INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D6.2.1 – Requirements for multi-core compilation

WP 6.2: Efficient operation and simulation on multi-core platforms

Work Package 6: Modelling and simulation services

MODRIO (11004)

Version 0.2

Date 01/07/2013

Authors

Quang Dinh	Dassault-Aviation
Emmanuel Ledinot	Dassault-Aviation
Eric Thomas	Dassault-Aviation

Summary

Executive summary

This document provides some requirements for multi-core compilation and simulation of Modelica models, derived from the needs of aircraft development.

The objective is to speed-up the simulation of large models in two complementary ways: modular distribution (coarse grain concurrency compatible with black box import of models) and transparent parallelization (fine grain concurrency requiring white box models).

The requirements apply to the different stages of the development process: model edition, compile-time and analysis, run-time monitoring, debugging, and profiling.

The target hardware architectures are multi-core PCs under Windows and clusters under Linux, whose nodes may include GPU co-processors.

The results of this Work-Package will be tested in the demonstrators (WP8.4).

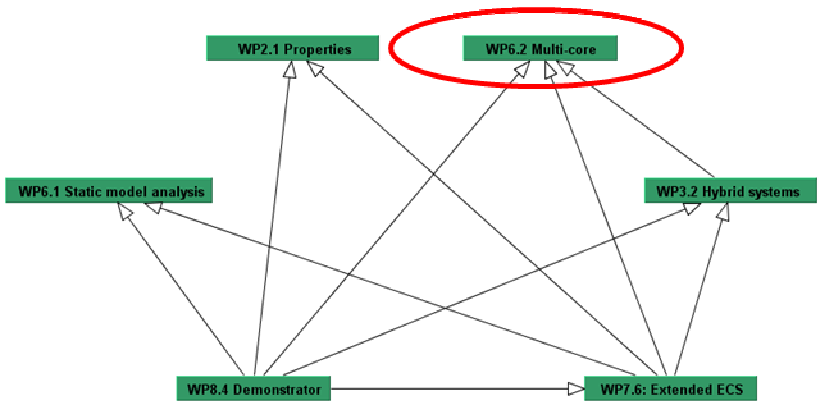
1. SUMMARY

This document details the requirements for efficient simulation and analysis of large Modelica models on multi-core platforms and in particular the needs motivated by design or on-line monitoring of aircraft vehicle management systems.

For Dassault-Aviation, the main challenge of Modrio is to be able to make fast, accurate and selective diagnostics from data measurement made on the aircraft. It is also to improve the current status concerning simulation of large models during the design phase.

The purpose of this work-package is to focus on features required to fulfill the overall objectives: enable sufficiently fast systems simulations to make system design and diagnosis from comparison between aircraft measurements and models simulations.

Dependencies with other Work-Package:



This work package doesn't depend on other Work-Packages, but should give improvements that will be used within WP 3.2, 7.6 and 8.4.

1.1 Acronyms

- API : Application Programming Interface
- BIZJET : Business aircraft
- ECS : Environmental Control System
- FMI : Functional Mock-up Interface
- HIL : Hardware In the Loop
- HPC : High Performance Computing
- MIMD: Multiple Instruction Multiple Data
- QoS: Quality of Service
- SIL : Software In the Loop
- SIMD: Single Instruction Multiple Data

- SMP: Symetric Multi-Processing
- UAV : Unmanned Aerial Vehicle

1.2 Definitions

- "Black box" models are encrypted or encapsulated binary models that can't be analyzed for generating concurrent executable object codes. They are necessarily executed as sequential units, possibly communicating with other distributed sequential units.

- "White box" models are such that their Modelica script is accessible and can be analyzed to generate parallelized executable files.

- "Grey box" models are black box models that export parameters (read/write access) and/or observers (inner variables or properties) for monitoring and value injection.

2. USAGE OF SIMULATIONS AND ASSOCIATED REQUIREMENTS

2.1 General context for design assessments and diagnosis

The need to get fast simulations is a long term requirement from designers. But, even if computer velocity grows up steadily, models that designers would like to simulate tend to become more and more complex as well, so that the gap between needs and capabilities remains large.

