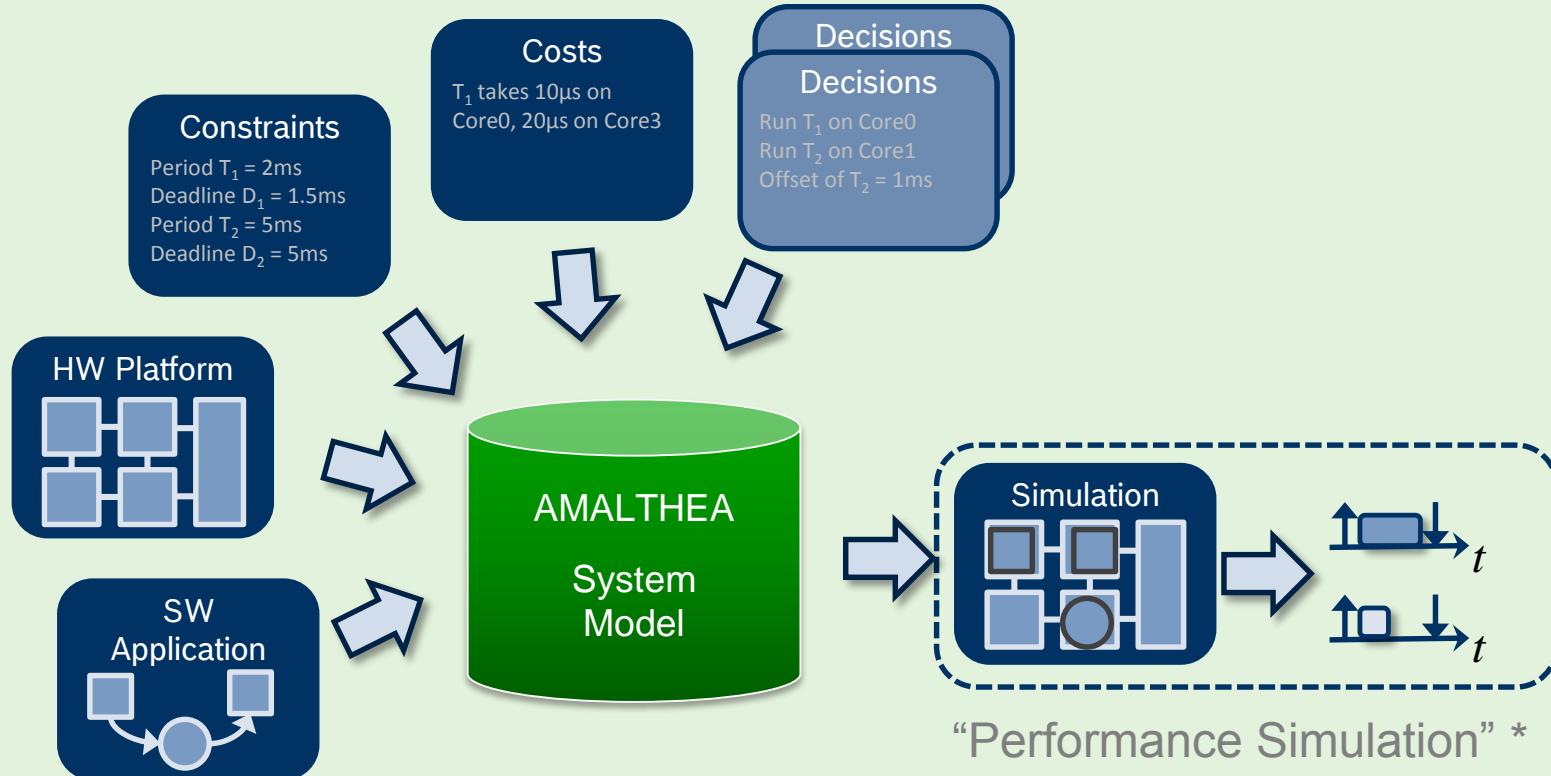


Today's automotive Multi-Cores already have Many-Core characteristics

APP4MC – Application Platform Project for MultiCore

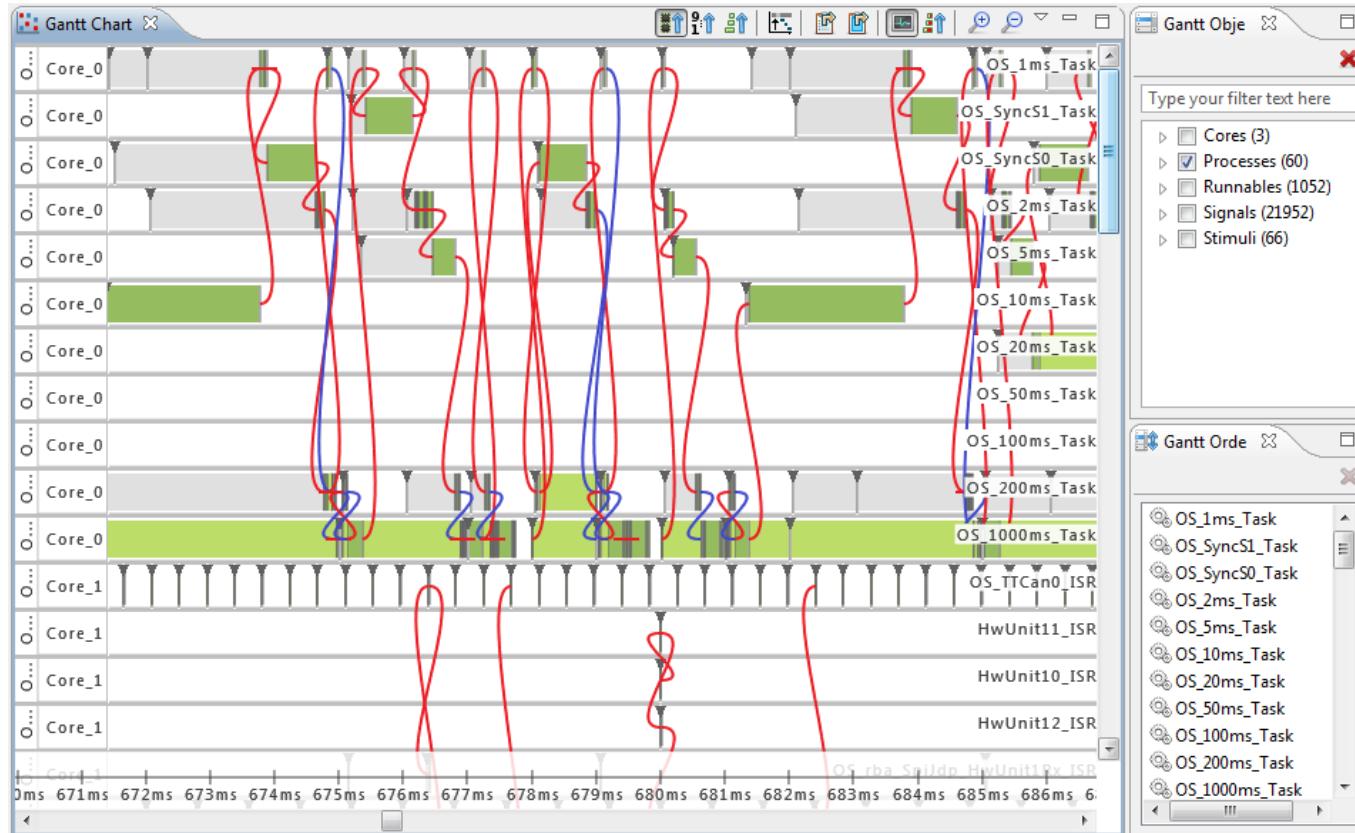
- AMALTHEA - Timeline and current project(s)
- Challenges for embedded multi- and many-core systems
- **The AMALTHEA Platform**
 - **High level description**
 - **Use cases**
 - **Connections to other Eclipse projects**
 - **Technical decisions and experiences**
- Demo / Screenshots of current release
- APP4MC - Next steps

