David Chemouil (Onera/dtim)

SOME CHALLENGES FOR FORMAL METHODS

LESSONS LEARNT IN AND WITH THE EUROPEAN AEROSPACE INDUSTRY

IFIP WG 1.3 meeting 2010-07-05

THIS TALK

Objective

To share some lessons learnt and possible research directions for **formal approaches for systems engineering**, based on working in, and with the European aerospace industry for a few years.

Needless to say...

This is a subjective and partial view based on a personal experience.

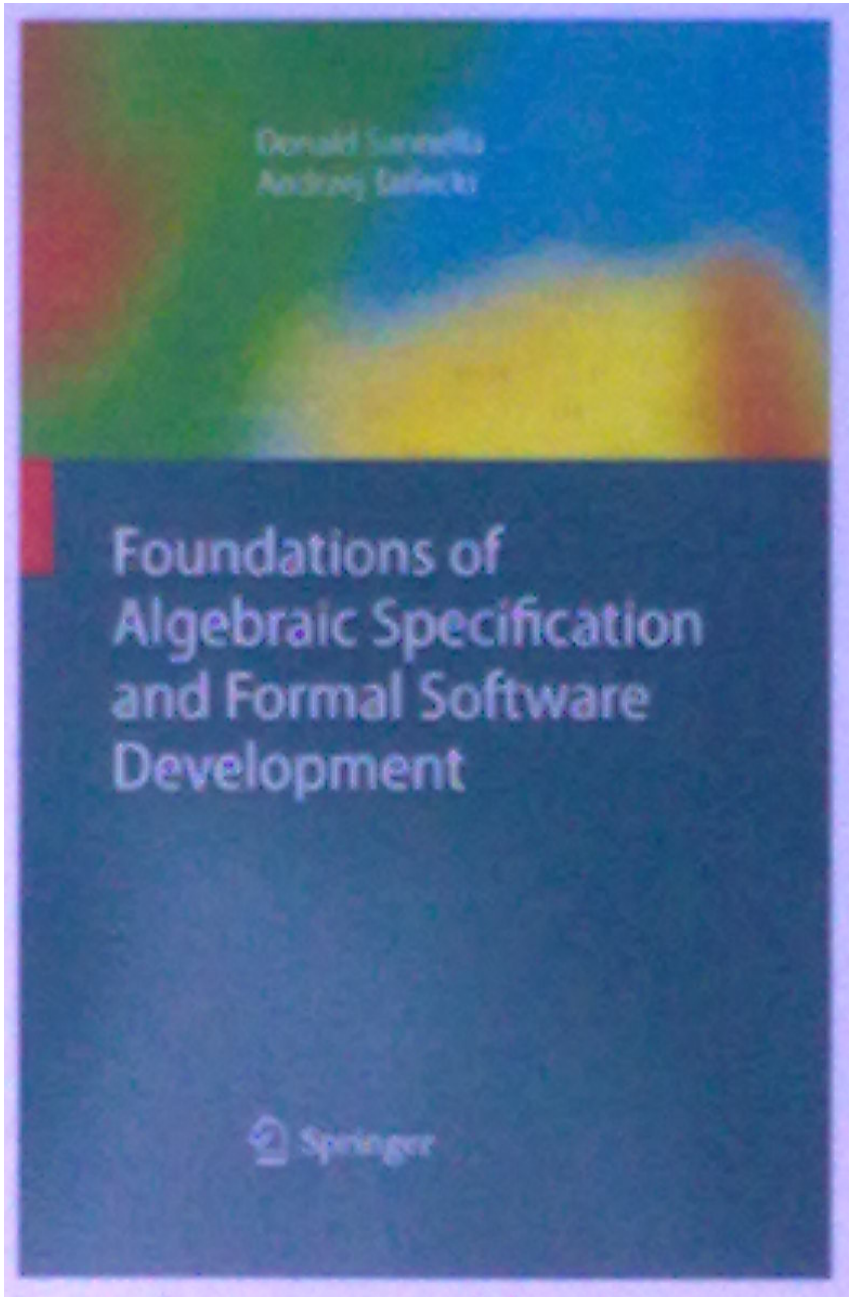
- Introduction
- Good news
- Bad news
- Some lines of research
- Conclusion

