Formation Flying
What's Coming Up

Research & Development directions for Formation Flying simulation and AIV

In cooperation with CNES and Estec

Fernand Quartier – Mathieu Joubert

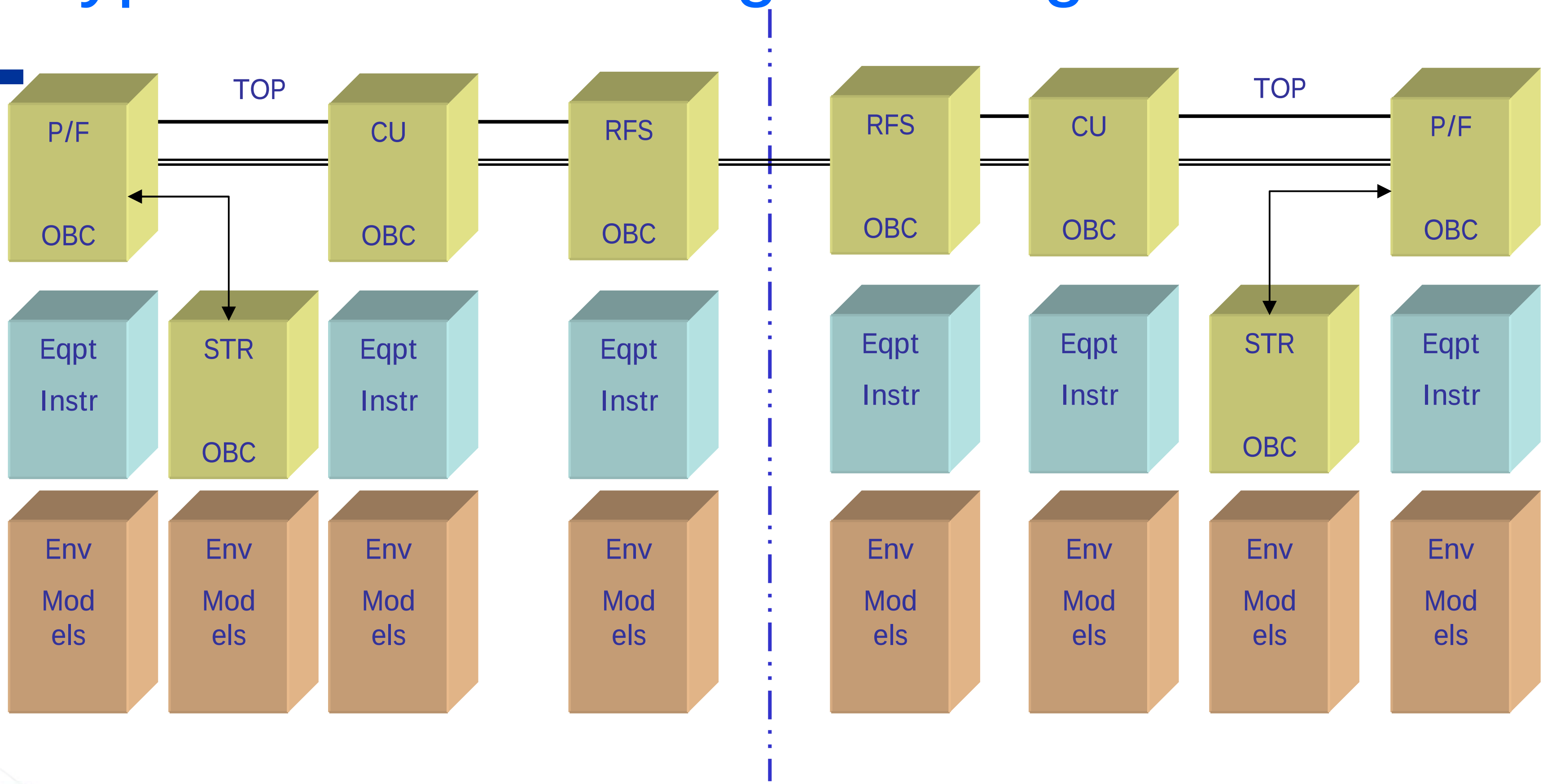
Summary

- Coming up: Formation Flying challenges
- New needs
- Separability
- Simulator distribution
- Hardware in the loop
- Various related projects
- Wish List
- Celestia

Formation Flight: Major Challenges

- Agility and accuracy of Formation Flight
  - Range of new technologies (metrology, formation flight, distributed instruments)
  - Limited know how and experience
  - ➢ More iterations and concurrent engineering
- Agility of development industrials, tools and teams
- Major tendency: study models become part of the flight software (Proba, Prisma)