AMALTHEA4public



* Focus on Timing, Scheduling

Performance Simulation

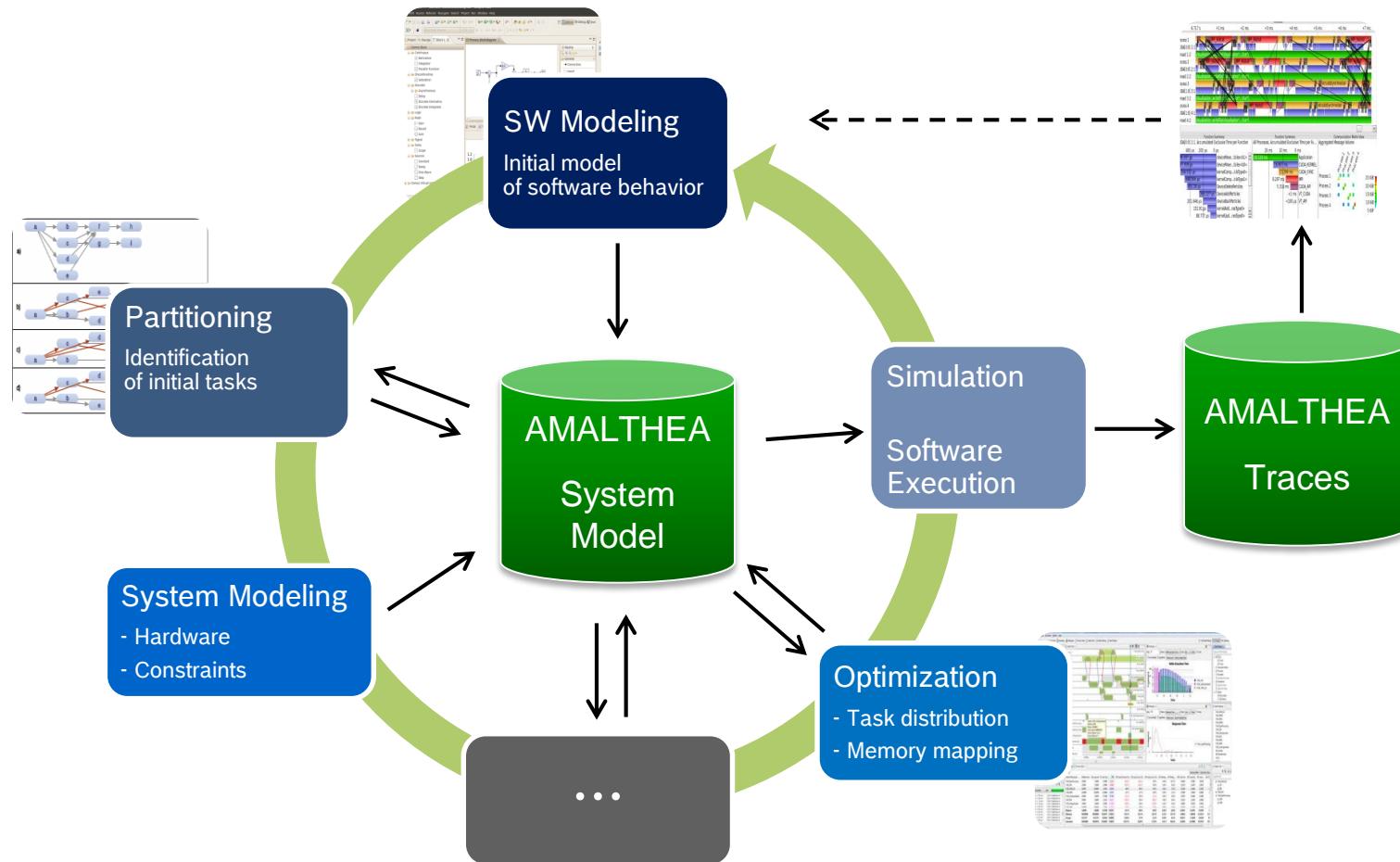


Example of a timing / scheduling simulation*

*Commercial tool – not part of the open source project

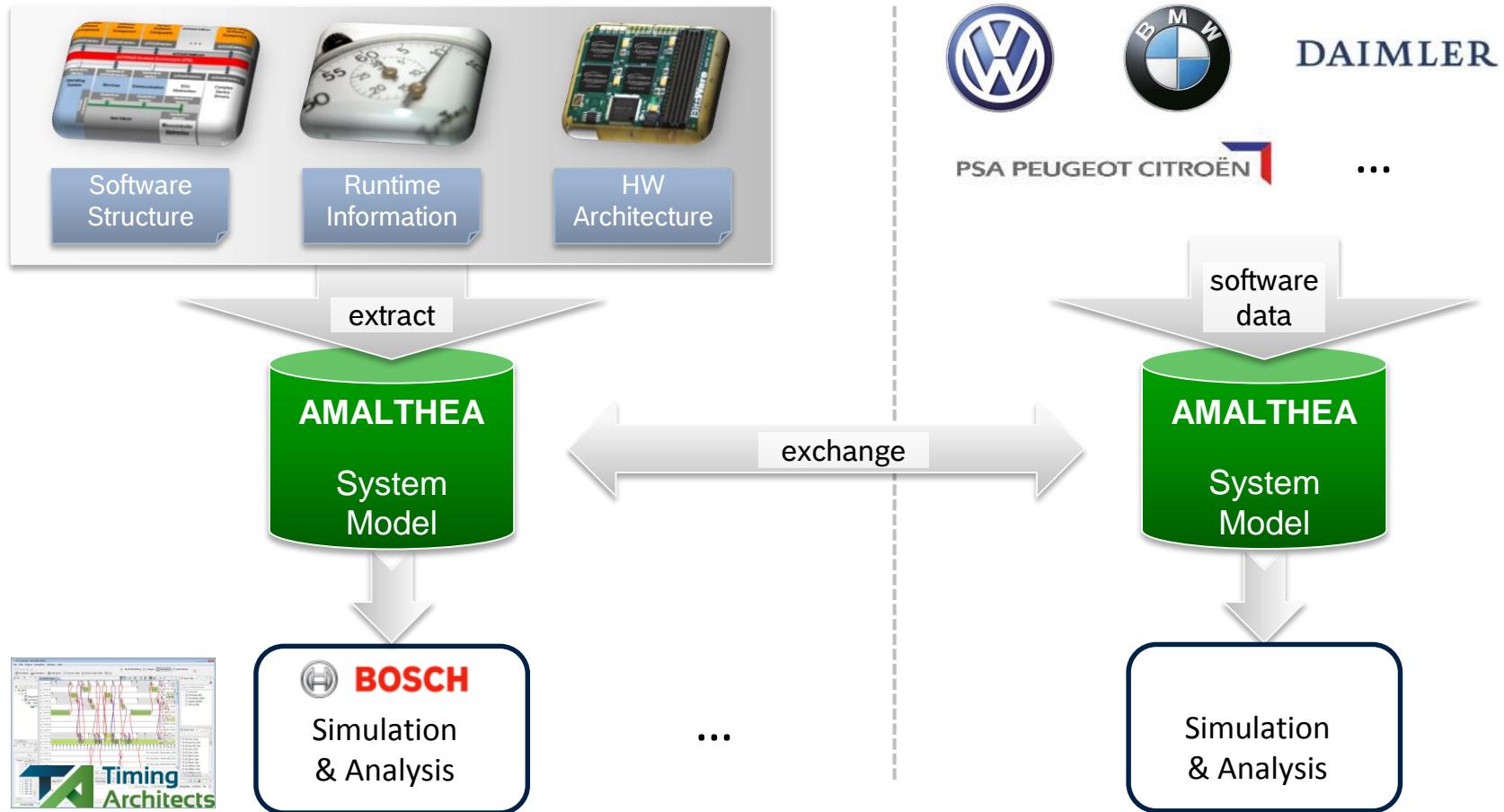
Tool platform AMALTHEA

Processing, Simulation and Analysis



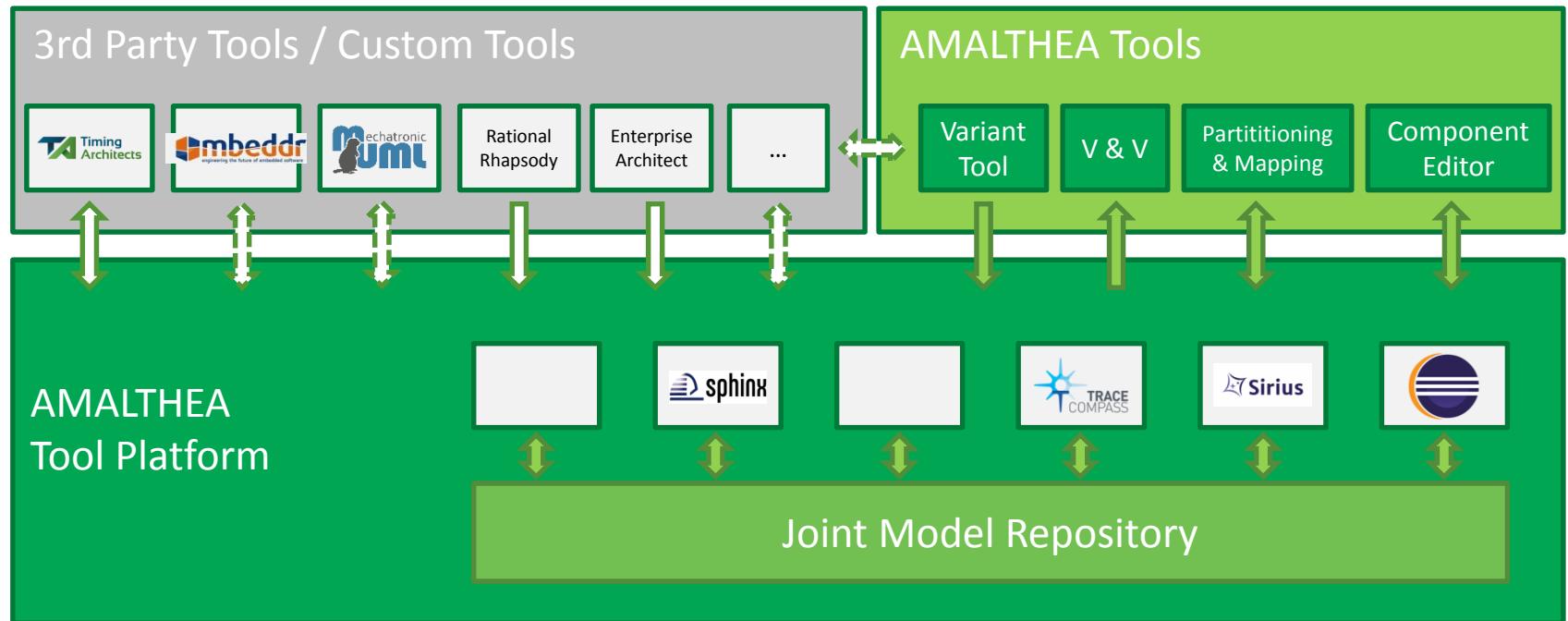
Tool platform AMALTHEA

Use cases @ BOSCH



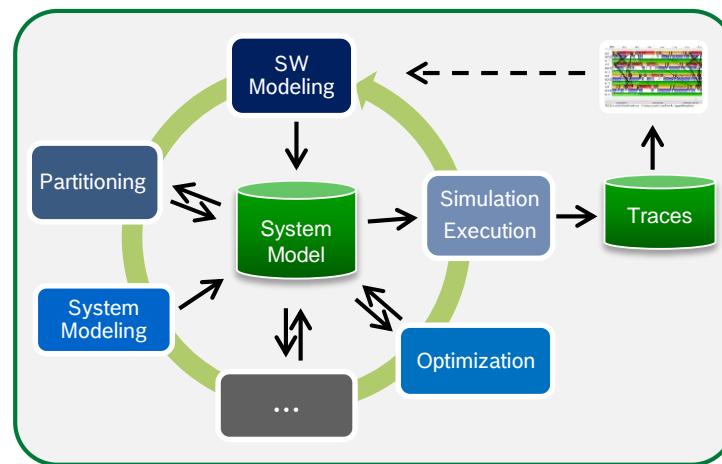
Tool platform AMALTHEA

Platform Architecture



Tool platform AMALTHEA

Connection to other Eclipse Projects

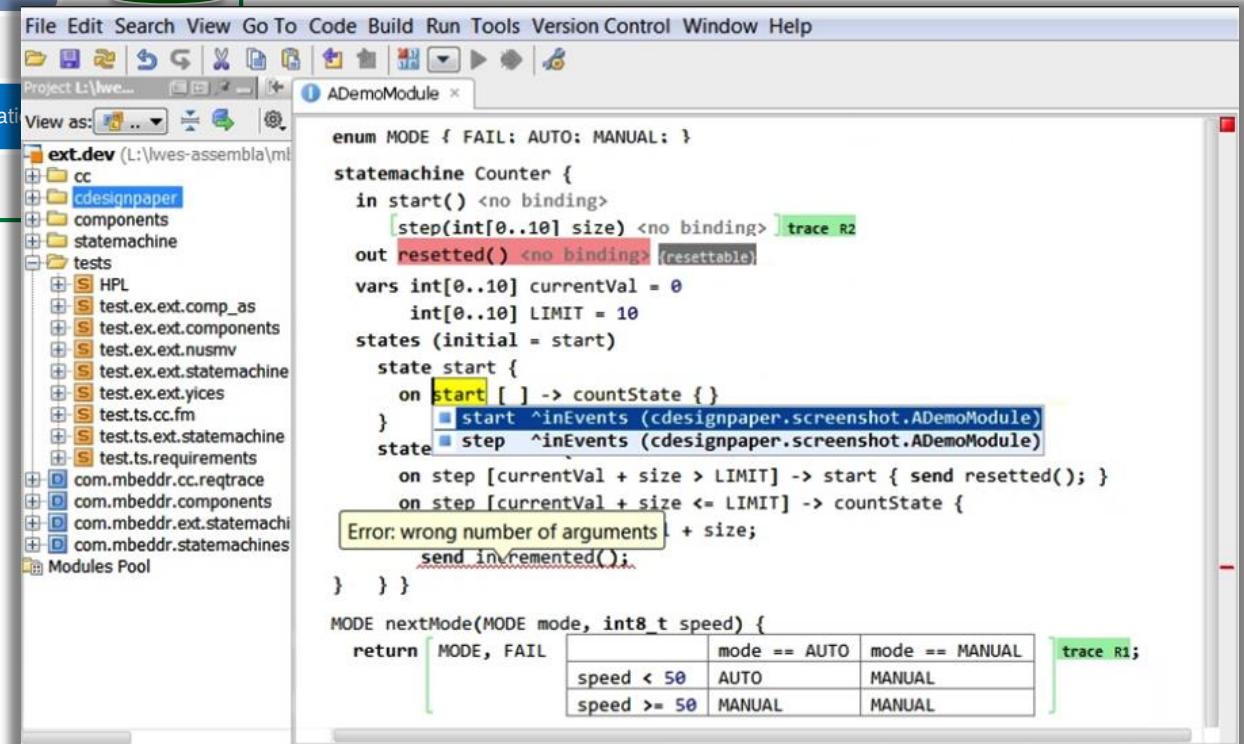
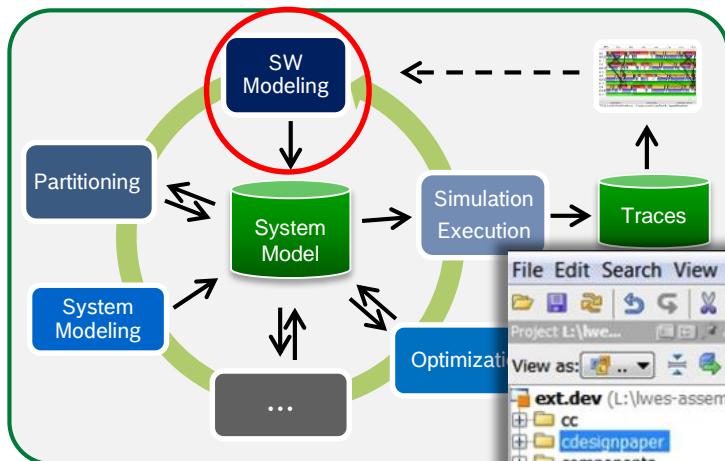


Francia



Tool platform AMALTHEA

Connection to other Eclipse Projects



The screenshot shows the AMALTHEA IDE interface. The top menu bar includes File, Edit, Search, View, Go To, Code, Build, Run, Tools, Version Control, Window, and Help. The left sidebar shows a project structure with 'ext.dev' as the root, containing 