R-ROSACE: adding redundancy to the ROSACE case study

Henrick Deschamps*, †, Gerlando Cappello*, Janette Cardoso†, and Pierre Siron†

*Airbus, Simulation department (EYYS)
†ISAE Supaéro, *Département d'Ingénierie des Systèmes Complexes* (DISC)

March 3, 2017

Abstract

In the aeornautical domain, simulation is becoming more and more important in the development and validation of systems, as well as for its usage in pilote training.

Generally speaking, a simulation of a Cyber-Physical System is an aggregation of models belonging to the physical world (for instance, the flight dynamic), and models belonging to the discrete world (for instance, controllers). Those models being interconected.

Irrespective of the quality of the models simulated, their executions and interactions must reflect the reality sufficiently well to be considered representative.

The goal of this paper is to introduce an extension to the ROSACE case study by taking into account controllers redundancy. We named this new case study R-ROSACE (R for redundant). R-ROSACE will allow a deeper study on communication and execution of simulated embedded systems, as well as their interactions with the simulated physical world.

Keywords — simulation, scheduling, CPS

1 Introduction

1.1 The original ROSACE case study

ROSACE (Research Open-Source Avionics and Control Engineering) is a case study covering different steps from the conception to the implementation of a baseline flight controller. Originally, the ROSACE case study started with the flight controller developed in Matlab/SIMULINK, ending with a multi-periodic controller executing on a multi/many-core target[1]. The case study itself is a longitudinal flight controller, designed to be used as a benchmark, and to illustrate the translating of Matlab/SIMULINK specifications to multi-threaded code executing on multi/many-core.

A major challenge on designing ROSACE controller is the need of interactions between control and software engineers. Control engineering and computer science does not consider the same problems in design, as these two disciplines are technically and culturally separated. For instance, computer science does not consider physical system requirements, such as stability, while control engineering ignores important computing limitation, such as tasks schedulability and network resources. This issue is peculiarly prevalent when designing CPS[2], endorsing our willingness to base our study upon a CPS, the ROSACE case study.

The ROSACE case study objective is to validate the real-time aspect, thus the properties analyzed are linked to the time and the errors. In the sequel, those properties are named:

P1 Settling time

P2 Overshoot

P3 Rise time

P4 Steady-state error

1.2 Original operational scenarios

An operational scenario is a set of events that includes the interaction of a system with its environment and its users. Operational scenarios are taken into account in the original case study, in the following cases:

case 1 The pilot set a new value to the inertial vertical speed (Vz). $Vz : 0m.s^{-1} \rightarrow 2.5m.s^{-1}$

case 2 The pilot set a new value to the true air speed (Va). $Va : 230m.s^{-1} \rightarrow 235m.s^{-1}$

case 3 The pilot wait for t = 50s then set a new inertial altitude (h). $h : 10000m \rightarrow 11000m$, with $Vz = -2.5m.s^{-1}$

case 4 The pilot regularly set a new inertial altitude. $h : 10000m \rightarrow 10500m \rightarrow 11000m \rightarrow 11500m \rightarrow 8000m$, with $Vz = -2.5m.s^{-1}$

All of them are while in cruise flight, at equlibrium.

Cases 1 and 2 also have the following requirements:

case 1 **P1** Settling time $Vz \leq 10s$

P2 Overshoot $Vz \leq 10\%$

P3 Rise time $Vz \leq 6s$

P4 Steady-state error $Vz \leq 5\%$

case 2 **P1** Settling time $Va \leq 20s$

P2 Overshoot $Va \leq 10\%$

P3 Rise time $Va \leq 12s$

P4 Steady-state error $Va \leq 5\%$

1.3 Context and objectives

This work is part of a CPS (Cyber-Physical System) simulation PhD thesis supported by Airbus, and the ISAE Supaéro.

The main objective of this PhD thesis is to define a formalism to express schedulings linked to a CPS simulation, following on from works from the ISAE supéaro on real-time distributed simulation[3], and from Airbus on Cyber-Physical System simulations framework analysis[4]. For this purpose, we decided to implement a case study, with both Airbus and ISAE simulation technologies, respectively DSS and CERTI HLA[5]. This case study is ROSACE, introduced in the previous section. In order to study some scheduling aspects, we decided to extend the original case study, by adding a mechanism to introduce and correct error on the controllers results, that complexify communications between models.

