From Simulink to RT tools
June 15, 2010

June 15, 2010 - EFDA Feedback Control Working Group Meeting

G. De Tommasi[1] A. Barbalace[2] A. Neto[3] F. Sartori[4]
L. Zabeo[5] and EFDA-JET PPCC contributors
[1]EURATOM-ENEA-CREATE, Università di Napoli Federico II
[2]Consorzio RFX
[3]Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear
[4]Fusion For Energy [5]ITER Organization

Outline

Motivations

A brief history

From Simulink to Real-Time Environments

Motivations

Main aim

- **Reduction of the time needed for commissioning on the machine**
- **Reduction of the risk on the plant**

Requirements

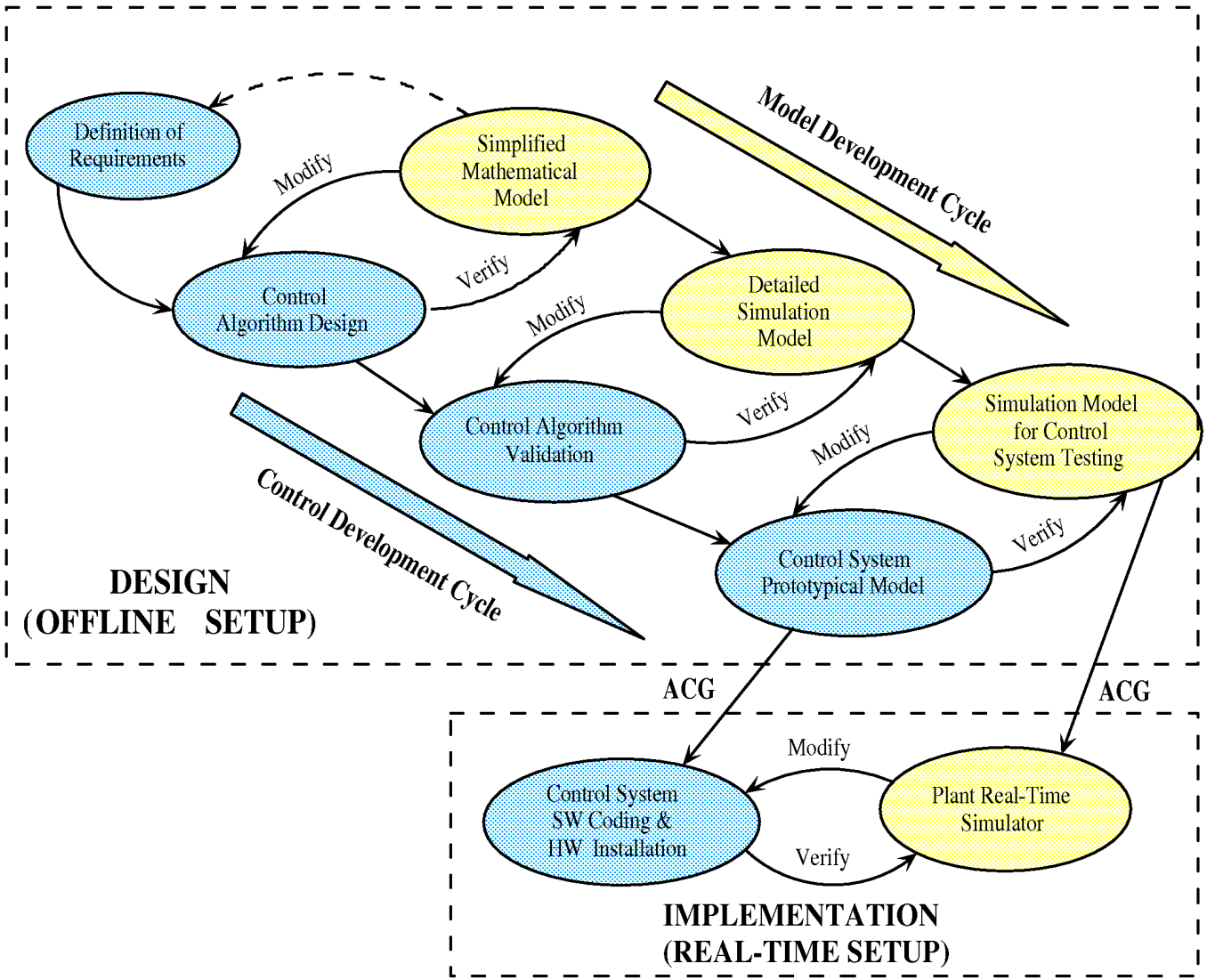
Requirements for the RT framework

- Standard architecture for real-time control systems
- Complete separation between the algorithmic part of a real-time application from the plant-interface software

Requirements for the design environment

- Model based design
- Validation via simulation
- Tools for the rapid prototyping of the real-time application