'cc', 'cdesignpaper', 'components', 'statemachine', 'tests' (with sub-folders 'HPL', 'test.ex.ext.comp_as', 'test.ex.ext.components', 'test.ex.ext.nusmv', 'test.ex.ext.statemachine', 'test.ex.ext.yices', 'test.ts.cc.fm', 'test.ts.ext.statemachine', 'test.ts.requirements', 'com.mbeddr.cc.retrace', 'com.mbeddr.components', 'com.mbeddr.ext.statemachines', and 'com.mbeddr.statemachines'), and 'Modules Pool'. The main editor area displays C code for a state machine 'Counter'. The code includes annotations for 'trace R2' and 'resettable'. A warning message 'Error: wrong number of arguments' is shown, with a suggestion to add 'size;'. The code also includes a table for the 'nextMode' function:

MODE, FAIL	mode == AUTO	mode == MANUAL
speed < 50	AUTO	MANUAL
speed >= 50	MANUAL	MANUAL

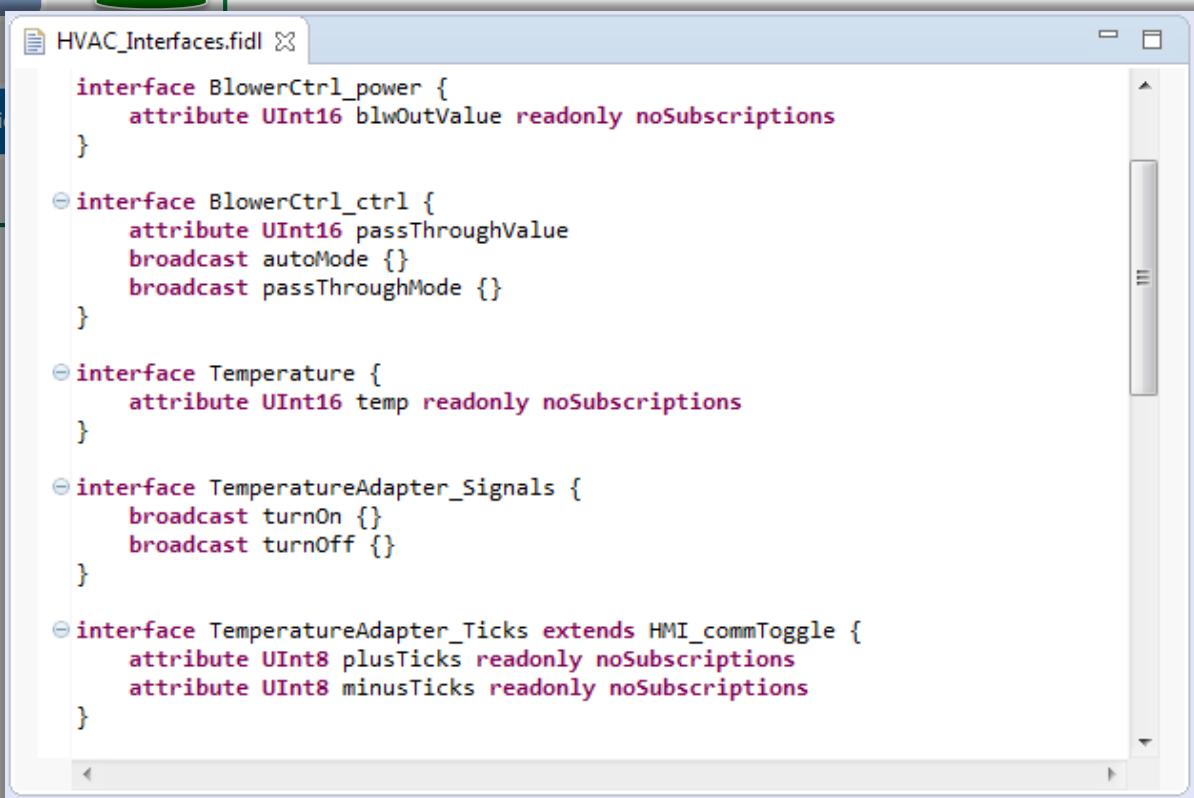
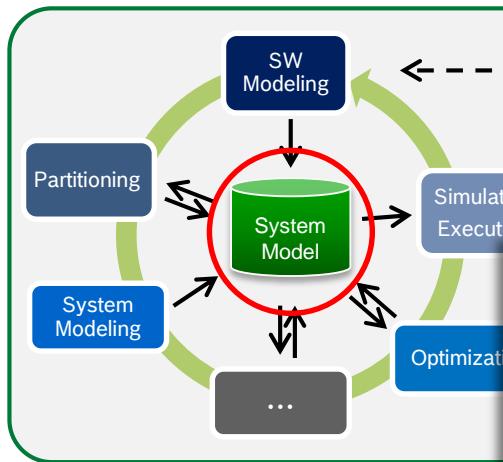
Annotations 'trace R1;' and 'trace R2;' are visible on the right side of the code editor.