This paper introduces this extended case study, with its Matlab/SIMULINK modelling.

2 R-ROSACE case study

The original ROSACE case study introduced in subsection 1.1 was extended by adding redundant controllers.

We created tree versions of R-ROSACE:

— A discrete R-ROSACE, from the original case study available in the ROSACE repository[6].
— A continuous R-ROSACE, from the continuous model provided by the ROSACE team.
— An hybrid R-ROSACE, from the two precedent versions.

As wanted to have a simulation with the most relevant behavior, arbitrarily choose a component breakdown that seems to be the closest to a real Aircraft. Avionic systems, which are discrete, are represented by discrete models, and physical components, such as the engine, are represented by continuous models. The choice of discrete or continuous model in the hybrid R-ROSACE depends on these constraints, and are illustrated in Table 1.

R-ROSACE model	A/C	controllers
Discrete	Discrete	Discrete
Continuous	Continuous	Continuous
Hybrid	Continuous	Discrete

Table 1 – Mapping of model types for R-ROSACE models

Nevertheless, even if the hybrid R-ROSACE is the closest to a real CPS, we will not be able to reimplement this version with our simulation frameworks.

Thus, in our further works, we will create simulations close to the discrete R-ROSACE, but the hybrid versions running on Matlab/Simulink will be used as reference.

The discrete R-ROSACE might eventually be used in our future works, as it is close to the future implementations, but the continuous one might not. The continuous R-ROSACE has been created to generate the hybrid R-ROSACE, and we do not see reasons that might lead us to reuse this version.

2.1 From controllers to redundant FCCs

2.1.1 Controllers packaging in FCC

The Flight Control Computers (FCC) correspond to the controllers. Three controllers already exist in the original case study (inertial vertical speed controller, true air speed controller and inertial altitude controller). We regroup them in a system, and add the monitoring smartness.

When in command mode, the FCC takes filters values, references and flight mode as inputs, and calculates the delta elevator command, and delta throttle command, similarly to what did the controllers in the original case study.

When in monitor mode, the FCC takes the same inputs, plus the output of a monitored FCC in command mode. After comparing its own calculated values with the ones from the monitored FCC, it outputs a command for a relay for each value. We will discuss about this relay in the next subsection. Ultimately, there could be multiple monitors for multiple commands. The monitor will also share with each other information in order to determine the master in law.

It has to be noted that the mode does not define the implementation of an FCC, and vice-versa. Depending on its inputs, an FCC determines its mode, in runtime.

This extension is illustrated in figure 1.

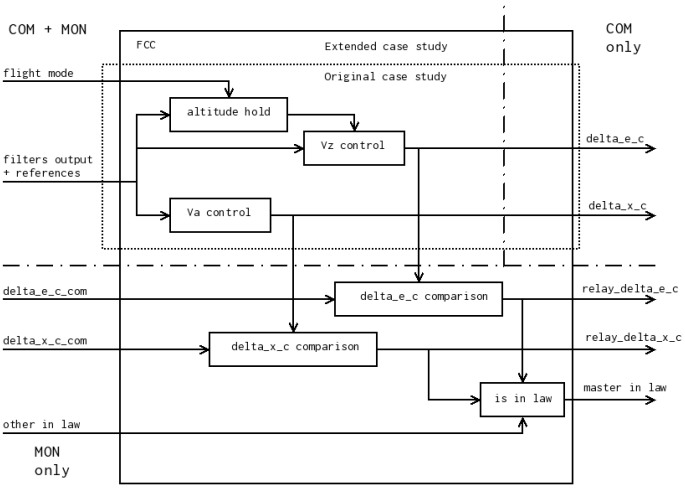

Figure 1 – Design of the FCC from controllers of original case study

2.1.2 Making FCC redundant

When using multiple FCCs, it must be determined which one will provide commands to the actuators. From the last subsection, relays from monitor were introduced. From the Airbus experience, we know that we have suficiently information to determine the command, using those relays, but we have to introduce another component that will take all the FCCs' outputs, and selecting the commands using the relays[7].

This component is a simple wiring model, containing switchs.

In R-ROSACE, we will consider a couple of FCCs, each one being composed of a command, and a monitor. We will also

add the wiring model between this couple of FCCs, and the actuators.

