art2kitekt: Modeling and Analyzing Multi-Domain Mixed-Criticality Applications

Sergio Sáez
ITI, Spain

EMC²: Mixed Criticality Applications and Implementation Approaches
HiPEAC 2015 - Amsterdam, January 20th, 2015

Outline

- Problem overview
- Goals and expected tool features
- Art2kitekt tool architecture
  - Domain-specific profiles
  - Interoperability and extendability
- Current status
- Expected results within EMC²

Problem overview

- Mixed-criticality applications from different domains have
  - Different kinds of execution platforms
    - Multiple processor connected by different communication channels (point-to-point links, NoC, etc)
    - Shared memory multiprocessors
  - Different criticality concept/models
    - Temporal isolation on shared priority spaces.
    - Spatial and temporal isolation by using partitioned systems
  - Different certification needs

➔ There is no analysis method that addresses **every** aspect in **every** application domain

Main goal

To allow the engineer (the **user**) to **model** only the kind of systems we (the **tool**) are able to **analyse**

E.g. in AADL and UML MARTE the designer can model systems for which there is no analysis techniques available

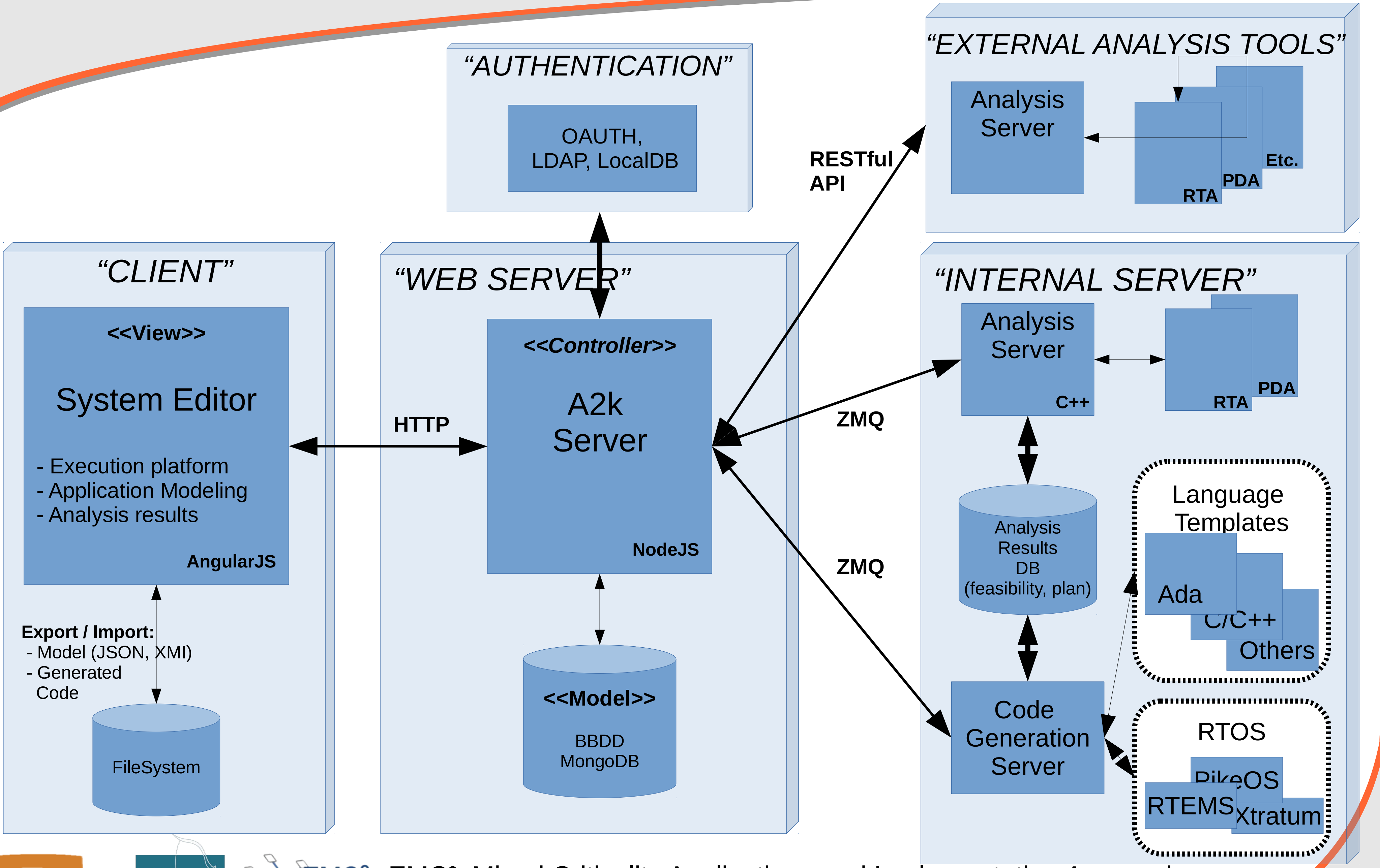
Expected features

- An integrated tool chain that allows the engineer to:
  - **Configure** the execution platform with the application specific details, e.g. physical devices, resources, RTOS overheads, ...
  - **Model** the software according to a domain-specific application model
  - **Specify** the criticality level of each software component
  - **Map** the software components to execution platform resources
  - **Analyse** extra-functional requirements of the system
  - **Generate** the low-level software code/configuration from the analysis results