The objective of being able to make diagnostic, or prognostic, from comparison between aircraft measurements and model simulations in a limited time is a challenge which will be solved only with new efficient solutions.

The following sections explain the context of use of simulations and associated requirement. All the requirements will be sum-up in section 2.2.

2.1.1 Aircraft design, architecture and process

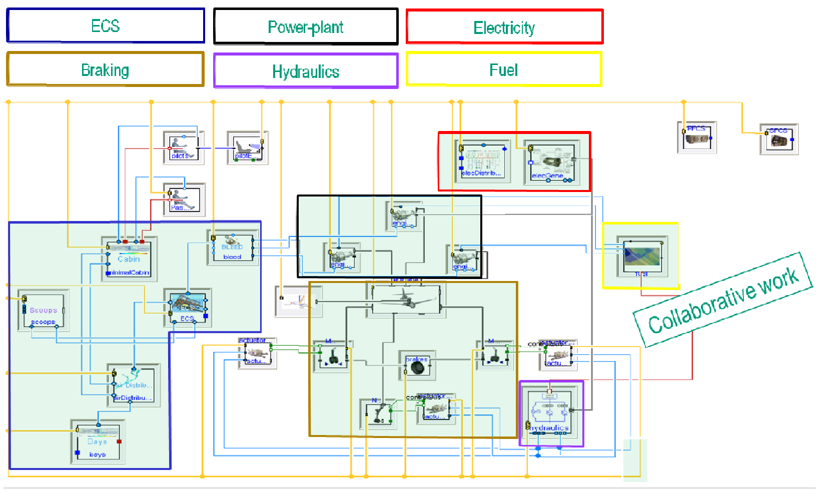
Dassault Aviation designs business and military aircrafts like those represented in the following pictures:

BIZJET FAMILY (Falcon F7X ...)

UAV (nEUROn ...)

MILITARY AIRCRAFT (Rafale ...)

For Energy Aircraft Vehicle Systems, like those represented below for conventional aircraft architectures, Dassault-Aviation works mainly as a sub-systems or equipment specifier and integrator.



A conventional aircraft is a network of more than one thousand equipments the development of which involves many partners. The design is then a collaborative work, from the conceptual design phase up to detailed design, integration and flight test.

The analysis of these energy management systems during design phase (sizing, verification, optimization, robustness analysis) and aircraft operation (on-line or off-line prognosis and diagnosis) requires fast simulation of accurate models, i.e. as fast as real-time, or even faster.

2.1.2 Target of Aircraft diagnosis for business aircraft

2.1.2.1 Experience in Aircraft diagnosis and prognosis for business aircraft

System@tic MODIPRO (2010 – 2012): Model-based and learning-based (flight data recorders) diagnosis and prognosis for business aircraft.

MODIPRO is exclusively focused on discrete event analysis and learning based techniques. There is no use of hybrid models to detect dynamic effects (failures or trends) for diagnosis and prognosis. In particular there is no work on data assimilation.

2.1.2.2 Objective

In Modrio, the purpose is to improve operational dispatch and introduce condition-based maintenance through enhanced system diagnosis and prognosis based on intensive flight data analysis.

State-of-the-art (Business Jets): on-board rule-based diagnosis, no prognosis. Data recorders are currently limited to crash recorders and monitoring of limited sets for parameters for on-demand system troubleshooting.

Innovation: upgrade data recorders to monitor all available parameters (thousands) on all flights. Offer new services to operators for accelerated troubleshooting and trend monitoring. Progressive introduction of condition based maintenance besides regulatory planned maintenance.

2.2 Constraints associated to simulation usage

During the design process (see illustration fig. 1), many types of behavior assessments are needed requiring different types of tools: that which describe physical and control sub-systems (functional/logical) as 0-1D representations defined for continuous or discrete time; that which are efficient for calculations based on 3D or 4D representations.