Design aided with modeling, simulation and rapid prototyping tool



Development phase

- The design and validation of the control algorithm was carried out in Matlab/Simulink environment (XSC Tools [1])
- The control algorithm were coded "by-hands" in the **JETRT** framework [2] (MARTe ancestor)
- **During the implementation phase the control algorithm was revised**
- **The XSC Tools were updated in order to take into account these changes!**
- The final version the XSC Tools allows the user to generate an **XSC scenario** (i.e. the file that is loaded by the session leader when preparing the experiment)

G. De Tommasi et al.,

XSC Tools: a software suite for tokamak plasma shape control design and validation
IEEE Transactions on Plasma Science, vol. 35(3), Jun. 2007

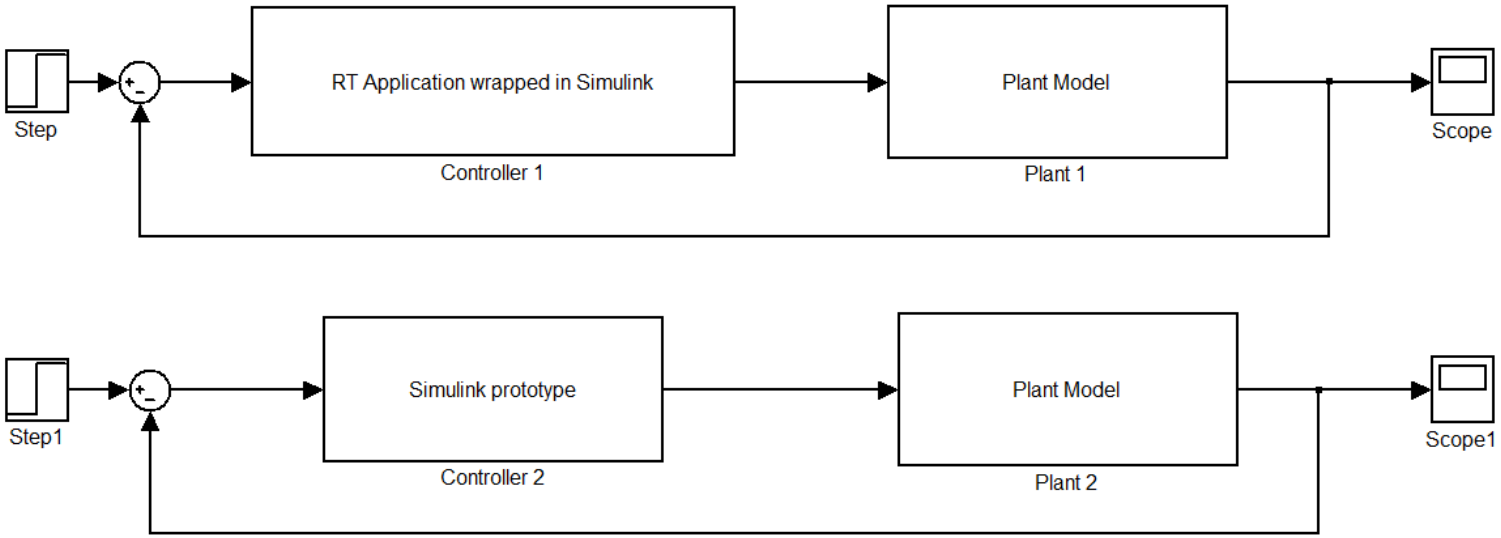
G. De Tommasi et al.,

A flexible software for real-time control in nuclear fusion experiments
Control Engineering Practice, vol. 14(11), Nov. 2006

Commissioning phase

- Each time a new *XSC scenario* has been released the tuning of the controller parameters and its validation has been performed in the Matlab/Simulink environment with the XSC Tools
- The real-time boundary reconstruction code (XLOC/Felix) has been plugged in the Simulink scheme in order to minimize the differences between the offline and the real-time environment
- **In this case we "embedded" the real-time code into Simulink**

From real-time environment to Simulink



"Embedding RT code into Simulink"

- In most of the cases to check the real-time version of the controller it is convenient to run it against the plant model in the Simulink environment.
- The real-time version and the Simulink version of the system can be run in *parallel* in order to perform the validation

Lessons learned from the XSC experience

Wishes

- Avoid implementation *by-hands* $\rightarrow$ automatic real-time code generation
- Allow to perform closed–loop validation with the real-time code

Limitations

- **JETRT** didn't provide a **real** separation between the algorithmic part of a real-time application from the plant-interface software
- **JETRT** didn't allow the user to plug in a plant model in order to perform closed–loop validation of the real-time system

Further development of the JETRT framework 2005–2007

- In 2005–2007 a first attempt to automatically generate user application starting from Simulink was carried out in collaboration with ENEA
- This activity was abandoned due to licensing problems at JET
- The development of the JETRT framework stopped since we moved to MARTe, which provides a real separation between the algorithmic part of a real-time application from the plant-interface software

Development phase

- The design and validation phases were still carried out in Matlab/Simulink environment
- The control algorithm was still coded "by-hands" as a Generic Application Module (GAM) in MARTe [1]
- Validation of the real-time code were performed by closed–loop simulations *GAMifying* the CREATE plasma model [2]