QUICK BIO

2000–2004 IRIT/université Paul Sabatier — Toulouse
PhD on isomorphisms of inductive types in simply typed λ-calculus and abstract rewriting methods for higher-order extensional rewriting

2004–2008 CNES — Toulouse
On-board software architect; also in charge of R&D regarding on-board software engineering

2008–... Onera/DTIM — Toulouse
Research on formal approaches for "large-scale"-systems engineering, mainly in aeronautics and space
Aim: theory and methods sparked off by "real" problems

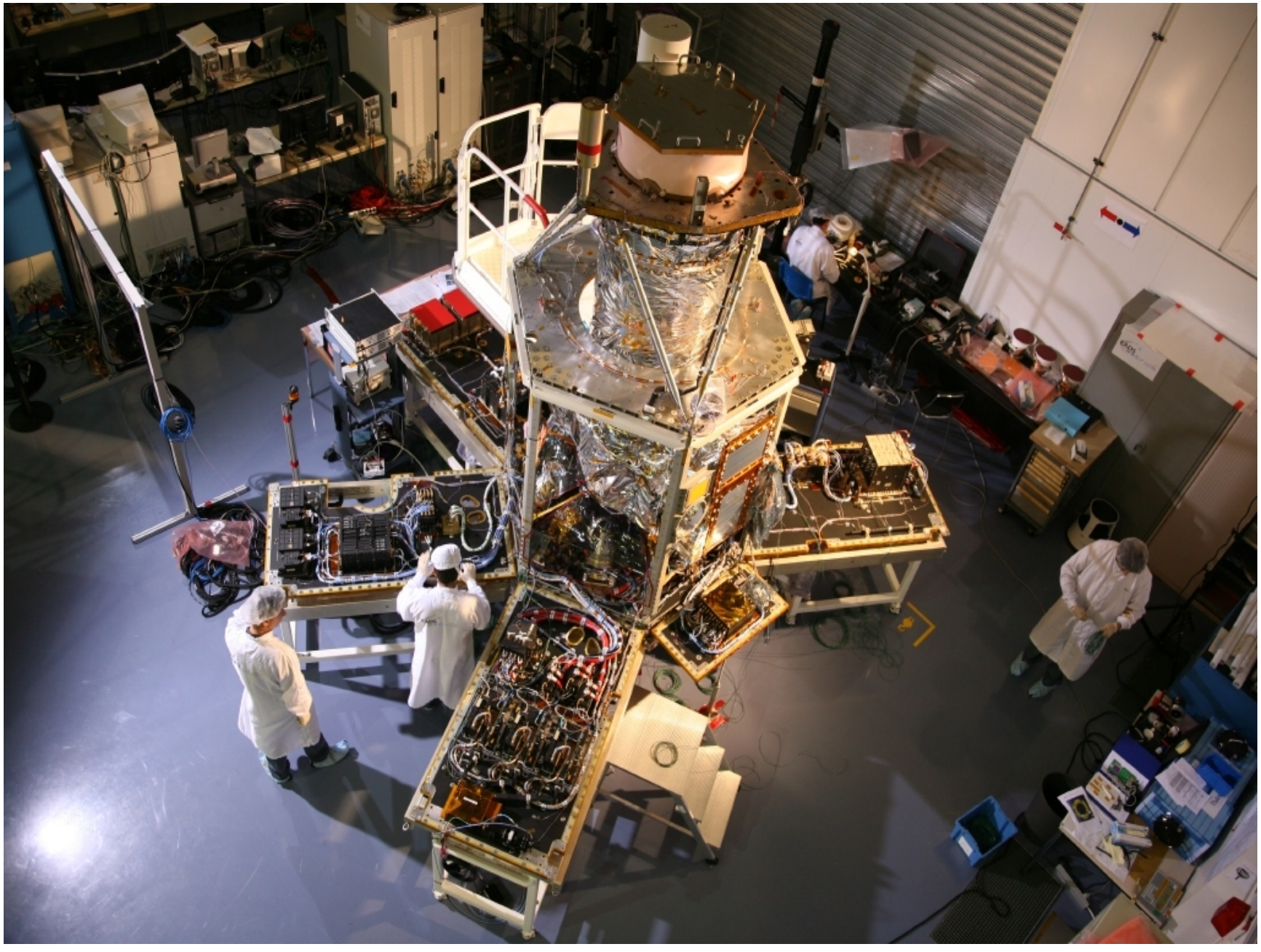