All these tools contribute to the verification of the architectures. But up to now, most of these analyses are still made only on parts of the architectures, by quite independent and heterogeneous tools.

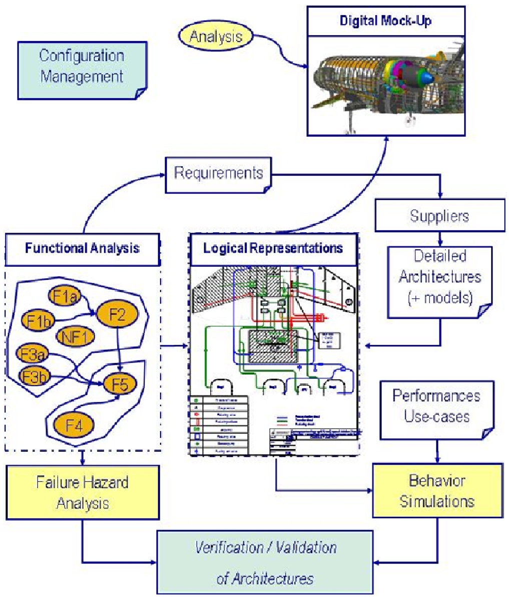


Figure 1 : Design process

Now, modeling and simulation tools have better capabilities to deal with designers’ needs, they are more flexible and allow a large set of connections to other tools, even if tool coupling is still often inefficient.

Therefore the purposes of model distributed co-simulation is to abstract from existing tools and collect designers’ needs to make heterogeneous integration of models run fast and afford efficient work.

The following pictures illustrate the context for the designers.

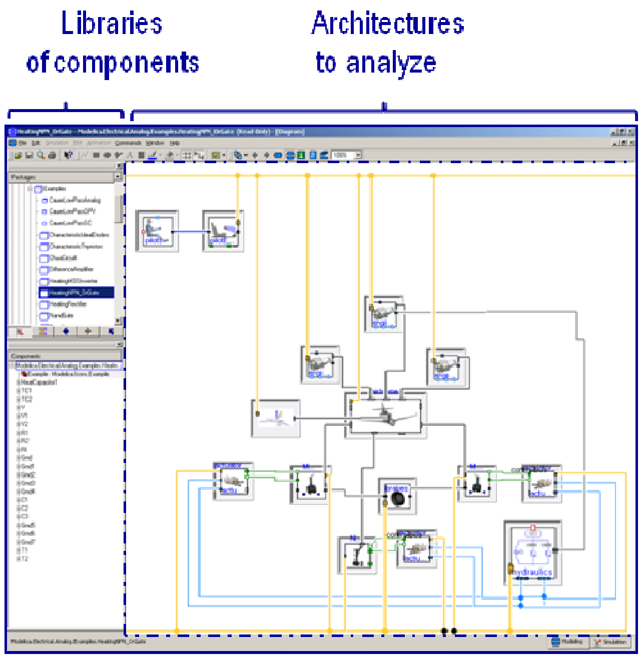


Figure 2 : simulation tool for simulating several sub-systems (e.g. Power-plant, Hydraulics and Landing gears)

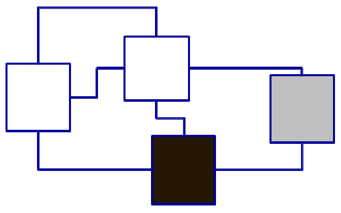


Figure 3 : interconnection of components seen by the tool as “white”, “grey” or “black” boxes

Existing tools have the capability to build-up architectures. Based on theses architectures, designers would like to be able to make different types of analysis and in particular to assess the resulting behaviors in different use-cases:

- By association of behaviors to parts of the architecture. The solution should then take into account that:
  - ♦ Depending on whether the models are developed internally or provided by some partner, they could be seen by the concurrent simulation tool as “white”, “grey” or “black” boxes,
  - ♦ Each model will have its own dynamic (continuous time resolution of the solver, continuous time critical constants, discrete time clocks), which could be much faster or slower than that of the other models. These timing characteristics could influence or react to the surrounding sub-systems or components.
- Depending on the appropriate level of modeling in the different circumstances, one may need at the same time:
  - ♦ Rough and quick calculations based on high level behavioral abstractions with few parameters, to make thousands of calculations to find out the right architectures.
  - ♦ Several accurate calculations to validate the optimized architecture, and check its robustness.
  - ♦ Real-time calculation, or even faster than real-time calculations, when simulation is used to compare (on-line and/or off-line) flight operation records with model predictions, or when simulation is used on validation test beds (SIL, HIL …)
- Depending on the available hardware resources:
  - ♦ multi-core processors on personal PC workstations
  - ♦ clusters of multi-core CPUs of computing centers

In all cases fast calculation is at stake. The following section details some user requirements regarding the development, deployment and tuning of concurrent Modelica simulations.

2.3 Requirements

They are grouped by development stages, and then by use cases. As far as possible they are need-oriented rather than solution-oriented.

2.3.1 Deployment stage

A global Modelica model to be executed concurrently has been edited. User-supplied information has to be added to define the concurrent units and their possible associated mapping constraints to logical hardware resources.

[RD01] Two modes of concurrency should be available: modular distribution and transparent parallelization (mandatory)

- *[RD01a]* Modular distribution is dedicated to coarse grain distributed memory concurrency. It shall support distributed execution of FMI black box models. It shall support synchronized multi-clocked discrete time and global continuous time over the processing network.
- *[RD01b]* Transparent parallelization is dedicated to shared memory concurrency (SMP in particular). It shall support automatic SIMD or MIMD parallelization of computation intensive white box Modelica models, previously tagged by the user as candidates for transparent parallelization.
- *[RD01c]* The models subject to fast concurrent simulation may be made up with all kinds of equations generated by Modelica tools. It should speed-up calculations either of pure continuous models or of hybrid systems (with discrete equations and/or synchronous parts).

[RD02] Concurrency directives can be defined in two ways: scripting (mandatory) and diagramming (nice to have)

A makefile approach to the concurrent simulation build is advocated to be preferred to some UML-like deployment diagramming, though diagramming is valuable as well in the long run. The concurrency directives include:

- The definition of the distributed and parallelized Modelica units,
- The allocated resources of the target multi-processor / multi-core / multi-threaded execution platform,
- The mapping constraints of the model units onto the execution units, possibly compatible with load balancing functionalities (middleware dependent),
- The mapping constraints of logical time onto processor-time, and possibly onto physical time if the distributed simulation includes interactions with physical devices (include issues of dates, events, frequencies, phases, durations),
- The numerical precision constraints of concurrent execution .wrt. the equivalent sequential execution. Preferably a QoS approach is adopted, or control on the options and parameters of the distributed solvers is provided.

[RD03] Analysis and/or synthesis tools to support the design of the concurrent simulator (mandatory)

- *[RD03a]* Models will contain different dynamics. Therefore, it should be interesting to allow designers to detect which components or sub-systems slow-down the whole simulation. Before soft and/or hard solution, smart diagnosis to localize causes of dysfunction or slow simulation should be provided to users.
- *[RD03b]* Profiling of sequential and concurrent simulation is needed to support identifying the parts of the model to be distributed or parallelized, as well as tuning the mapping and the QoS parameters.
- *[RD03c]* A data-flow and causality analysis tool would also help the user in splitting a large model in a causally consistent way. Some checks must be made and feedback given to the user on the compatibility between the acausal and physically causal aspects of the simulation on one hand, and the computational causalities induced by the distributed (and/or parallelized) execution on the other hand.

[RD04] compatibility with customer-supplier relationships

Models will be built-up as the interconnection of many components, with potentially hidden codes (FMI imports). The solutions should be compatible with these constraints.

[RD05] The concurrent deployment language (textual and graphical) should include traits to define possible monitoring and logging features to be added to the concurrent simulation (nice to have)

For off-line debugging, the event occurrences (zero-crossings) in the execution network may need to be traced, logged and then displayed on a common time line, or at least consistently with causal ordering in the computing network. Properties are a particular source of events to be optionally monitored in concurrent simulations.