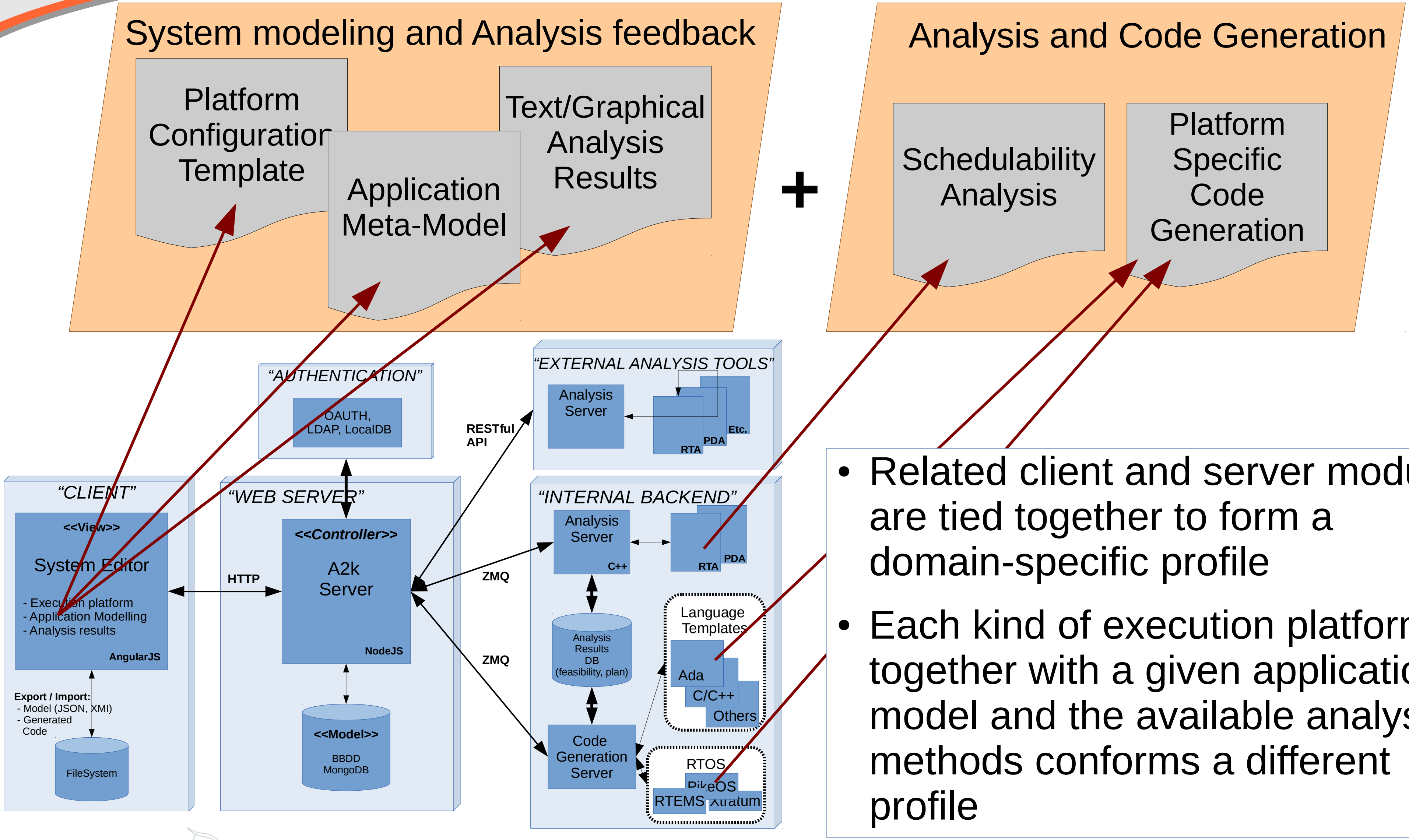
art2kitekt: A tool for building MCS

- Application domain profiles
  - Execution platform, application model and analysis methods are strongly coupled.
  - Different platform/model/analysis profiles will be provided for each kind of system
- Interoperability and extendability
  - Interoperability with external tools should be possible, e.g. WCET analysis, high-level application modeling, etc.
  - Importing/exporting system models using common formats, e.g. XMI, JSON, ...
  - Data-binding and APIs for common tool programming languages, e.g. C/C++, Ada, PHP, Python, ...
- A simple and fast tool deployment based on web technologies

Tool architecture



Domain-specific profiles



- Related client and server modules are tied together to form a domain-specific profile