Something I ought to read!

SCOPE AND TERMINOLOGY

- Systems
  - Not necessarily pure software: equipment, humans, organisations...
  - Concurrency, distribution, mobility, asynchronism
- Systems engineering
  - Definition
  - Operation
  - End of life
- System architect

Still, this presentation will suffer a bit from a software bias...



Pléiades HR assembly, integration and tests (© CNES, 2008 – photo Pierre JALBY)

© ESA – P.Carril

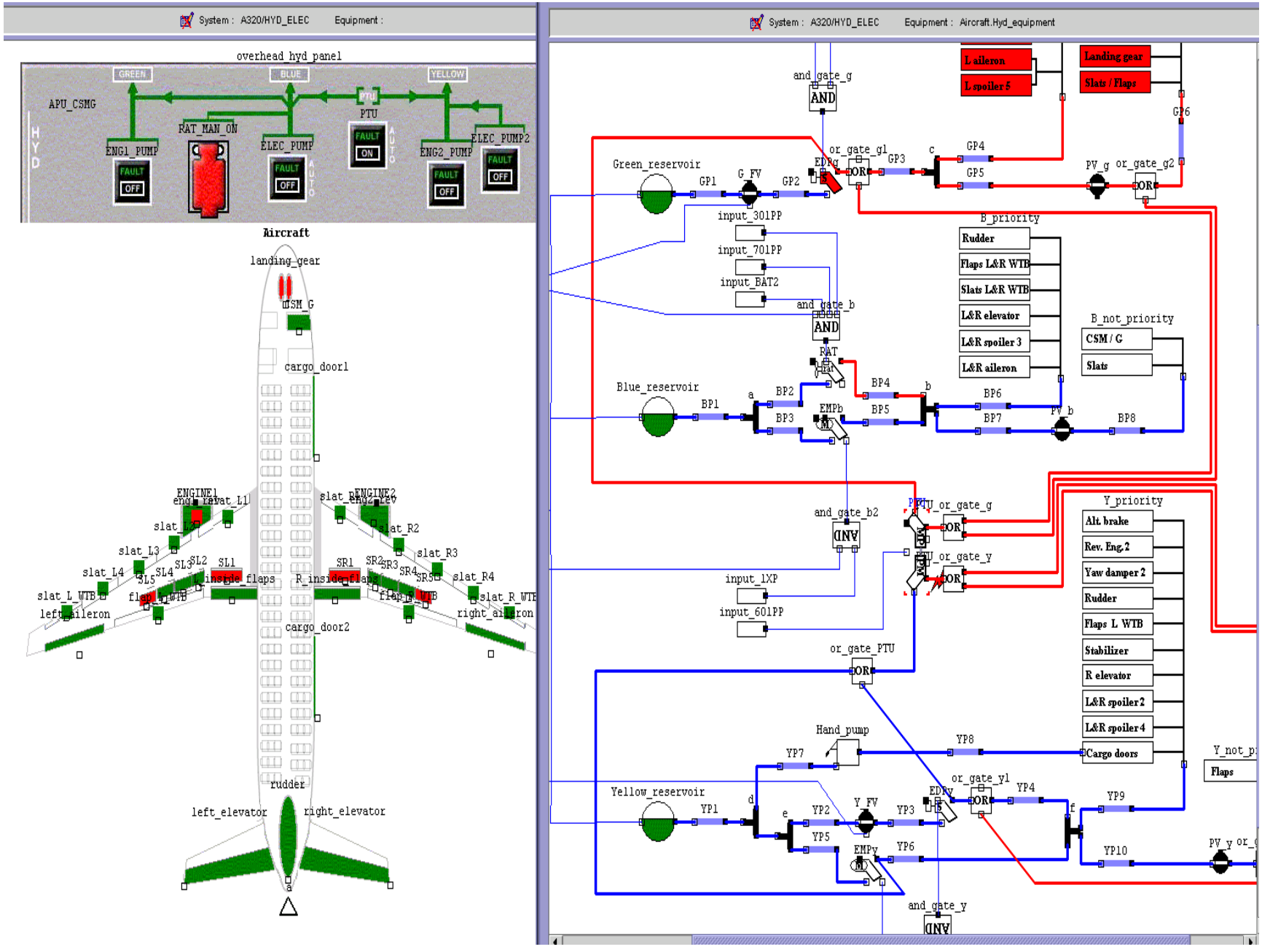
- Introduction
- Good news
- Bad news
- Some lines of research
- Conclusion