planned extension

- Timing Annotations

Tool platform AMALTHEA

Connection to other Eclipse Projects



A screenshot of a code editor window titled 'HVAC_Interfaces.fidl'. The code is written in Franca, a domain-specific language for modeling interfaces. The code defines several interfaces:

```
interface BlowerCtrl_power {
    attribute UInt16 blwOutValue readonly noSubscriptions
}

interface BlowerCtrl_ctrl {
    attribute UInt16 passThroughValue
    broadcast autoMode {}
    broadcast passThroughMode {}
}

interface Temperature {
    attribute UInt16 temp readonly noSubscriptions
}

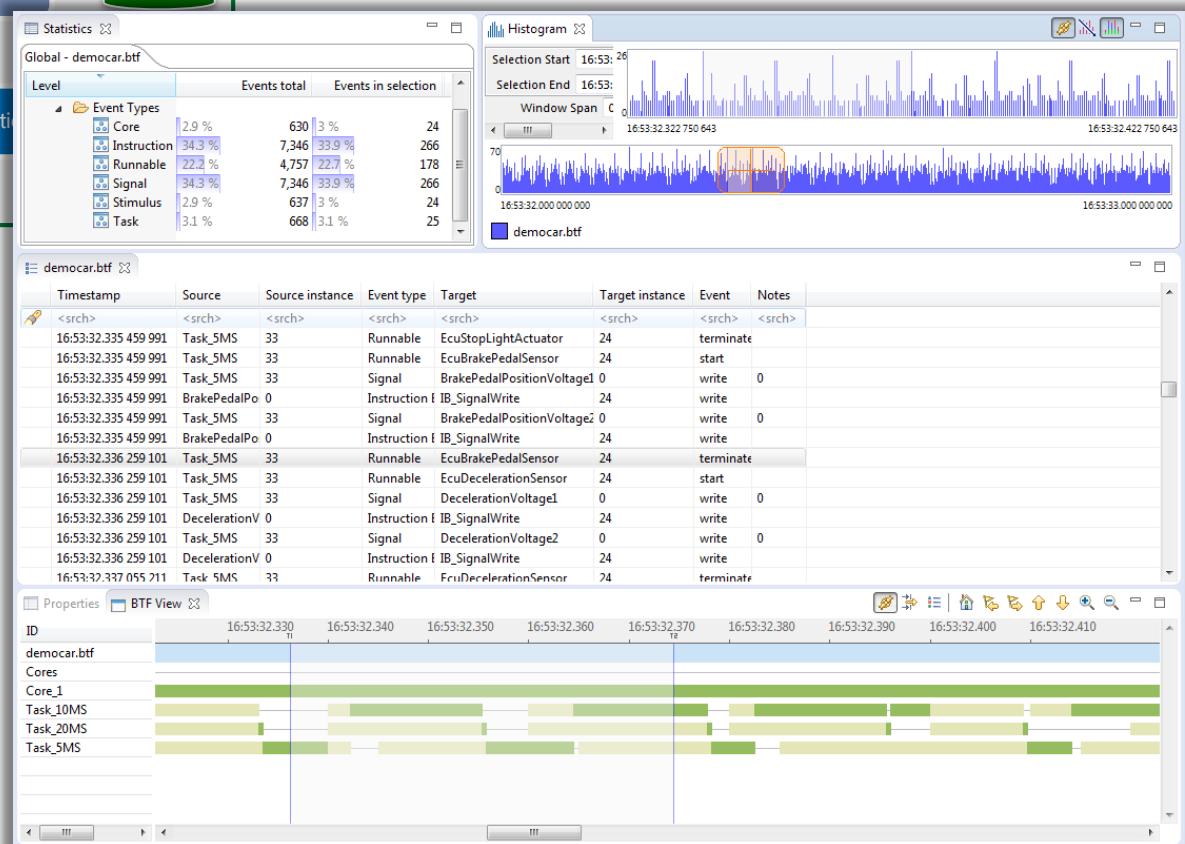
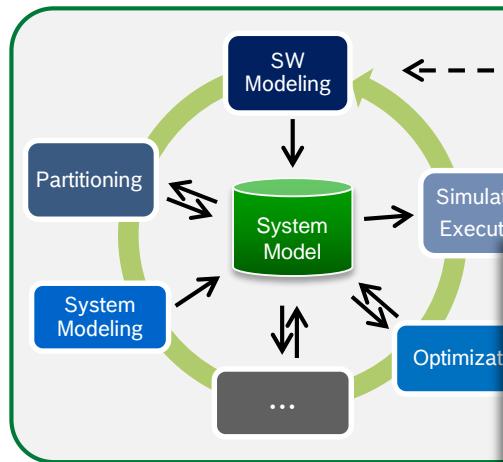
interface TemperatureAdapter_Signals {
    broadcast turnOn {}
    broadcast turnOff {}
}

interface TemperatureAdapter_Ticks extends HMI_commToggle {
    attribute UInt8 plusTicks readonly noSubscriptions
    attribute UInt8 minusTicks readonly noSubscriptions
}
```

to describe
▪ Interfaces

Tool platform AMALTHEA

Connection to other Eclipse Projects



implemented

- BTF Import
- BTF View

- **Eclipse IDE for Automotive Software Developers**



- **Xcore**

-> *Definition of the AMALTHEA Ecore model*

- **Sphinx**



-> *Workspace Handling, Validation, Tree Editor*

- **Xtend2**



-> *Code generation, model transformation*

- **Lyo**



-> OSLC

(separate Java libraries, no bundles, not included in orbit)

- **Ecore Tools**

-> *Class diagrams*

EcoreTools

(only limited connection to Xcore models)

- **Sirius**



-> *Graphical editors (first prototype)*

APP4MC – Application Platform Project for MultiCore

- AMALTHEA - Timeline and current project(s)
- Challenges for embedded multi- and many-core systems
- The AMALTHEA Platform
- **Demo / Screenshots of current release**
- APP4MC - Next steps

Platform Demo

AMALTHEA Model Editor



The screenshot displays the AMALTHEA Model Editor interface. At the top, a title bar shows the file "AMALTHEA_Democar.amxmi". Below the title bar, the "AMALTHEA Contents Tree" is visible, showing a hierarchical structure of the model elements. The central workspace is titled "Hardware Contents" and shows a detailed view of the hardware architecture, including a "Core_type" node and several "Ecu" nodes (Quartz, Processor, Memory, Core_1, Core_1_prescaler, Core_2, Core_2_prescaler). To the right, a panel titled "AMALTHEA_Democar.amxmi Contents" lists the contents of the model, including Software, Runnables, Labels, Tasks, ISRs, OS Events, Process Chains, Process Prototypes, Activations, Sections, Type Definitions, Tags, Modes, ModeLabels, and Custom Entities.

AMALTHEA_Democar.amxmi

AMALTHEA Contents Tree

AMALTHEA_Democar.amxmi Contents

This section enables the contents of this element to be edited.

- AMALTHEA
 - Software
 - Hardware
 - Operating System
 - Stimuli
 - Constraints
 - Events
 - Mapping <address type>
 - Configuration
 - Components

Hardware Contents

This section enables the contents of this element to be edited.

- Hardware
 - Core_type
- System
 - Ecu
 - Quartz
 - Processor
 - Memory
 - Core_1
 - Core_1_prescaler
 - Core_2
 - Core_2_prescaler

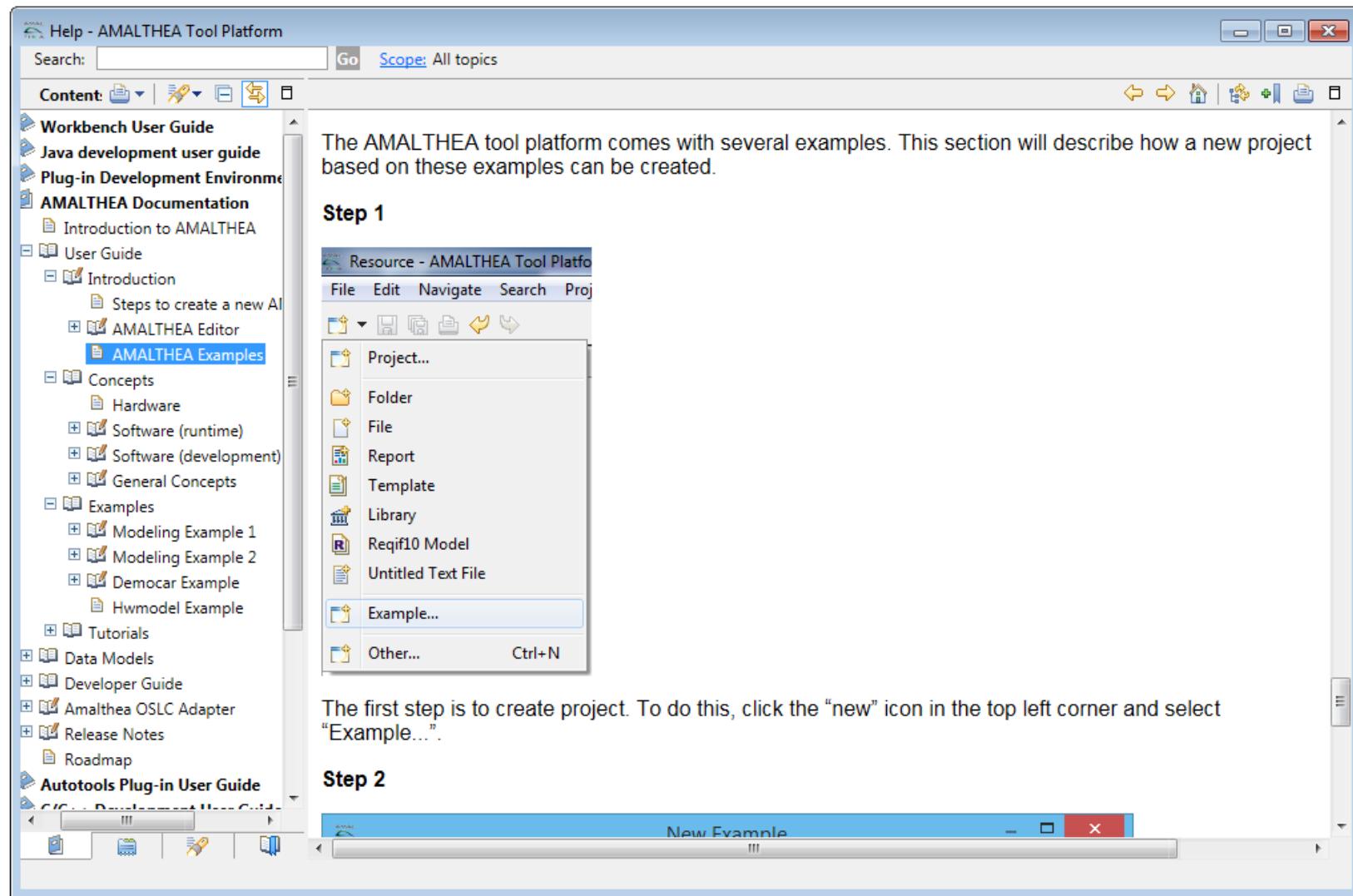
AMALTHEA_Democar.amxmi Contents

This section enables the contents of this element to be edited.