Typical Formation Flight Configuration



Basiles

Ambition Basiles → Simulators to test benches

→ **Most important needs:**

- Modularity, reconfigurability and performance
- Input/output capacity
- Overall synchronous time view

Formation Flight: New Needs

- Increased processing capability
- Distribute on multi-cores, multi-processor and multi-systems
- Productivity
- Study simulators to operational simulators:
  - Evolve from synchronous to asynchronous (clock drifts)

Formation Flight: Hardware in the Loop

- It should be possible to replace almost every (sub) system model by real equipment
- Modular small reconfigurable bench hardware components preferred
  - Many teams in several places
  - Many potential configurations
    - (Large test hardware limits usability, deployment and concurrent development)
- Validation procedure reuse can generate major cost savings

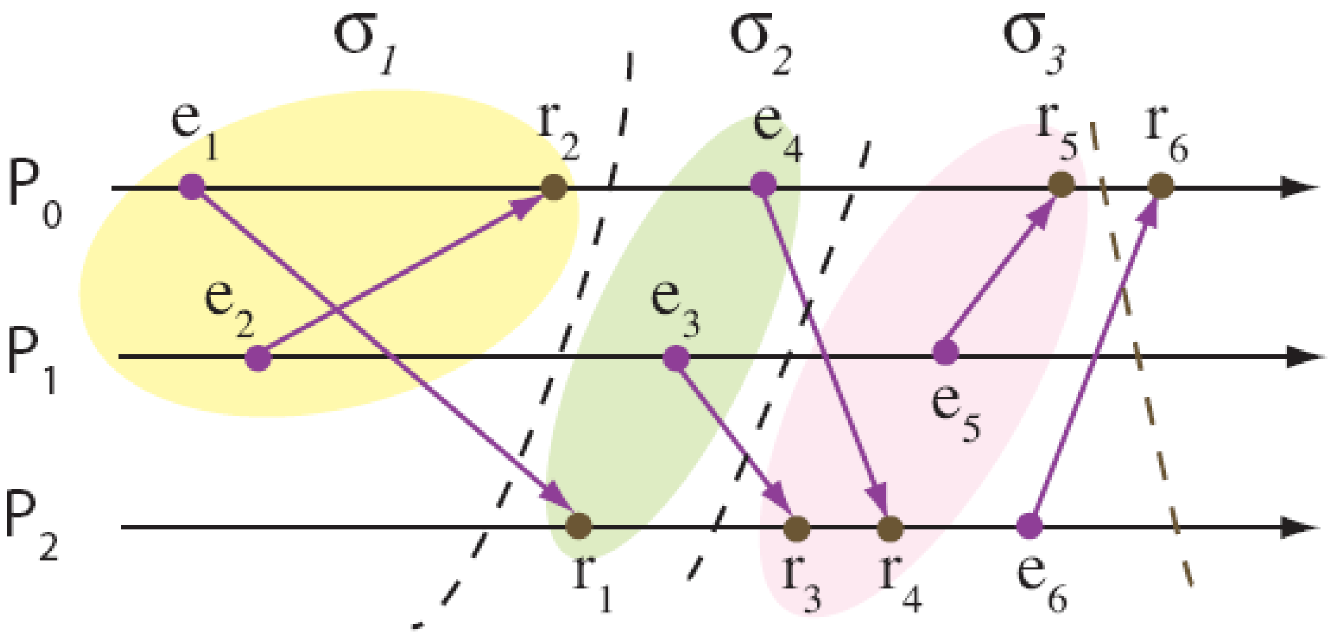
Hardware in the Loop: Study Targets

- Mainstream systems using mainstream Linux
- Real-time needs
  - Overall coherent clock system in all participating subsystems
    - Difference from one system to another better than 100 µs
  - Ability to address external I/O with a precision in the 100 µsec range (TBC)
  - Investigation shows that 10 µsec range or better might be in reach
- But
  - Linux services:
    - Unavoidable: log, windows I/F, sockets, network, Tcl/Tk commands
    - Introduce major jitter
      - 1 to 5 milliseconds: can be managed
      - > 5 milliseconds: need design changes or double buffering
      - Need be characterised
- Overall:
  - Decoupling needed between real-time and Linux side
  - Key is timing elasticity between simulators/Linux and hard real-time devices