- Each kind of execution platform together with a given application model and the available analysis methods conforms a different profile

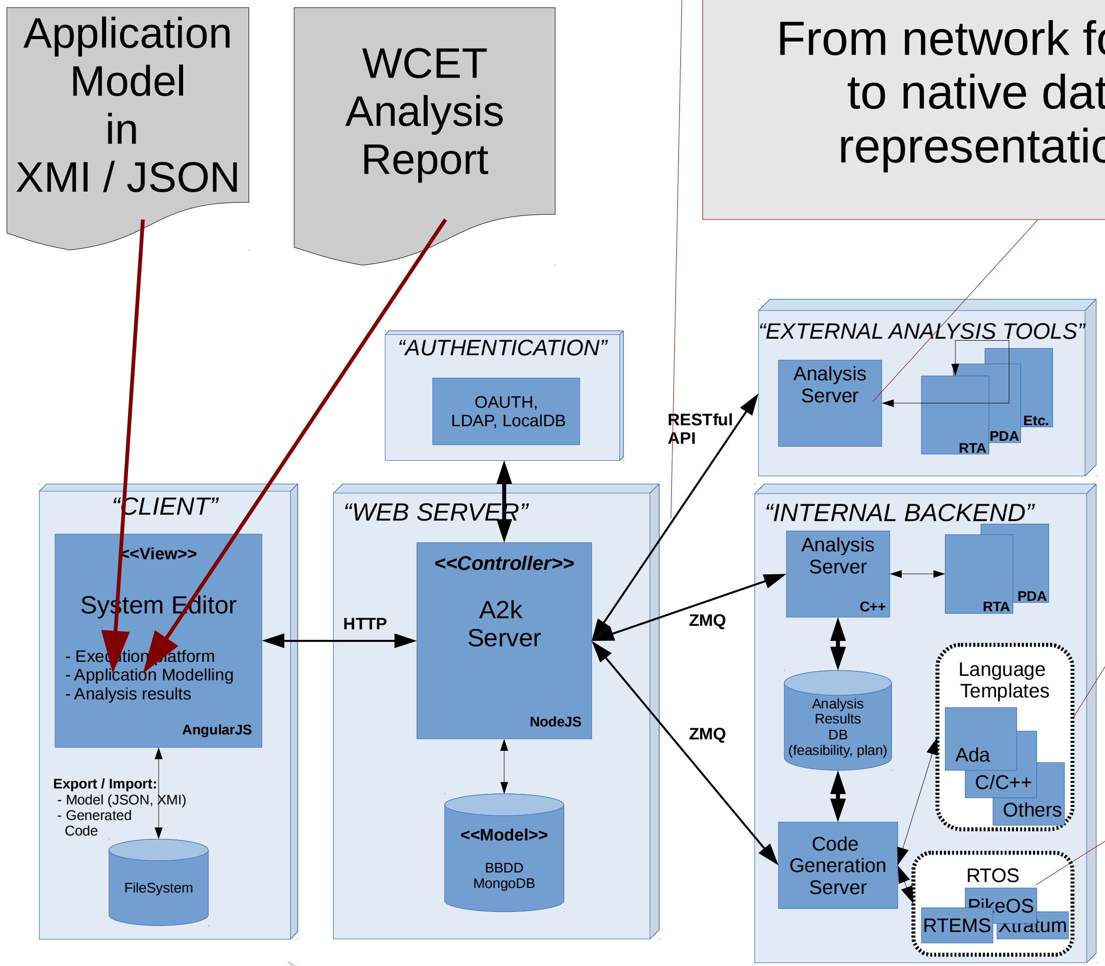
Domain-specific profiles

Example profiles

	Healthcare domain profile	**Space domain profile**
Task Model	• Communicating tasks modeled as end-to-end flows	• Tasks with data and control dependencies using shared resources
Criticality model	• Priority based time isolation with overrun detection and operational mode changes	• Temporal and spatial isolation based on time/space partitioning
Prog. Lang.	• C language over a POSIX interface	• Ada Ravenscar profile
RTOS	• RTEMS operating system	• ORK+Linux over XtratuM hypervisor
Exec. Platf.	• Several processors with point-to-point links	• Shared memory symmetric multiprocessor