- Software
 - Runnables (43)
 - Labels (71)
 - Tasks (3)
 - ISRs (0)
 - OS Events (0)
 - Process Chains (0)
 - Process Prototypes (1)
 - Activations (0)
 - Sections (0)
 - Type Definitions (0)
 - Tags (10)
 - Modes (0)
 - ModeLabels (1)
 - Custom Entities (0)

Platform Demo

Eclipse Help – Examples



The AMALTHEA tool platform comes with several examples. This section will describe how a new project based on these examples can be created.

Step 1

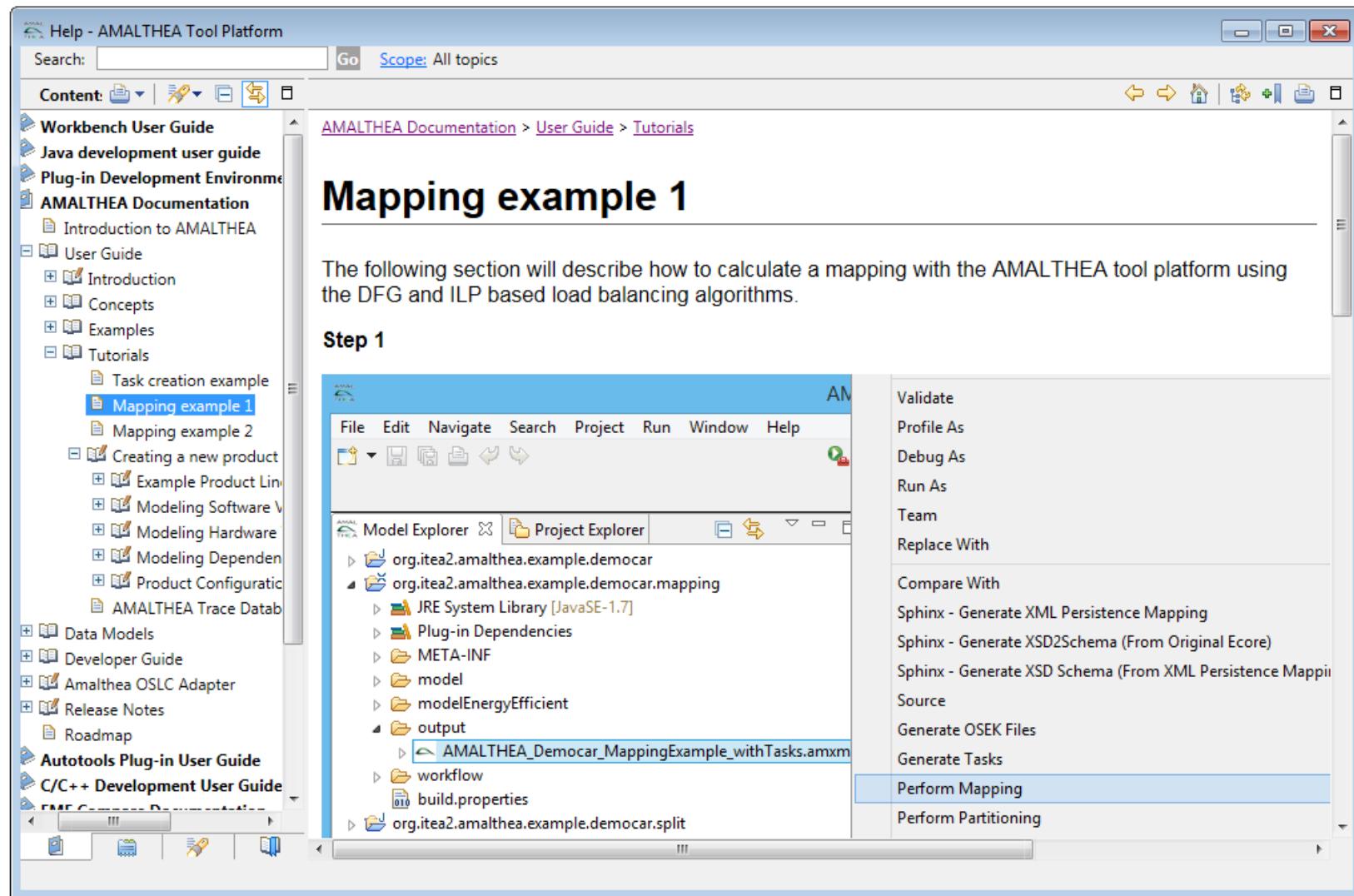
The first step is to create project. To do this, click the “new” icon in the top left corner and select “Example...”.

Step 2

The screenshot shows the AMALTHEA Tool Platform Help content on the left and a “New Example” dialog box on the right. The Help content displays the “AMALTHEA Examples” section under the “User Guide” category. The “New Example” dialog box shows a list of project types: Project..., Folder, File, Report, Template, Library, Reqif10 Model, Untitled Text File, Example..., and Other... (with “Ctrl+N” as a keyboard shortcut).

Platform Demo

Eclipse Help – Tutorials



The screenshot shows the AMALTHEA Tool Platform Help interface. The left sidebar contains a table of contents for the User Guide, with the 'Tutorials' section expanded to show 'Mapping example 1' as the selected item. The main content area displays the 'Mapping example 1' tutorial, which describes how to calculate a mapping using DFG and ILP based load balancing algorithms. Below the tutorial, the 'Step 1' section is visible. The center of the screen shows the Eclipse IDE interface with the 'Model Explorer' and 'Project Explorer' views. The 'Model Explorer' view shows a project structure for 'org.itea2.amalthea.example.democar.mapping' with subfolders like 'JRE System Library [JavaSE-1.7]', 'Plug-in Dependencies', 'META-INF', 'model', 'modelEnergyEfficient', 'output' (containing 'AMALTHEA_Democar_MappingExample_withTasks.amxmx'), 'workflow', and 'build.properties'. The 'Project Explorer' view shows the same project structure. The right side of the screen shows a context menu with options like 'Validate', 'Profile As', 'Debug As', 'Run As', 'Team', 'Replace With', 'Compare With', 'Sphinx - Generate XML Persistence Mapping', 'Sphinx - Generate XSD2Schema (From Original Ecore)', 'Sphinx - Generate XSD Schema (From XML Persistence Mapping)', 'Source', 'Generate OSEK Files', 'Generate Tasks', 'Perform Mapping' (which is highlighted in blue), and 'Perform Partitioning'.

Platform Demo

Eclipse Help – Data Models



Help - AMALTHEA Tool Platform

Search: Go Scope: All topics

Content 

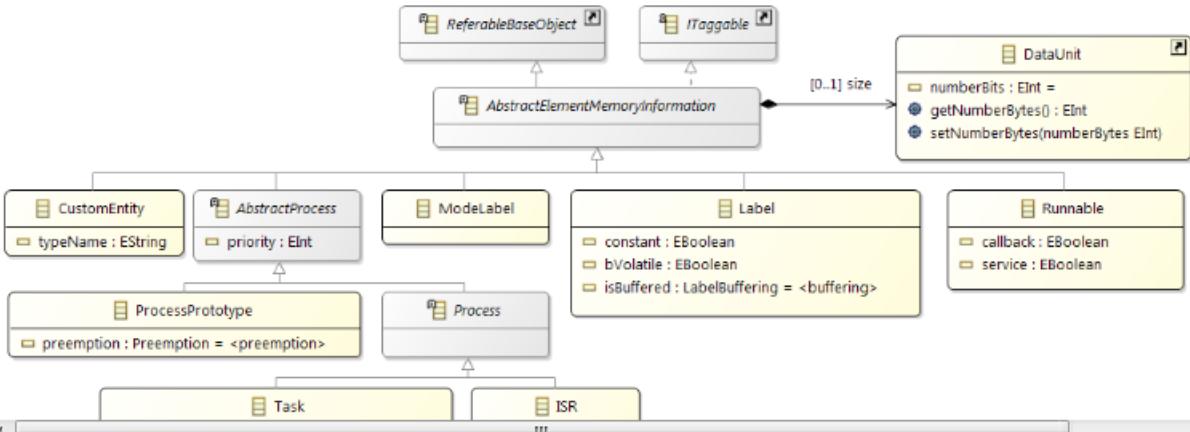
- Workbench User Guide
- Java development user guide
- Plug-in Development Environment
- AMALTHEA Documentation
 - Introduction to AMALTHEA
 - User Guide
 - Data Models
 - Data Model Overview
 - Hardware model
 - Common Model
 - Configuration Model
 - Constraints Model
 - Event Model
 - Mapping Model
 - OS Model
 - PropertyConstraints Model
 - Stimuli Model
 - Software Model
 - Components Model
 - Developer Guide
 - Amalthea OSLC Adapter
 - Release Notes
 - Roadmap
- Autotools Plug-in User Guide
- C/C++ Development User Guide
- EMF Compare Documentation
- Franca User Guide
- GNUTL Tool User Documentation

Software Model

The AMALTHEA software model is central accessible through the *SWModel* element. The namespace for the model is "http://www.amalthea.itea2.org/model/1.3.0/sw".