Separability: Study Goals

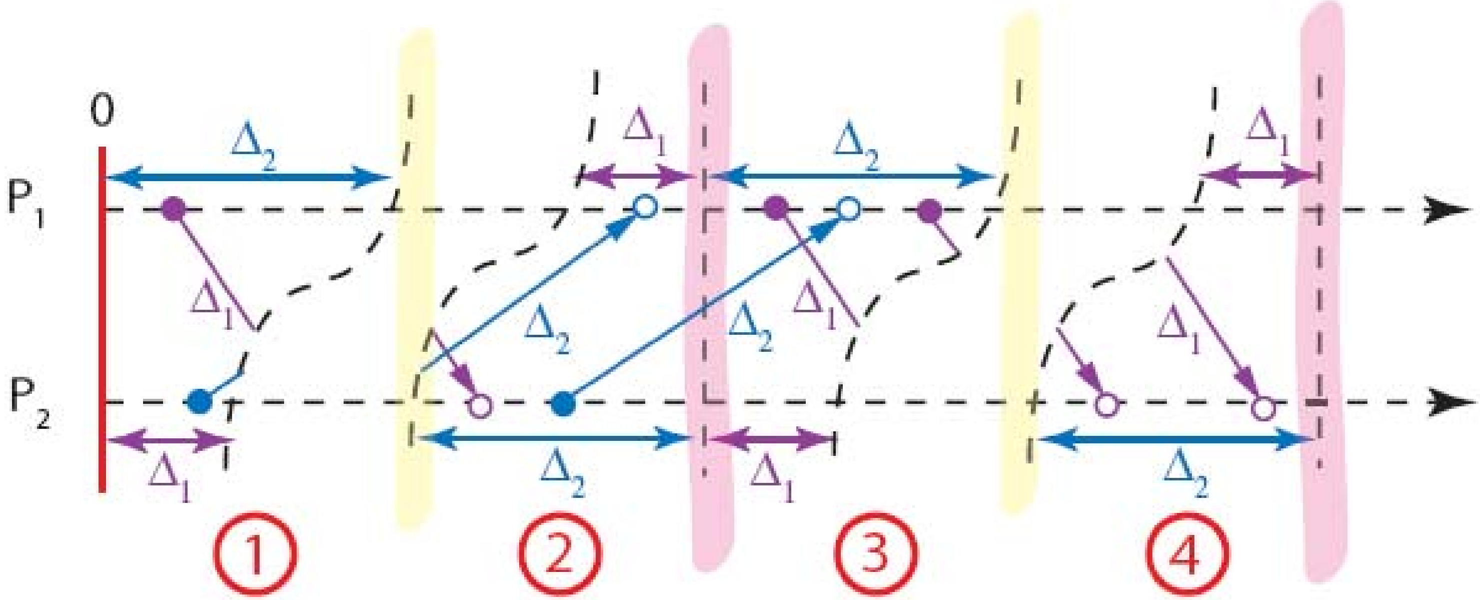
- Aims at
  - Finding methodology for defining time dependencies between subsystems
  - Defining if their simulation can be distributed
- In cooperation with CNES and IRIT University of Toulouse
- Timing variations because of moving satellites is negligible in respect with other timing jitter and clock drifts

Separability: Coupes (Slices)



A coupe or slice is a time slice where no activity in one system has a causal effect on the other

Separability: Coupe Execution



Coupes allow for:

- Communication between two coupes
- Timing freedom within the coupe

Separability: Procedure

- Characterise all communication paths:
  - Period or minimal distance between two communications
  - Delay: time between the intention to communicate and its arrival
  - Offset: initial timing offset
  - Jitter: maximal jitter (and clock drift)
- Little tool to generate for each simulator
  - Timing domain: coupe distance (initial distance, sequence of distances)
  - Frequency domain: (initial phase, frequency)

Separability: Tests and Conclusions

- Tests on
  - Simulator Pleiades computer and Doris instrument
    - Pseudo parallel simulation
    - Synchronisation frequency and interactions have been reduced with a factor of 50
  - Test cases
    - Confirm the theory (ongoing)
    - Looks that there could be some potential optimisations
      - Simulation ahead of time
- Conclusions:
  - We have a way to define the separability of models (in terms of timing constraints)
  - Bad separability seem to point to bad testability, potential race conditions and integration problems with real equipment

Simulator distribution approach

- Using HLA standard
  - Product CERTI HLA RTI, public domain from Onera, Toulouse
- Spacebel produced a small library that greatly simplifies
  - Communication of variables and events
  - Synchronisation
- External discrete event simulators integration with Basiles
  - Saber, Estec SMOS Payload simulator, Mirasim
    - Environment integration in one of two months
    - Model communication integration in days