[RD06] Execution platforms

The following list of execution platforms encompasses the needs for aircraft design (ground-based concurrent simulations) and model-based diagnostics (ground-based or airborne fast simulations):

- *[RD06a]* multi-core processors (mandatory):
  - ♦ 2-core processors: Intel® Core™2 Duo CPU E8400 @3.00GH; RAM 12.0 Go; System : Windows 7 – 64 bits
  - ♦ 4-core processors: Intel® Xeon E5-1620 0 @ 3.60 GHz; RAM 16.0 Go; System : Windows 7 – 64 bits
- *[RD06b]* available hardware for parallel/distributed computing (mandatory):
  - ♦ Systems using Windows HPC pack 2008 (and upper) API

- *[RD06c]* embedded maintenance computers and flight data recorders (nice to have)
    - ♦ Systems using Linux on PPC processors

[RD07] Strong scalability through time domain decomposition of DAE integration (long term perspective)

For the aircraft design use case, when hundreds of scenarios on long time integration horizons (thousands or tens of thousands of steps) have to be computed, time decomposition integration techniques on HPC environments would be nice to have. Any provision on integration schemas and concurrent simulation middleware to reach this goal is worth considering.

2.3.2 Run-time stage

[RR01] Concurrent simulation control and monitoring

Facilities to monitor distributed simulation progress, and basic simulation control commands (start, stop, suspend, resume, reset) should be provided.

[RR02] Support for massive simulation campaigns

For the aircraft design use case (parameter sweep, robustness analysis) the concurrent simulator may be supplied with a large number of simulation cases generating large output data sets. Such functionalities should be supported and scale-up (thousands of eventually independent calculations, in which systems of hundreds of equations have to be solved).

[RR03] Fault-tolerance

Hardware failures of the HPC simulation platform may occur during massive simulation campaigns. All or nothing modes should be avoided through some aspects of check-point mechanisms.

3. REFERENCES

[R01] Modelica Conference 2012: "Collaborative complex system design applied to an aircraft system"; Eric Thomas, Michel Ravachol, Jean Baptiste Quincy and Martin Malmheden

[R02] Modelica Conference 2012: "A Data-Parallel Algorithmic Modelica Extension for Efficient Execution on Multi-Core Platforms"; Mahder Gebremedhin, Afshin Hemmati Moghadam, Peter Fritzson, Kristian Stavåker

[R03] PhD Thesis 2011: "Contributions to Parallel Simulation of Equation-Based Models on Graphics Processing Units"; Kristian Stavåker

[R04] Modelica Conference 2009: "Higher-Order Non-Causal Modelling and Simulation of Structurally Dynamic Systems"; George Giorgidze, Henrik Nilsson

[R05] SMI 2008: "Multicore computing – the state of the art"; Karl-Filip Fax, Christer Bengtsson, Mats Brorsson, Hakan Grahn, Erik Hagersten, Bengt Jonsson, Christoph Kessler, Bjorn Lisper, Per Stenstrom, Bertil Svensson

4. APPENDIX

A.1. FPP extract

WP 6.2: Efficient operation and simulation on multi-core platforms (<u>LIU</u>)
Detailed activity: The objective is to speed up the operation and simulation of models, e.g. to meet real-time deadlines, by using compilation and run-time techniques for efficient execution on multi-core platforms. The main goal is to obtain significantly faster simulation and/or control, thereby also enabling applications with short real-time deadlines. New improved compilation and run-time techniques will be developed for efficient simulation of integrated synchronous-continuous multi-physics and digital control models to multi-core platforms. The compilation techniques aim at using both fine-grained task parallelism, data parallelism, and coarse-grained parallelism from model partitioning. Parallel solvers will also be investigated, including the Wave Form Relaxation method. Moreover parallel execution of groups of independent simulations will be utilized, e.g. in sensitivity analysis of model parameters. **Starting point:** Dassault Aviation background on multi-core in embedded systems and high performance scientific computing (ITEA2 ParMA and H4H). ITI has implemented different simulation runtime systems used in SimulationX and FMI components. The possibility of targeting multi-core systems will enhance the capabilities significantly. LIU has been a pioneer in methods of compiling Modelica equation-based models to parallel multi-core architectures. Early prototypes have been developed in OpenModelica. FH Bielefeld has developed Jacobian evaluation support in OpenModelica. IFPEN: Current version of xMOD that has multi-core and multi-OS simulation capabilities, but needs more rigorous approaches to ensure the numerical stability of parallel and distributed simulation runs. Supmeca : Expertise in analysis and simulation of mechatronic systems Triphase runs high-speed real-time motor drive and electric powertrain controls on x86 multi-core architectures.