MODELLING

- *System* modelling
  - If any, models more intended at documentation
  - Languages often lacking software-engineering constructs
- *Software* modelling
  - Used for years
  - Still fragile, complete model-based processes are rare
- Other fields
  - Commonly used, with specific tools
  - Integration with other fields is a problem

DEPENDABILITY

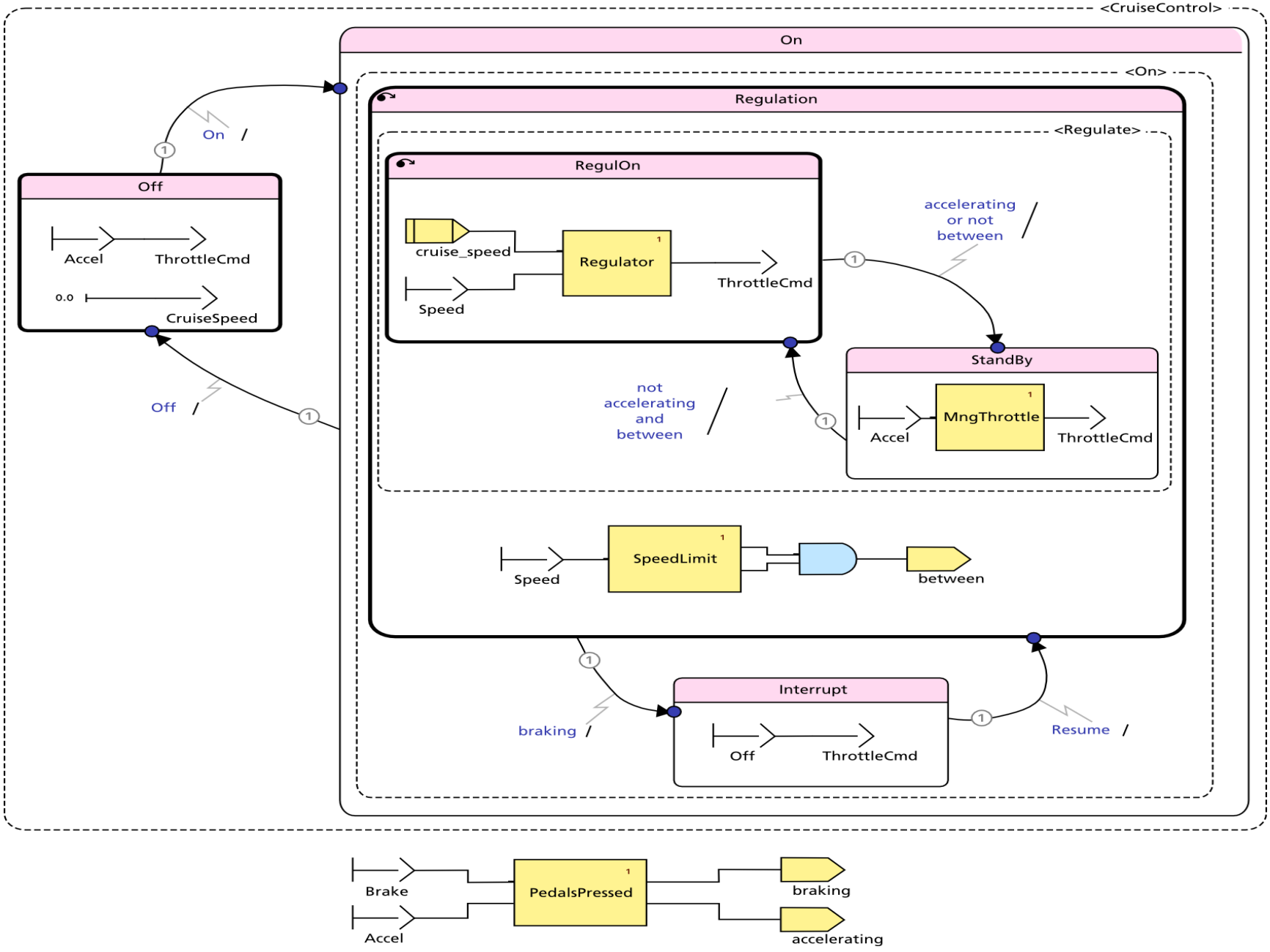
- Cecilia OCAS: graphical notation for Altarica
- Process
  - Modelling of the Dassault Falcon 7X flight control system
  - Generation of fault trees
  - Other possibilities
    - Verification of probabilistic requirements (Monte-Carlo, Markov)
    - Model-checking for qualitative requirements



