Brief history of the MARTe framework

CODAC MARTe Meeting
Barcelona, September 30, 2013

G. De Tommasi[1]
[1]CREATE, Università di Napoli Federico II

Outline

Where we started from - JETRT

JETRT

From JETRT to MARTe

Conclusions

PPCC Control systems in the mid 90s

PPCC systems for plasma magnetic control

The two main systems run at JET by the Plasma Position and Current Control Group were (and still are!):

Shape Controller (SC) C code deployed on a VxWorks/VME/Motorola68k platform

the Vertical Stabilization System (VS) C code deployed on 4 Texas Instruments DSPs

- The code was *tailored* for the specific platform
- Lack of modularity
- Different software solutions to interface with the JET software infrastructure (pre-pulse system configuration, post-pulse data collection,. . .)

M. Lennholm et al.,
Plasma control at JET,
Fus. Eng. Design, vol. 48(1–2), Aug. 2000

A new framework for RT applications

- In 2001/2002 the revamping of the SC was planned in order to add the eXtreme Shape Controller algorithm (XSC)
- Within the PPCC group, it was decided to move to a common framework for the development of real-time applications

A new framework for RT applications

- In 2001/2002 the revamping of the SC was planned in order to add the eXtreme Shape Controller algorithm (XSC)
- Within the PPCC group, it was decided to move to a common framework for the development of real-time applications

Aims (User Requirements)

- Standardize the development of real-time applications
- Increase the code reusability
- **Separate (as much as possible) the user application from the software required to interface with the plant infrastructure**
- **Reduce the time needed for commissioning**

A new framework for RT applications

- In 2001/2002 the revamping of the SC was planned in order to add the eXtreme Shape Controller algorithm (XSC)
- Within the PPCC group, it was decided to move to a common framework for the development of real-time applications

Aims (User Requirements)

- Standardize the development of real-time applications
- Increase the code reusability
- **Separate (as much as possible) the user application from the software required to interface with the plant infrastructure**
- **Reduce the time needed for commissioning**

High Level System Requirements

The new framework would have been:

- **portable (multi-OS and multi-platform)**
- **modular – the user application would have been easily plugged into an executor of real-time application**
- written in C++ (object oriented approach)

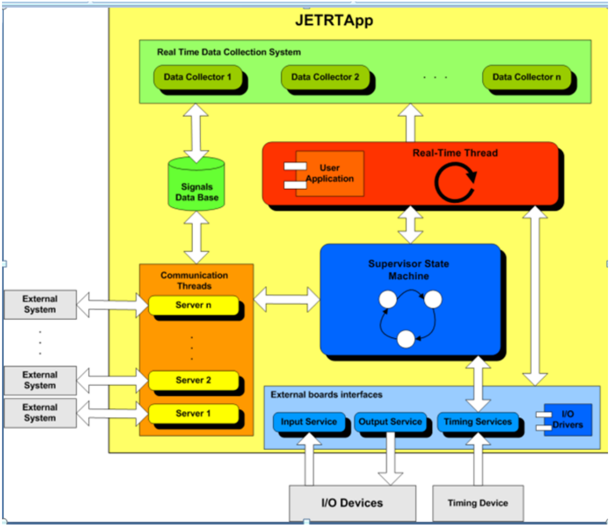
Separation between application and infrastructure software

Why we want to separate application from infrastructure software?

- Scientists (process experts) can abstract from the plant interfaces
- Increase code reusability
- Achieve standardization

JETRT

- The JETRT framework was developed in 2002/2003 to deploy the XSC
- JETRT is based on the cross-platform BaseLib library (developed within the PPCC group)



Services and servers

1. Identification of the services

Services and servers

1. Identification of the services
2. Definition of the *servers* interface

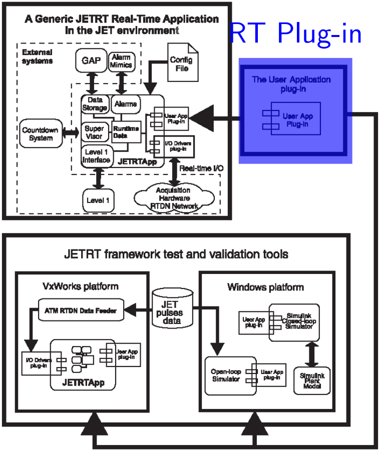
Services and servers

1. Identification of the services
2. Definition of the *servers* interface
3. Implementation (technological solutions)

Real-time application plug-in

The **Real-time Application Plug-in** that can be used to:

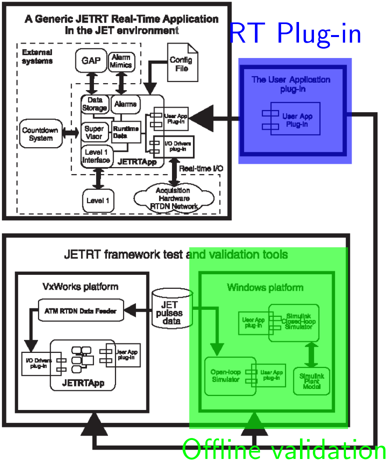
- ▶
- ▶
- ▶



Real-time application plug-in

The **Real-time Application Plug-in** that can be used to:

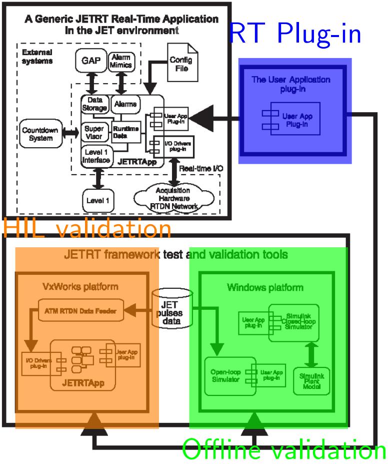
- perform offline validation against a plat model
-
-



Real-time application plug-in

The **Real-time Application Plug-in** that can be used to:

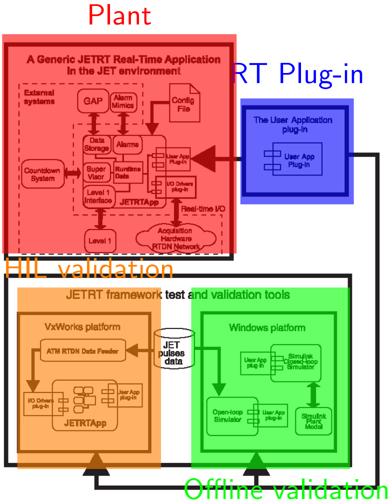
- perform offline validation against a plat model
- perform real-time validation with hardware-in-the-loop
-



Real-time application plug-in

The **Real-time Application Plug-in** that can be used to:

- perform offline validation against a plat model
- perform real-time validation with hardware-in-the-loop
- run the real-time system on the plant



XSC

- The *new* SC (including the XSC) was deployed on a 400 MHz G4 PowerPC running VxWorks
- 2 ms control loop (but it can easily run at 1 ms)

Commissioning of the JETRT framework and of the XSC

- Thanks to portability, an exhaustive debug of both the JETRT framework and the XSC was performed
  - **offline**, on a Windows–based platform

XSC

- The *new* SC (including the XSC) was deployed on a 400 MHz G4 PowerPC running VxWorks
- 2 ms control loop (but it can easily run at 1 ms)

Commissioning of the JETRT framework and of the XSC

- Thanks to portability, an exhaustive debug of both the JETRT framework and the XSC was performed
  - **offline**, on a Windows–based platform
  - **in lab**, with a mockup of the JET timing system and of the I/O

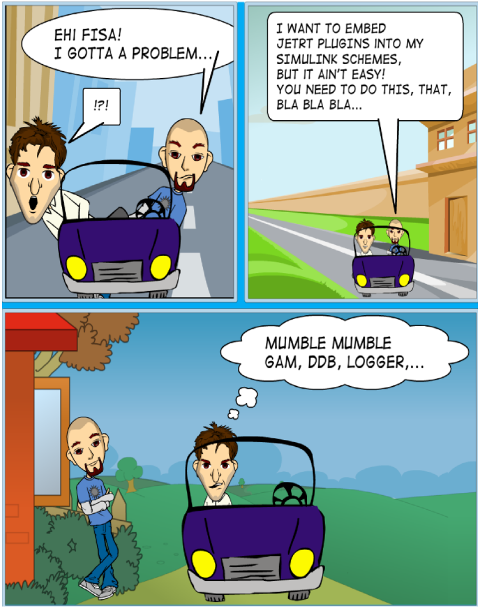
XSC

- The *new* SC (including the XSC) was deployed on a 400 MHz G4 PowerPC running VxWorks
- 2 ms control loop (but it can easily run at 1 ms)