Figure 2 illustrates the placement of the identified components in the redundant ROSACE case study, with the wiring. Moreover, the nature of each component is highlighted (continuous or discrete), which will be discussed later.

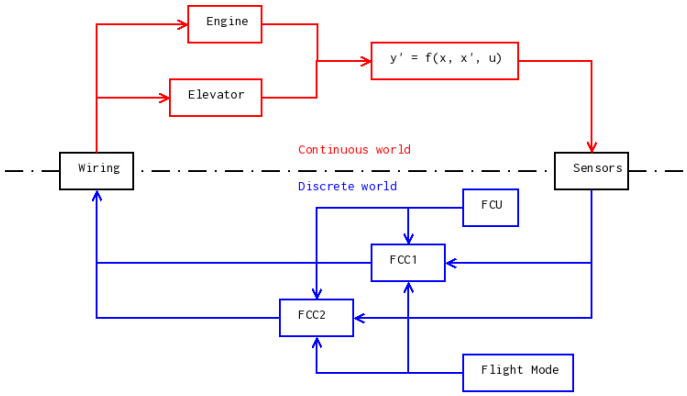

Figure 2 – The R-ROSACE case study components view

2.1.3 Injecting errors

To test the redundancy, we have to implement error injection in the models.

As for now, there is no redundant filters, plus, in R-ROSACE, we only handle a unique error. This error injection mechanism is illustrated in figure 3.

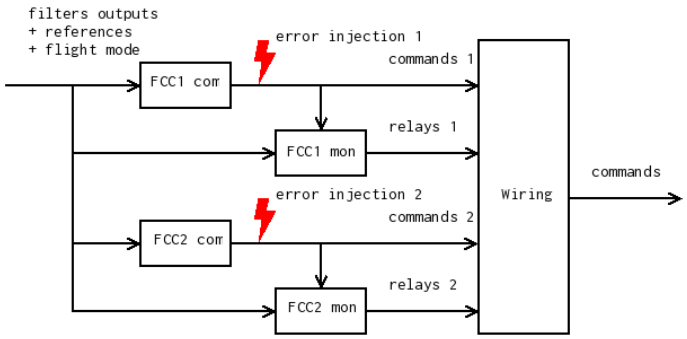

Figure 3 – Simple view of the implementation of redundancy and error injection

2.2 Discrete to cyber-physical models

Two more models are introduced in R-ROSACE. One with continuous models, from a continuous ROSACE provided by the ROSACE team, and one called “hybrid”. The hybrid model uses continuous models for physical components, and discrete model for cyber components, as illustrated in Table 1. This last model will be used as reference for further studies in CPS simulations.

The mapping of cyber and physical components is explicted in figure 2. Depending on their belonging worlds, components are implemented with continuous or discrete models in hybrid R-ROSACE.

2.3 Test cases

In order to prove that the new case study can be used as reference for further works, we need to test that requirements from the original case study are still meet with the extended version.

Due to the multiple addings to the original case study, we cannot browse all the test in this paper. We will limit the considered case to the first one of the original case study, presented in subsection 1.2.

Redundant controllers This test aims to check that the adding of redundant FCC in the discrete models does not impact the results.

Discrete to cyber-physical model This test allows verifying that the manipulation of discrete and continuous models is still valid in regard with the requirements in subsection 1.2.

The goal of this test is to verify that the CPS simulation, called “hybrid”, still meet the original ROSACE requirements.

We compare the 3 main results, true air speed, inertial vertical speed and inertial altitude, from the discrete R-ROSACE case study, with the ones from the hybrid R-ROSACE case study.

Error injection . The test consists in two injections:

— From t = 10s to t = 20s: add 0.001 rad to delta_e_c from FCC1;

— From t = 30s to t = 40s: add 0.001 rad to delta_e_c from FCC2.

When the error is detected on FCC1, FCC2 become the master, and when the error is detected in FCC2, as FCC1 is no longer in error, FCC1 become the master.

3 Results

3.1 Redundant controllers

Figure 4 shows the results of the comparing of the original ROSACE case study with the R-ROSACE one. The difference between the original ROSACE case study and the R-ROSACE one for Va, Vz and h is always equal to zero.

We can consider the R-ROSACE model as equivalent to the original ROSACE model.