Basiles Distribution Approach (1)

- Starting from normal mono thread simulator
  - Basic validation and qualification on such system (Many months of work)
- Basic approach based on Basiles's connection approach
  - All variables and activations (events) are connected through Tcl script commands at start up
  - Distribution consist in changing the connections
  - No model recompilation required to change distribution

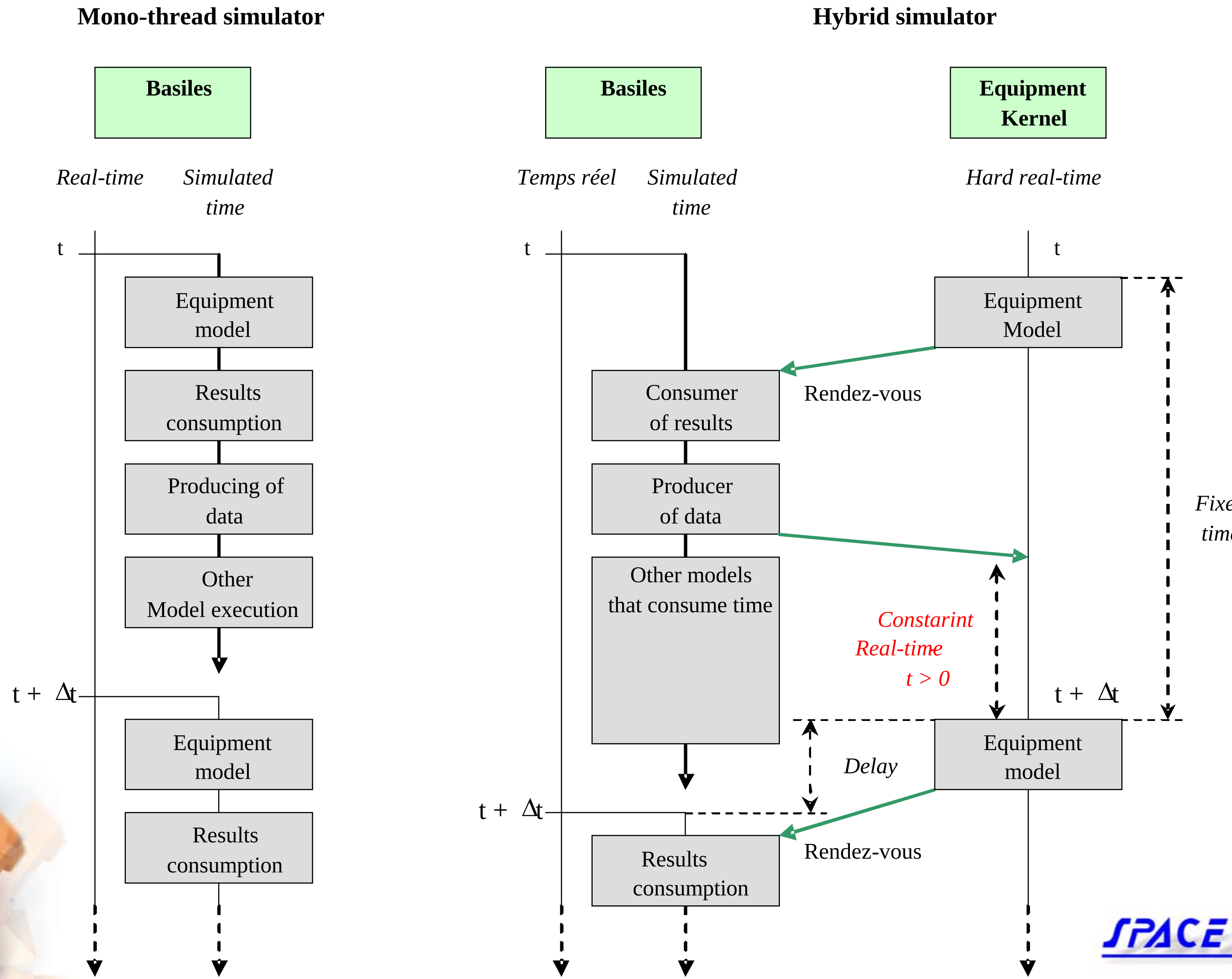
Basiles Distribution Approach (2)