Memory Information

Analyzing and mapping the software structure to available memories needs additional information of the included elements. This type of information targets the consumed size of memory of an element, represented by the *size* attribute of type *DataUnit*. The element *AbstractElementMemoryInformation* is a generalized element that provides this data. The following image shows the structure and also the elements of the software model that are extending *AbstractElementMemoryInformation* (the overview picture is only showing the hierarchy and not possible relationships between the elements):



Platform Demo

Eclipse Help – Developer Guide



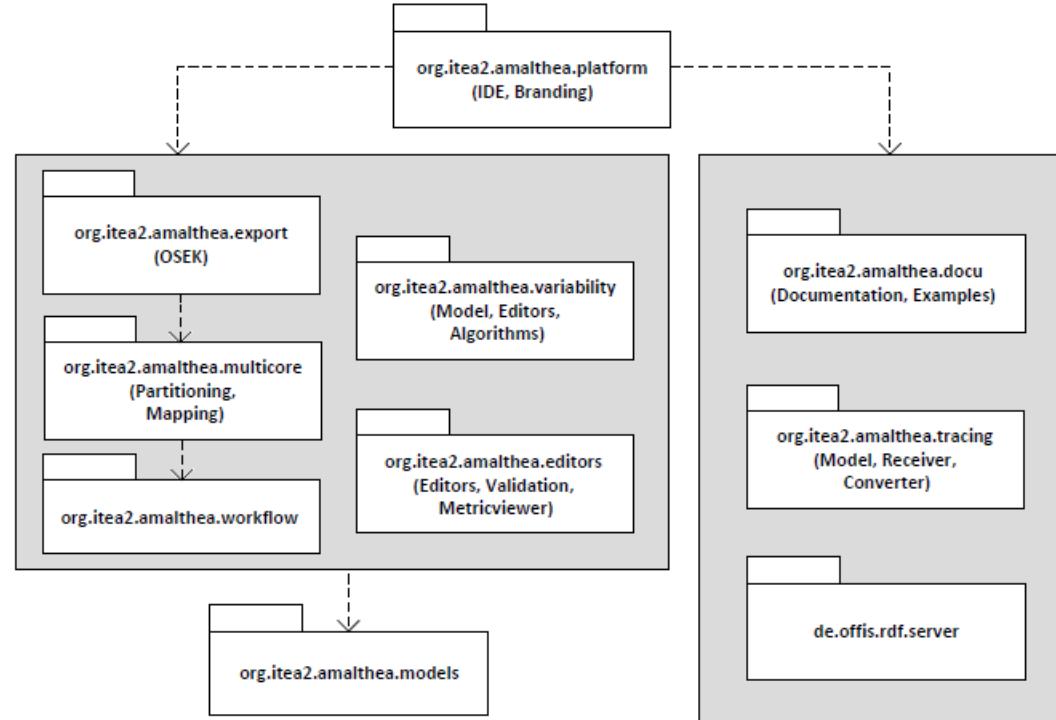
Help - AMALTHEA Tool Platform

Search: Go Scope: All topics

Contents

- Workbench User Guide
- Java development user guide
- Plug-in Development Environment Guide
- AMALTHEA Documentation
 - Introduction to AMALTHEA
 - User Guide
 - Data Models
 - Developer Guide
 - Overview of Features and Plug-ins
 - Model Workflow
 - Validation
 - AMALTHEA Trace Database
 - Partitioning
 - AMALTHEA Model XSD Schema
 - Amalthea OSLC Adapter
 - Release Notes
 - Roadmap
 - Autotools Plug-in User Guide
 - C/C++ Development User Guide
 - EMF Compare Documentation
 - Franca User Guide
 - GNU Tools On-line Documentation
 - Scripting User Guide
 - Statechart Tools User Guide
 - Subversive User Guide
 - Xtend User Guide
 - Xtext Documentation

AMALTHEA Tool Platform Features

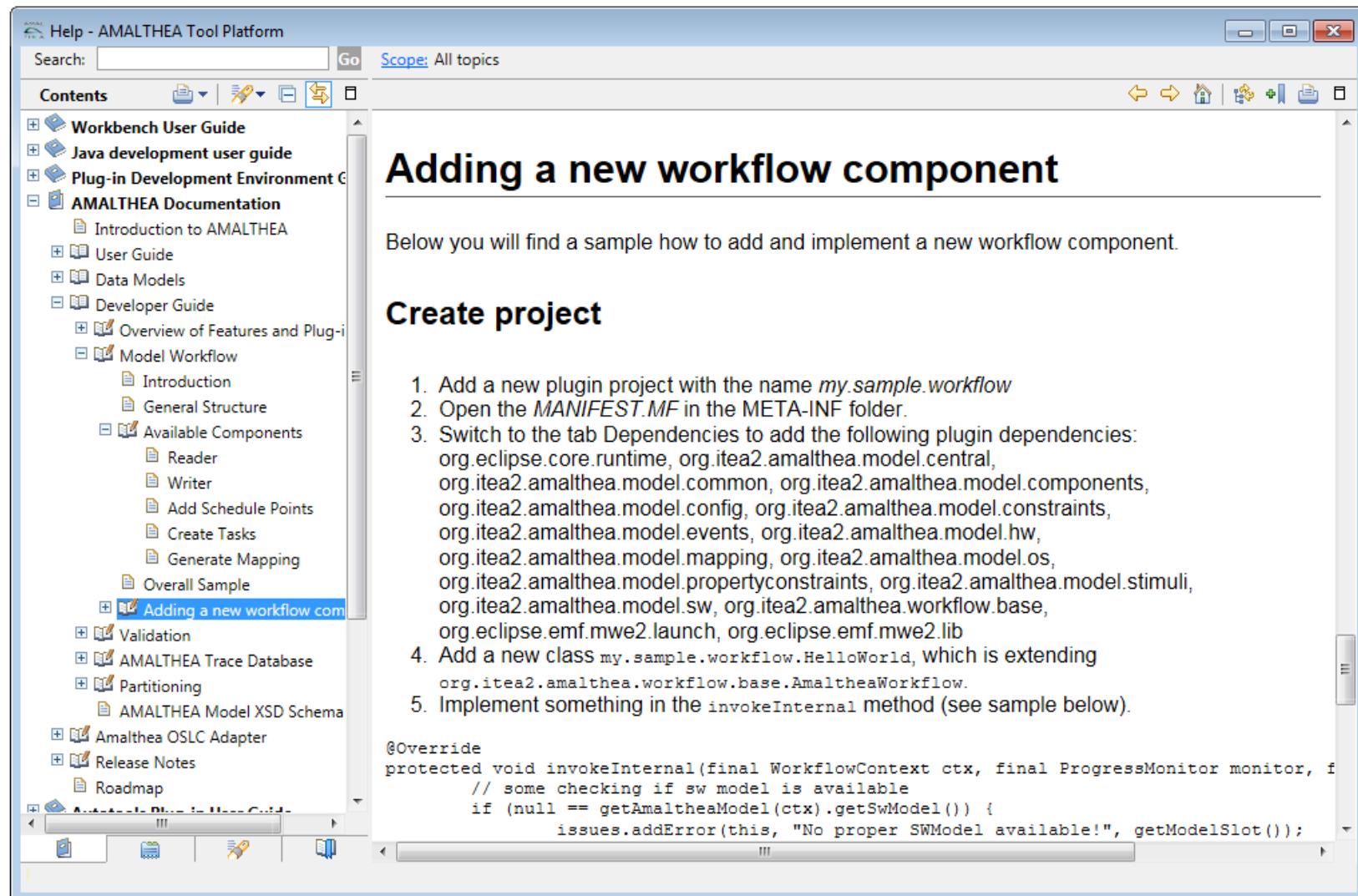


```
graph TD; subgraph Central [AMALTHEA Tool Platform Features]; A[org.itea2.amalthea.export (OSEK)]; B[org.itea2.amalthea.variability (Model, Editors, Algorithms)]; C[org.itea2.amalthea.multicore (Partitioning, Mapping)]; D[org.itea2.amalthea.workflow]; E[org.itea2.amalthea.models]; end; subgraph Top [ ]; F[org.itea2.amalthea.platform (IDE, Branding)]; end; subgraph Right [ ]; G[org.itea2.amalthea.docu (Documentation, Examples)]; H[org.itea2.amalthea.tracing (Model, Receiver, Converter)]; I[de.offis.rdf.server]; end; F -.-> A; F -.-> B; F -.-> C; F -.-> D; F -.-> E; G -.-> B; G -.-> H; G -.-> I;
```

For all of these features there also exists an SDK containing the sources. If you install the

Platform Demo

Eclipse Help – Developer Guide



The screenshot shows the Eclipse Help interface for the AMALTHEA Tool Platform. The left pane is a tree view of documentation topics, and the right pane is the content of the selected topic.

Contents

- Workbench User Guide
- Java development user guide
- Plug-in Development Environment
- AMALTHEA Documentation
 - Introduction to AMALTHEA
 - User Guide
 - Data Models
 - Developer Guide
 - Overview of Features and Plug-ins
 - Model Workflow
 - Introduction
 - General Structure
 - Available Components
 - Reader
 - Writer
 - Add Schedule Points
 - Create Tasks
 - Generate Mapping
 - Overall Sample
 - Adding a new workflow component
 - Validation
 - AMALTHEA Trace Database
 - Partitioning
 - AMALTHEA Model XSD Schema
 - Amalthea OSLC Adapter
 - Release Notes
 - Roadmap

Scope: All topics

Adding a new workflow component

Below you will find a sample how to add and implement a new workflow component.