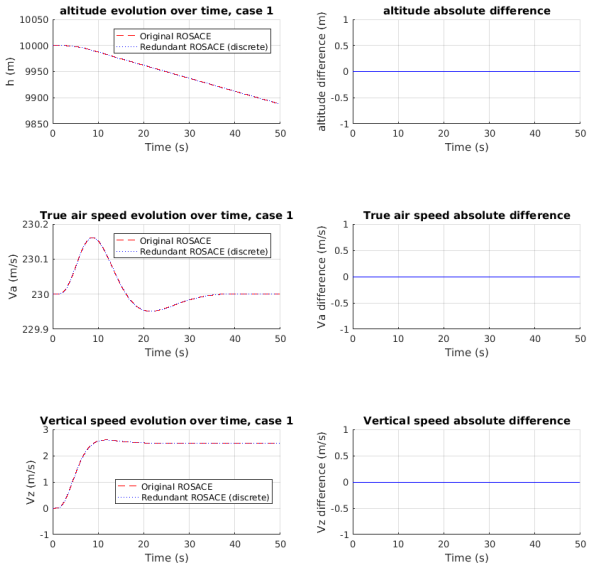

Figure 4 – Comparing of R-ROSACE results with original ROSACE

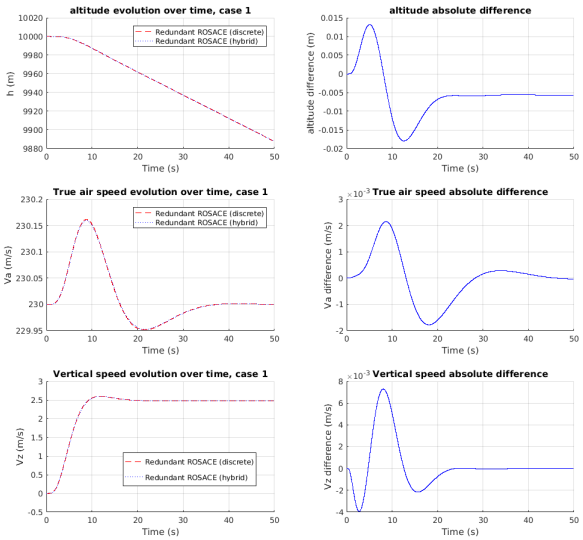

Figure 5 – From discrete to cyber-physical models

3.2 Discrete to cyber-physical models

Figure 5 shows the results of the comparing of discrete R-ROSACE with hybrid R-ROSACE.

We can notice in the left side of this figure that these two models does not have the same results. However, the differences are negligible, and original ROSACE case study requirements are still meet (subsection 1.2). These differences are due to the use of solver by Matlab when simulating the components with continuous models.

Nevertheless, we can consider the hybrid models as sufficiently correct, and we can use it as reference for the redundant ROSACE case study.

3.3 Error injection

As we want to use the hybrid model as reference, we will only show results with this model. However, introduction of redundance in the other models (discrete and continuous) R-ROSACE does not change the output of the system neither.

Figure 6 shows the error injection on delta_e_c from FCC1 and FCC2, their detection by the FCCs MON (traducted by toggling relays), and the swapping of master in law.

The FCC1 relay is correctly toogle on when the error start, and off when it ends. Ditto for FCC2.

When an error is injected on FCC1 commands, FCC2 become the new master in law. When the error stops on FCC1 commands, FCC2 still the master in law, until an error occurs on FCC2 commands. Then FCC1 is master in law again, as expected.

Figure 7 shows the impact on true air speed, inertial vertical speed and inertial altitude values. There are no differences between the hybrid R-rosace] model with errors injection, and the one without errors injection. We can conclude that the redundancy mechanism works correctly.