- Models are assigned to adequate federates or threads
- Models can be replaced by Hardware in the Loop models
- Productivity:
  - distributing a new simulator in days
  - changing model connections < 1 hour
  - Updating/replacing models almost immediate
- Connections of variables and activations get more type parameters:
  - Direct: no delay (cyclic systems)
  - Next time: next synchronisation or look-ahead time
  - @ specified time

Basiles Distribution Federate Types

- Standard Basiles anywhere on a network (standard HLA)
  - 100 to 500 Hz
- Standard Basiles in the same machine (Light weight Certi)
  - 1.000 to 5.000 Hz
- Parallel threads in the same process space
  - 10.000 to 100.000 Hz
- Hardware in the loop pseudo model

Basiles versus Hard Real-time



Additional Basiles Distribution Activities

- Further instrumentation of Basiles kernel
  - to measure min, max, histograms of time consumption, limit reporting
  - to prepare for future soft real-time kernels
- Integrate a 1553 controller in Presto/SMOS to drive a real payload
- Prototype a run-time model dependency analyser
  - To identify model dependencies
  - To identify model parallel execution and time gain potential
  - Towards a potential automated distribution system

Various Other Related Activities (1)

- SMP2 industrialisation and DemoSim
  - Intensive and iterative cooperation
  - All players are responsible for a subsystem and generate a model to be used by another player
  - Work with Presto/Basiles, Eurosim and Simsat 4
  - Evaluate tools
  - Feedback to standardisation process
  - Proves heavy CNES involvement in SMP2
- Investigate the large scale aeronautics approach
  - Tens of computers and subsystem providers
  - Potential Arinc 653 (IMA) advantages and approaches

Various Other Related Activities (2)

- AGATA: autogeneration of Flight SW
  - Before hardware configuration is defined
  - Providing a “hardware interface” level infrastructure that communicates with the simulator models
- LEON processor emulator
  - LEON TSIM and Target Simulator characterised against hardware
  - New generation LEON Target Simulator in preparation:
    - Dynamic translation
    - Goal cycle precise for I/O emulation
    - Target > 180 MHz Leon III simulation
    - Representativity and checking variable in function of time and type of emulated code
    - Various cache modes: precise, statistical, best case, worst case
    - Detection of self-modifying code (corruption)

Wish List

- Bringing people of study simulators closer to operational simulators
- Investigate the potential of the SCICOS (SCIlab) design tools as universal tool
- Investigate improved validation and qualification systems

Wish List: Productivity Tools (1)

- When starting to build a new simulator:
  - 80 % of the models exist in some form or another
  - 95 % of parameters, data, connections, wiring (and implicit design) do exist
- Presto (Jason1 & 2, SMOS, Calypso, Corot): autogeneration
  - A couple of days to make the files that generate the simulator and its ICD

Wishlist: Productivity Tools (2)

- The day we have full reuse (100 %):
  - Development cost converges to 0 %
  - Validation and documentation converge to 100 %
  - How to add quickly a simulator design around the existing models and data
  - Exploit better modern tools (XTCE, XIF) to transport information
- We can produce the most sophisticated simulator, but are we unable to
  - Connect the various levels of documentation and the simulator configuration
  - The simulated values and the corresponding documentation
  - “Living documentation” could be autogenerated with the simulator

Graphical 3D Rendering

- Our job: space systems, not graphical 3D systems
- Celestia
  - Universe simulator and OpenGL 3D with huge celestial database
  - Public domain software - +1 million downloads, Windows-Linux-Mac
  - Enthusiastic contributions from all over the world
  - Modular – Lua scripting language
  - Evolving towards a real-time system
    - Can now be driven by simulators and operational systems
    - Separation of space mechanics from 3D Display

Celestia

- Spacebel created a small tool (CFM) to facilitate 3D image integration
- Productive environment
  - 0,5 days to 5 days to integrate new satellite
  - 1 to 4 weeks to integrate with new simulator family (Presto, Simsat) or operational system
  - 0,2 to 2 days to integrate with new simulator
  - 2 to 5 days to generate new mission movie
- Used in
  - Proteus family of satellites (Jason1 & 2, SMOS, Calypso, Corot)
  - Proba FF testbed (Simsat 4)
  - Concurrent engineering facility CNES
- Looking to start an European consortium to support Celestia and its evolutions
- Replacement of Topaz/Opale visualisation toolkit under development

Celestia Example: SMOS Deployment

The End

We still have to learn a lot ...

Thank you

Time for questions and ...

a little SMOS movie

(Produced in one hour)