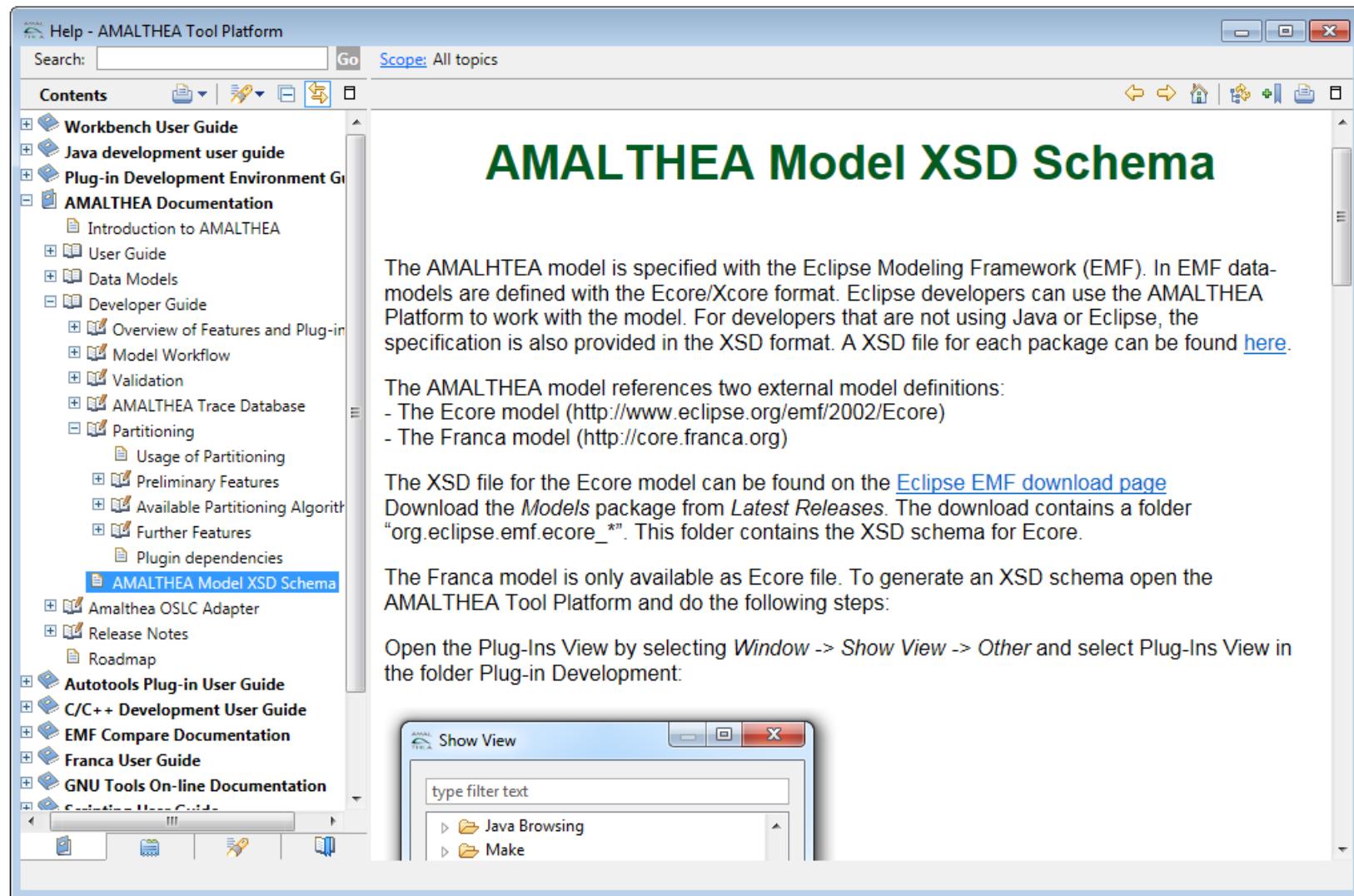
Create project

 1. Add a new plugin project with the name *my.sample.workflow*
 2. Open the *MANIFEST.MF* in the META-INF folder.
 3. Switch to the tab Dependencies to add the following plugin dependencies:
org.eclipse.core.runtime, org.itea2.amalthea.model.central,
org.itea2.amalthea.model.common, org.itea2.amalthea.model.components,
org.itea2.amalthea.model.config, org.itea2.amalthea.model.constraints,
org.itea2.amalthea.model.events, org.itea2.amalthea.model.hw,
org.itea2.amalthea.model.mapping, org.itea2.amalthea.model.os,
org.itea2.amalthea.model.propertyconstraints, org.itea2.amalthea.model.stimuli,
org.itea2.amalthea.model.sw, org.itea2.amalthea.workflow.base,
org.eclipse.emf.mwe2.launch, org.eclipse.emf.mwe2.lib
 4. Add a new class *my.sample.workflow.HelloWorld*, which is extending *org.itea2.amalthea.workflow.base.AmaltheaWorkflow*.
 5. Implement something in the *invokeInternal* method (see sample below).

```
@Override
protected void invokeInternal(final WorkflowContext ctx, final ProgressMonitor monitor, f
    // some checking if sw model is available
    if (null == getAmaltheaModel(ctx).getSwModel()) {
        issues.addError(this, "No proper SWModel available!", getModelSlot());
    }
}
```

Platform Demo

Eclipse Help – Developer Guide



The AMALTHEA model is specified with the Eclipse Modeling Framework (EMF). In EMF data-models are defined with the Ecore/Xcore format. Eclipse developers can use the AMALTHEA Platform to work with the model. For developers that are not using Java or Eclipse, the specification is also provided in the XSD format. A XSD file for each package can be found [here](#).

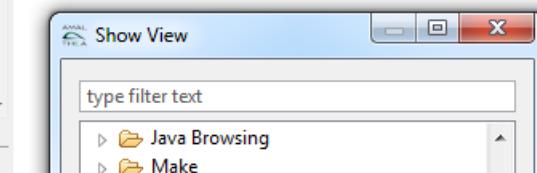
The AMALTHEA model references two external model definitions:

- The Ecore model (<http://www.eclipse.org/emf/2002/Ecore>)
- The Franca model (<http://core.franca.org>)

The XSD file for the Ecore model can be found on the [Eclipse EMF download page](#). Download the *Models* package from *Latest Releases*. The download contains a folder "org.eclipse.emf.ecore_*". This folder contains the XSD schema for Ecore.

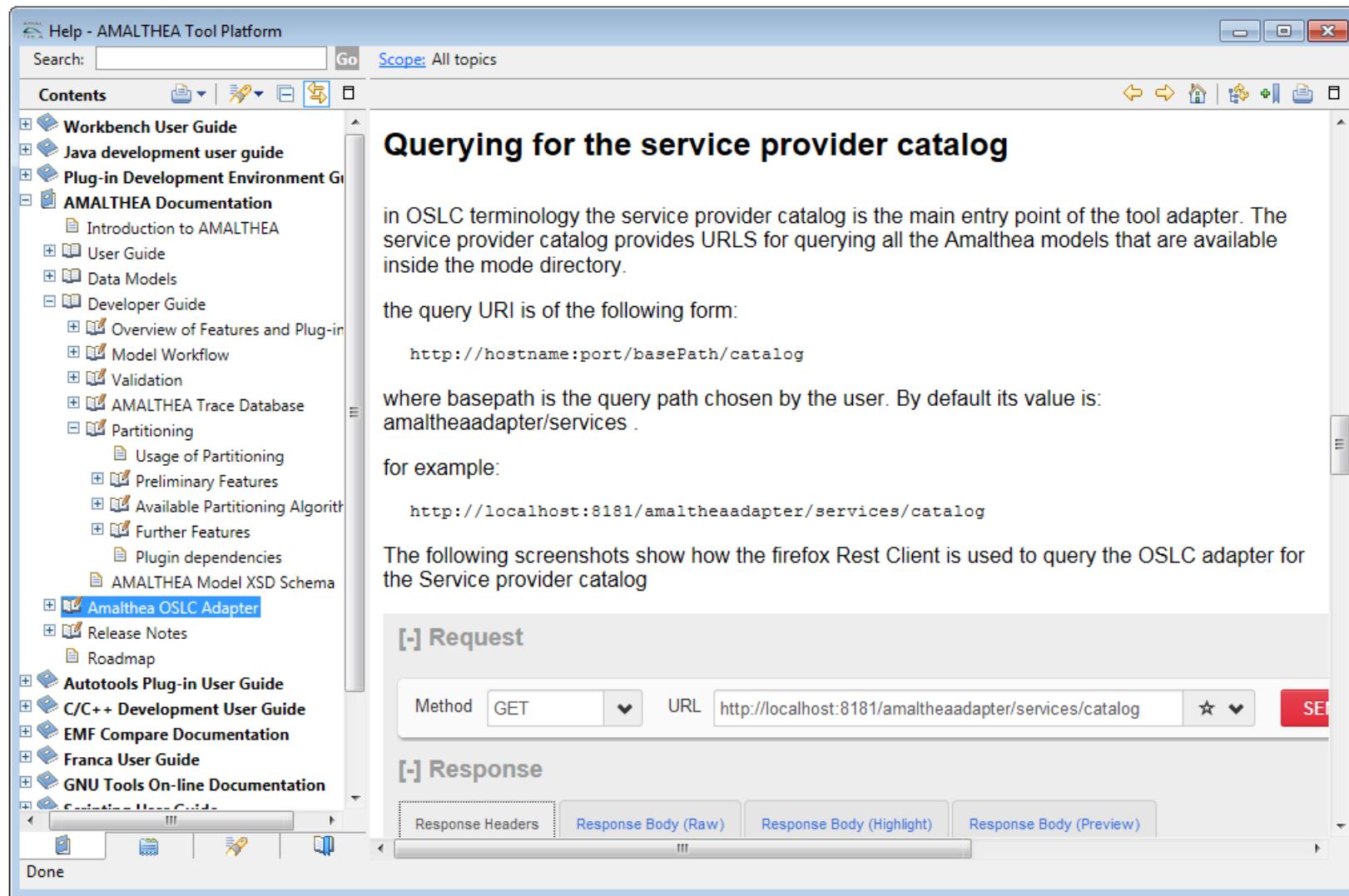
The Franca model is only available as Ecore file. To generate an XSD schema open the AMALTHEA Tool Platform and do the following steps:

Open the Plug-Ins View by selecting *Window -> Show View -> Other* and select *Plug-Ins View* in the folder *Plug-in Development*:



Platform Demo

Eclipse Help – Developer Guide



The screenshot shows the Eclipse Help interface for the AMALTHEA Tool Platform. The left sidebar contains a table of contents with various documentation sections. The main content area is titled "Querying for the service provider catalog". It explains that in OSLC terminology, the service provider catalog is the main entry point of the tool adapter. It provides a query URI template: `http://hostname:port/basePath/catalog`, where `basepath` is the query path chosen by the user. By default, its value is `amaltheaadapter/services`. An example is given: `http://localhost:8181/amaltheaadapter/services/catalog`. Below this, a screenshot of the Firefox REST Client is shown, demonstrating the request to the service provider catalog.