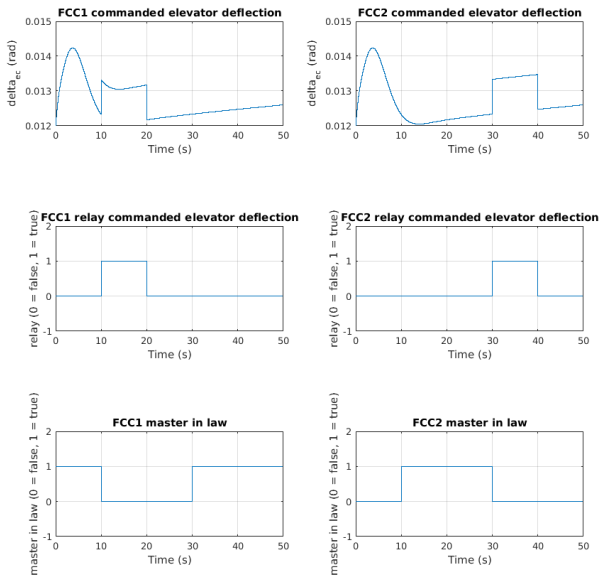

Figure 6 – Error injection on delta_e_c, and its detection

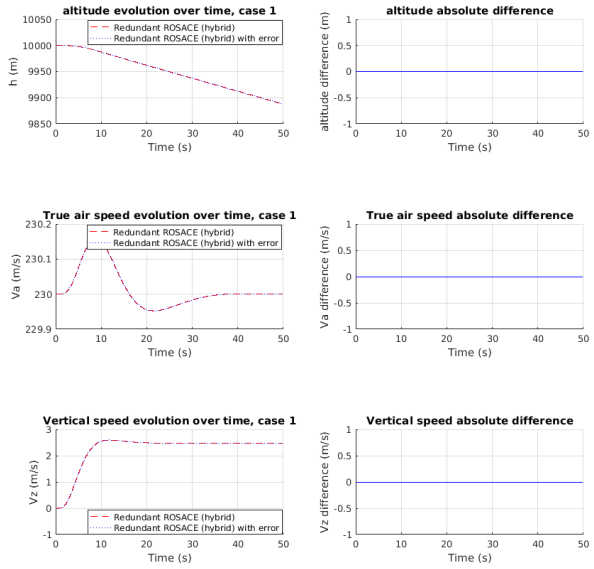

Figure 7 – Impact of the error injection on h, Vz and Va

4 Conclusion

In this paper, we introduced an extension of the ROSACE case study considering controlers redundancy and Cyber-Physical System simulation.

We also shown that those extensions does not impact significantly the simulation results, that still meet the original case study requirement.

For the next phases of our works, we will use the redundant ROSACE. We will implement it with different distributed simulation frameworks, using the hybrid models loop as reference for our study of Cyber-Physical System simulations with HLA CERTI and DSS.

5 Acknowledgement

The work described in this paper is supported through an Industrial Agreement for Research Training – CIFRE – financed by National Association for Research in Technology (ANRT). This work is also financed and supervised by Airbus, and supervised by the ISAE Supaéro.

We would like to thank C. Pagetti, D. Saussié, R. Gratia and E. Noulard for their publication of ROSACE under open source license, allowing us to use this case study as base for our own.

References

[1] C. Pagetti, D. Saussié, R. Gratia, E. Noulard, and P. Siron, "The ROSACE case study: from Simulink specification to multi/many-core execution," in *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2014, pp. 309–318. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6926012

[2] D. Henriksson and H. Elmqvist, "Cyber-Physical Systems Modeling and Simulation with Modelica," 2011. [Online]. Available: https://modelica.org/events/modelica2011/Proceedings/pages/papers/20_1_ID_121_a_fv.pdf

[3] J.-B. Chaudron, *Architecture de simulation distribuée temps-réel*. Toulouse, ISAE, Jan. 2012. [Online]. Available: http://www.theses.fr/2012ESAE0002

[4] J. Regueiro, "Scheduling of the simulation of a Cyber-Physical System," Master's thesis, ISAE Supaéro, Sep. 2013, master thesis.

[5] J. Cardoso, J.-C. Chaudemar, A. Hamez, J. Hugues, and P. Siron, "PRISE: une plate-forme de simulation distribuée pour l'ingénierie des systèmes embarqués," *Génie Logiciel*, no. 108, pp. 29–34, 2014. [Online]. Available: http://oatao.univ-toulouse.fr/11468/

[6] Rosace simulink repository. [Online]. Available: https://svn.onera.fr/schedmcore/branches/ROSACE_CaseStudy/simulink/

[7] R. Bernard, J.-J. Aubert, P. Bieber, C. Merlini, and S. Metge, "Experiments in model based safety analysis: Flight controls," *IFAC Proceedings Volumes*, vol. 40, no. 6, pp. 43–48, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015310934