A. C. Neto et al.,
MARTe: a Multi-Platform Real-Time Framework,
IEEE Transactions on Nuclear Science, vol. 57(2), Apr. 2010

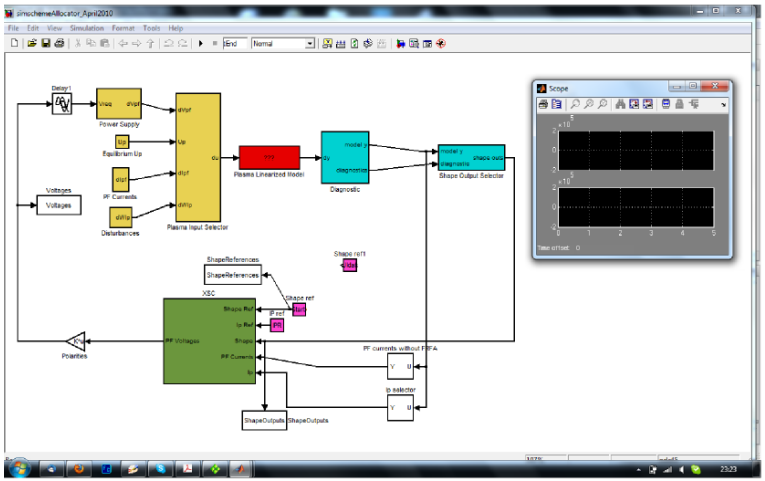
T. Bellizio et al.,
A MARTe based simulator for the JET vertical stabilization system,
submitted to 26^{th} *Symposium on Fusion Technology (SOFT'10)*, 2010

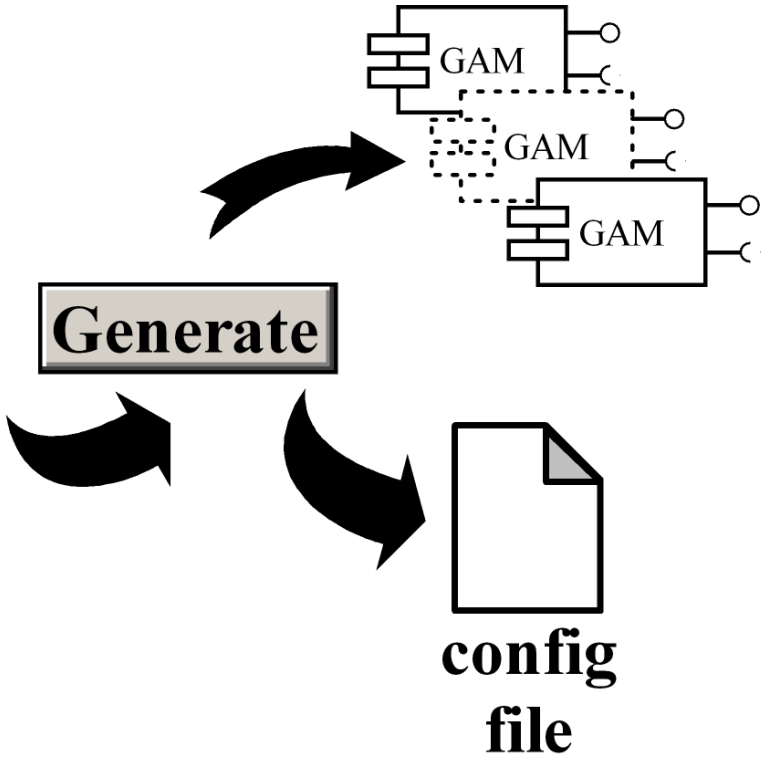
Something is still pending!

- Within MARTe we still need automatic code generation tools!
- We would like to exploits the Mathworks Real-Time Workshop to automatically generate GAMs starting from Simulink schemes

We would like to make it easy!

We would like to just press a button and generate the code for real-time





Automatic code generation

Automatic code generation is used for rapid prototyping in different fields:

- Automotive industry
- Aerospace industry
- Electrical drives
- ...

Many commercial tools are available:

- **Mathworks Real-Time Workshop**
- dSpace
- Labview Simulation Interface Toolkit (which partially exploits Mathworks Real-Time Workshop)
- ...

but there are also open source tools:

- Scilab/Scicos (compatible RTAI)
- Ptolemy/Kepler
- ...

Target Hardware

- Generally these tools generate either a *general purpose* code or they are tailored for a specific HW platform
- Thanks to MARTe architecture GAMs code is not linked to any specific HW platform
- Within MARTe the automatic code generation can be performed regardless of the HW platform

From Simulink to GAM

Matlab/Simulink environment is a standard *de facto* for the design of control algorithms

Possible activity

Automatic GAM generation starting from Simulink can be done:

- Exploiting the Real-Time Workshop Target Language Compiler (TLC) to generate a new *target* for the MARTe GAMs, which is HW independent
- Generating a *general purpose* C/C++ code with the Real-Time Workshop and then wrap it into a GAM (a similar solution is currently adopted by RFX)