Interoperability and extendability

Input data from different tools



Automatic data-binding

From network format to native data representation

External analysis tools

- RESTful plugins with capability negotiation
- Automatic Data-Binding for internal plugins

Programming languages

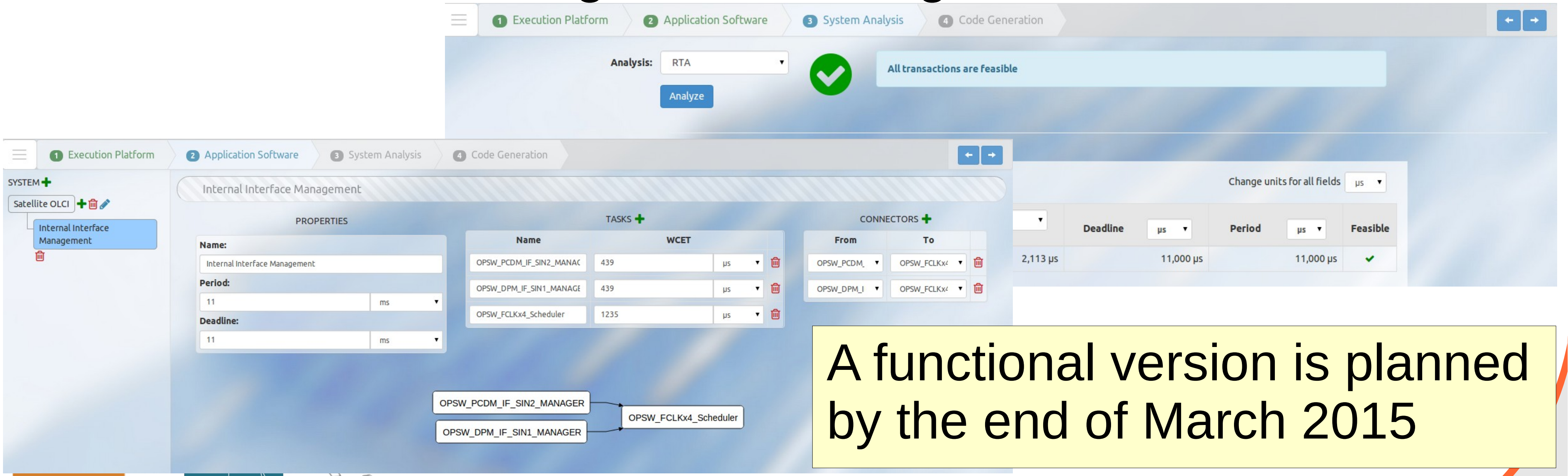
Templates for code generation in new programming languages

Operating Systems

New RTOS support through overhead datasheets and execution models

Current status

- A basic profile is under development
  - Execution platform based on Linux SMP platform
  - Schedulability analysis based on Response Time Analysis
  - C/POSIX code generation engine



Analysis: RTA

All transactions are feasible

Change units for all fields: μs

	Deadline μs	Period μs	Feasible
2,113 μs	11,000 μs	11,000 μs	✓

Internal Interface Management

PROPERTIES

Name: Internal Interface Management

Period: 11 ms

Deadline: 11 ms

TASKS

Name	WCET	
OPSW_PCDM_IF_SIN2_MANAC	439	μs
OPSW_DPM_IF_SIN1_MANAGE	439	μs
OPSW_FCLKx4_Scheduler	1235	μs

CONNECTORS

From	To
OPSW_PCDM_	OPSW_FCLKx4
OPSW_DPM_I	OPSW_FCLKx4

A functional version is planned by the end of March 2015

Expected results

- Complete implementation of two Space domain profiles
  1. Single core platform
     - Task with dependencies and shared resources
     - Priority based scheduling
     - RTEM OS
  2. Multiprocessor platform
     - Parallelization of task
     - Some details still pending
       - Priority or time triggered scheduling?
       - RTEMS OS?

Thank you for your attention!

Questions?