Commissioning of the JETRT framework and of the XSC

- Thanks to portability, an exhaustive debug of both the JETRT framework and the XSC was performed
  - **offline**, on a Windows–based platform
  - **in lab**, with a mockup of the JET timing system and of the I/O
- **Only 3 days of testing on the plant were needed for the commissioning of the new system**

MARTe - The origins (Fall Winter 2004/05)


EH! FISA!
I GOTTA A PROBLEM...
!?!
I WANT TO EMBED JETRT PLUGINS INTO MY SIMULINK SCHEMES, BUT IT AIN'T EASY! YOU NEED TO DO THIS, THAT, BLA BLA BLA...
MUMBLE MUMBLE
GAM, DDB, LOGGER,...

JETRT limitations

- JETRT didn't provide a real separation between the user application from the plant-interface software!

JETRT limitations

- JETRT didn't provide a real separation between the user application from the plant-interface software!
- In 2011, about 1 ppm was needed to include the Current Limit Avoidance system in SC!

JETRT limitations

- JETRT didn't provide a real separation between the user application from the plant-interface software!
- In 2011, about 1 ppm was needed to include the Current Limit Avoidance system in SC!

From JETRT to MARTe

- *More* modularity → Generic Application Modules (GAMs)
- *Real* separation → Dynamic Data Buffer (DDB)

JETRT limitations

- JETRT didn't provide a real separation between the user application from the plant-interface software!
- In 2011, about 1 ppm was needed to include the Current Limit Avoidance system in SC!

From JETRT to MARTe

- *More* modularity → Generic Application Modules (GAMs)
- *Real* separation → Dynamic Data Buffer (DDB)

. . .what happened after is a well-known story

What I like (as a user). . .

- I need to know very few things about the framework to write my GAMs
- I can make the functional tests in my office without any specific hardware
- I'm sure that the code I'm writing will be the one deployed on the plant
- I can easily build a web-based HMI that will not affect the system performance
- If someone develops something interesting, I can easily use it in my project

. . .and what I don't

- Writing the *config* files by hand!

. . .and what I don’t

- Writing the *config* files by hand!
  - **Lack of tools that facilitate the system setup**

. . .and what I don’t

- Writing the *config* files by hand!
  - **Lack of tools that facilitate the system setup**
- Although some of effort has been done so far. . .
  - **. . .documentation is one of the weakest point for MARTe**

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

- They will contribute coming on-site

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

- They will contribute coming on-site
- They will contribute developing new features at their home lab

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

- They will contribute coming on-site
- They will contribute developing new features at their home lab
- **Definition of the APIs for all the services (I/O drivers, data collection, communication services, etc.)**

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

- They will contribute coming on-site
- They will contribute developing new features at their home lab

- **Definition of the APIs for all the services (I/O drivers, data collection, communication services, etc.)**
- **Separation between the user application and the infrastructure**

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

- They will contribute coming on-site
- They will contribute developing new features at their home lab

- **Definition of the APIs for all the services (I/O drivers, data collection, communication services, etc.)**
- **Separation between the user application and the infrastructure**
- **Possibility of testing the real-time software with a simple PC (no need of special DAQ hardware)**

What (I think) is important for the ITER framework

Long term scenario

ITER is an international project and people from all over the world will contribute to the development of real-time systems (control systems and diagnostics)

- They will contribute coming on-site
- They will contribute developing new features at their home lab

- **Definition of the APIs for all the services (I/O drivers, data collection, communication services, etc.)**
- **Separation between the user application and the infrastructure**
- **Possibility of testing the real-time software with a simple PC (no need of special DAQ hardware)**
- **Automatic code generation(?)**