Partner	**Contribution**
<u>LIU</u>	Design and implementation in OpenModelica of improved methods to compile Modelica models to efficient parallel code for multi-core platforms. An integrated approach will be used, exploiting both task parallelism, data parallelism, and coarse-grained parallelism from weakly coupled submodels.
FH Bielefeld	Implementation of parallel compilation and evaluation of analytical sparse Jacobian matrices for multi-core platforms in OpenModelica.
Dassault-Aviation	Provide requirements and assessment of WP results. The WP results will be tested in the demonstrators (WP8.4).
Triphase	Provide requirements and assessment of WP results. The WP results will be applied and tested in the demonstrator WP8.8 on the Triphase real-time development target.
ITI	Investigate requirements of compilation for multi-core platforms in equation based simulation applications.
IFPEN	State-of-the-art document and implementation of parallel and distributed simulation methods.
Supmeca	State-of-the-art document and methods for analysis and decomposition of large scale hybrid systems for distributed simulation.

Date	**PU/CO**	**Deliverable**
M9 M18 M39	PU	D6.2.1 – Requirements for multi-core compilation (<u>Dassault-Aviation</u>, Triphase)
		Document which will describe needs for multi-core compilation for business aircrafts. The objective is to speed-up the simulation of large models.
M18 M39	PU	D6.2.2 – Intermediate and final report on multi-core compilation (<u>Dassault-Aviation</u>)
		Document which will describe results of the WP for business aircrafts.
M39	CO	D6.2.3 – Feasibility study for multi-core compilation (<u>ITI</u>)
		The concepts developed in this sub package will be tested for integration into SimulationX.
M18 M39	PU	D6.2.4 – Prototype implementation of integrated multi-core compilation in OpenModelica (<u>LIU</u>, FH Bielefeld, IFPEN)
		Design and implementation in OpenModelica of an integrated multi-core compilation approach for Modelica, exploiting both task parallelism, data parallelism, and coarse-grained parallelism from weakly coupled submodels.
M18 M39	PU	D6.2.5 – Prototype implementation of parallel Jacobian evaluation with multi-core compilation (<u>FH Bielefeld</u>, LIU)
		Prototype implementation of parallel compilation and evaluation of analytical sparse Jacobian matrices for multi-core platforms in OpenModelica.
M12	PU	D6.2.6 - State of the art methods for the parallel solution of hybrid ODE systems (<u>IFPEN</u>, Supmeca)

		This report will present a state-of-the-art of parallel resolution methods of hybrid ODEs, including the Wave Form Relaxation method.
M18	PU	D6.2.7 – State-of-the-art analysis of complex systems with varying parameters (<u>Supmeca</u>, IFPEN)
		Document describing techniques for the decomposition of large scale hybrid systems in the context of parallel and distributed simulation. It aims to provide rules of decomposition that will constitute an input to the tasks related to parallel compilation, execution and sensitivity analysis.
M24	CO	D6.2.8 - Prototype implementation of parallel and distributed simulation in xMOD <u>(IFPEN)</u>
		Generic implementation of parallel and distributed simulation in xMOD. A specified meta-model will allow the simulation kernel to be configurable in order to perform the Wave Form Relaxation method or other defined methods, and exploiting OpenModelica parallel code.