Contents

- Workbench User Guide
- Java development user guide
- Plug-in Development Environment Guide
- AMALTHEA Documentation
 - Introduction to AMALTHEA
 - User Guide
 - Data Models
 - Developer Guide
 - Overview of Features and Plug-in
 - Model Workflow
 - Validation
 - AMALTHEA Trace Database
 - Partitioning
 - Usage of Partitioning
 - Preliminary Features
 - Available Partitioning Algorithm
 - Further Features
 - Plugin dependencies
 - AMALTHEA Model XSD Schema
 - Amalthea OSLC Adapter
 - Release Notes
 - Roadmap
 - Autotools Plug-in User Guide
 - C/C++ Development User Guide
 - EMF Compare Documentation
 - Franca User Guide
 - GNU Tools On-line Documentation
 - System Model Guide

Search: Go Scope: All topics

Querying for the service provider catalog

in OSLC terminology the service provider catalog is the main entry point of the tool adapter. The service provider catalog provides URLs for querying all the Amalthea models that are available inside the mode directory.

the query URI is of the following form:

```
http://hostname:port/basePath/catalog
```

where `basepath` is the query path chosen by the user. By default its value is:
`amaltheaadapter/services` .

for example:

```
http://localhost:8181/amaltheaadapter/services/catalog
```

The following screenshots show how the firefox Rest Client is used to query the OSLC adapter for the Service provider catalog

[-] Request

Method: GET URL: `http://localhost:8181/amaltheaadapter/services/catalog`

[-] Response

Response Headers | Response Body (Raw) | Response Body (Highlight) | Response Body (Preview)

APP4MC – Application Platform Project for MultiCore

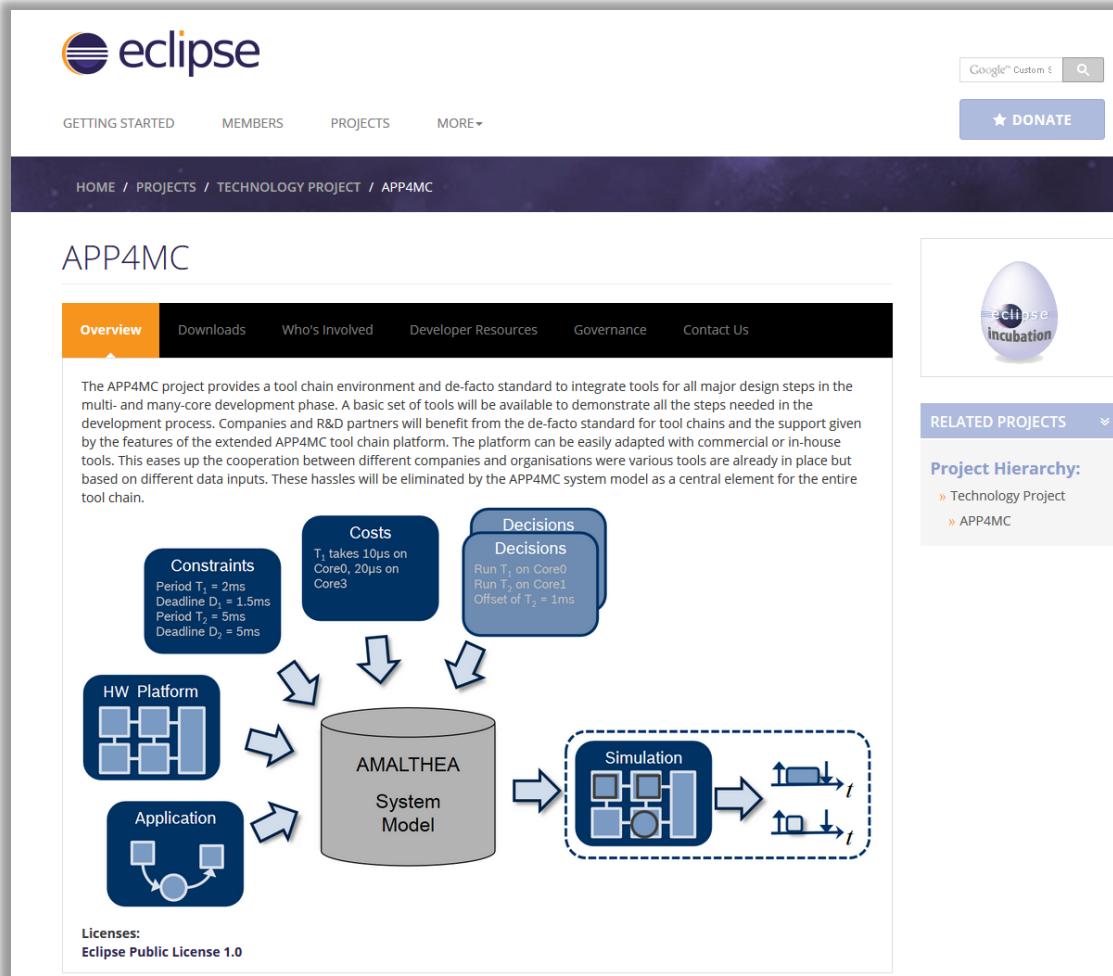
- AMALTHEA - Timeline and current project(s)
- Challenges for embedded multi- and many-core systems
- The AMALTHEA Platform
- Demo / Screenshots of current release
- **APP4MC - Next steps**

Project Activities

Eclipse Project APP4MC



<http://projects.eclipse.org/projects/technology.app4mc>



The APP4MC project provides a tool chain environment and de-facto standard to integrate tools for all major design steps in the multi- and many-core development phase. A basic set of tools will be available to demonstrate all the steps needed in the development process. Companies and R&D partners will benefit from the de-facto standard for tool chains and the support given by the features of the extended APP4MC tool chain platform. The platform can be easily adapted with commercial or in-house tools. This eases up the cooperation between different companies and organisations were various tools are already in place but based on different data inputs. These hassles will be eliminated by the APP4MC system model as a central element for the entire tool chain.

Constraints
Period $T_1 = 2\text{ms}$
Deadline $D_1 = 1.5\text{ms}$
Period $T_2 = 5\text{ms}$
Deadline $D_2 = 5\text{ms}$

Costs
 T_1 takes 10µs on Core0, 20µs on Core3

Decisions
Run T_1 on Core0
Run T_2 on Core1
Offset of $T_2 = 1\text{ms}$

HW Platform

Application

AMALTHEA System Model

Simulation

Licenses:
Eclipse Public License 1.0

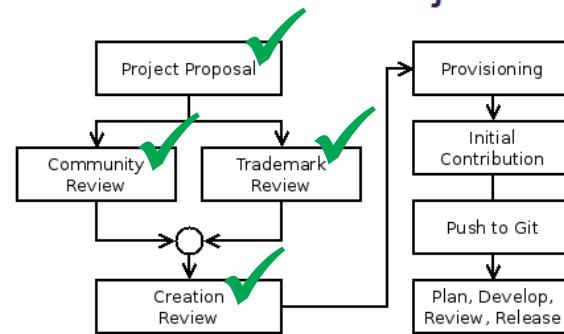
Project Activities

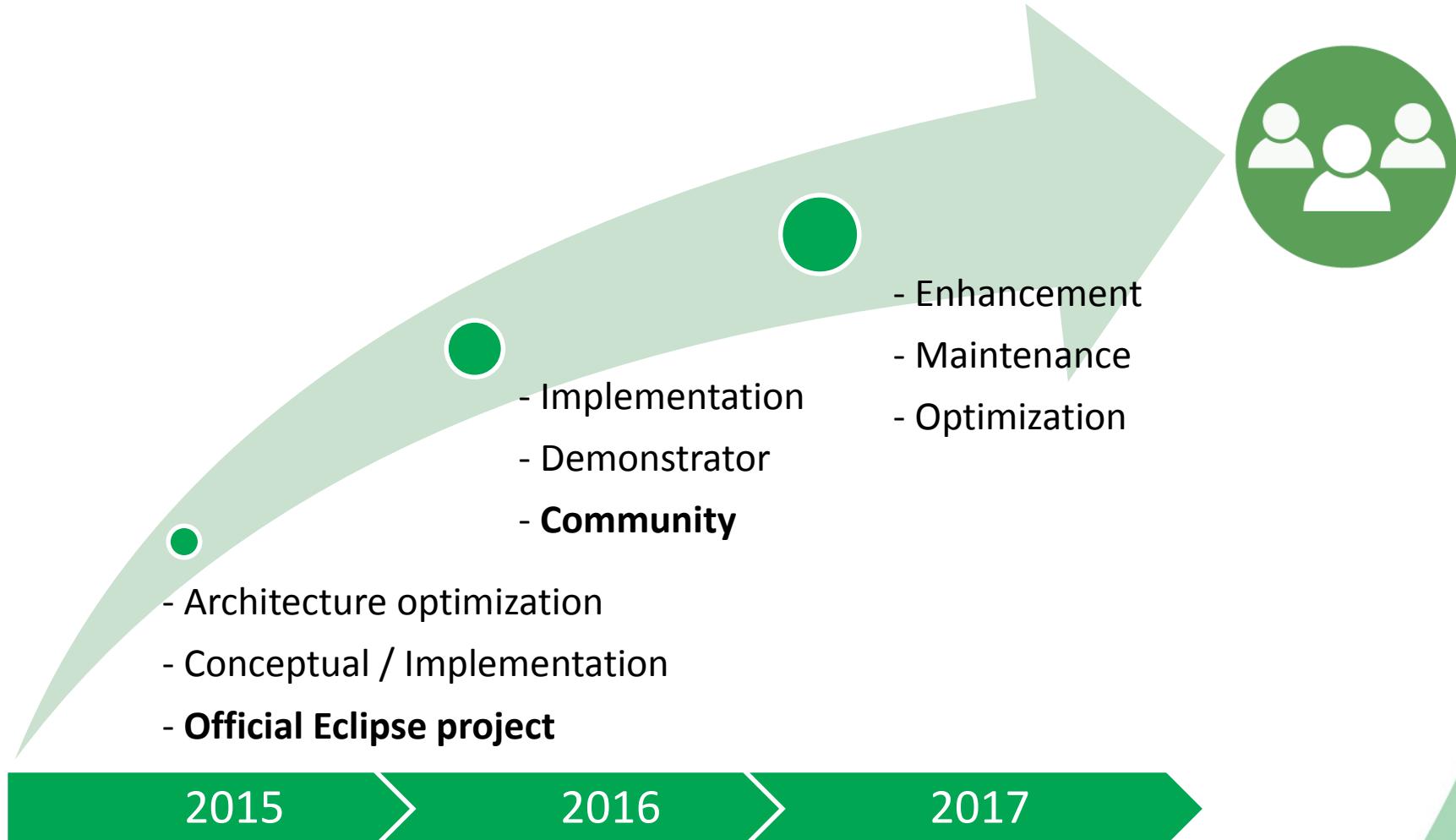
Eclipse Project APP4MC



- Open Source **Eclipse project** is created
- Current work: Committer Agreements, Code refactoring, IP clearance, ...

Overview of the Project Creation Process





A promotional poster for eclipsecon Europe 2015. The top half features the event logo (a stylized 'e' icon) and the text "eclipsecon Europe" in large, bold, blue and white letters, with "Ludwigsburg, Germany, 3 - 5 November 2015" below it. The bottom half is a photograph of a modern building with large glass windows and a bridge over a river in the foreground. Overlaid on the image are the numbers "+1", "0", and "-1" in large, bold, blue font, representing a rating scale.