Cecilia OCAS diagram

CONTROL LAWS

- Scade: graphical synchronous programming language (based upon Lustre and Esterel)
- Used for safety-critical on-board software (Airbus)
- Often presented as an “executable specification” language
- Process
  - Matlab/Simulink/Stateflow → Scade
  - KCG: “qualified” compiler Scade → C (no need for unit testing)



Scade diagram

STATIC ANALYSIS

- Abstract interpretation for code generated from Scade (Astrée)
- Deductive methods (Floyd-Hoare approach) for manual code (Caveat)
- Abstract interpretation for WCET and stack analysis (aiT)

ADOPTION IN INDUSTRIAL STANDARDS

Aeronautics (DO-178):

- Tool qualification
  - Development: KCG
  - Verification: aiT, Stack Analyzer, Caveat
- Product certification[1]
  - DO-178B: tests or *ad hoc* exceptions
  - DO-178C (forthcoming): specific objctives for formal verification

[1] Certification is a necessary condition for airwothiness.

- Introduction
- Good news
- Bad news
- Some lines of research
- Conclusion

INDUSTRIAL ORGANISATION

- Suppliers not only chosen for technical reasons
  (*e.g.* cost, geo-return, global contracts, business strategy, national interests, military restrictions...)
- Sometimes your supplier may also be your client
- Sometimes many suppliers for a same product
- Staff turnover both at client and supplier levels
- Sharing technical data can be difficult (security, IP, tool support)

INFORMATION

- Unsatisfactory requirements engineering
- Documents everywhere
  - Best durability
  - Microsoft Office is the only universal tool!
  - Documents as contracts
  - Certification
- Technical database
- Configuration management

Pléiades HR

- Thousands of (versioned) documents
- Thousands of technical data

PROCESSES

- Reasonably well-documented processes
- Not so much concerned with methodology
- Implemented through company-specific workflows and/or tools
- Importance of traceability
- Aeronautics: directly linked to certifiability

COMPETENCIES

- Importance of seasoned experts
- Noble disciplines (control theory, aerodynamics…)
- Software often viewed as a supporting technology
- Strongly varying education (including in certification authorities)
- Outsourcing of “non-core” competencies
  - Loss of internal knowledge
  - Pressure on service providers

No shared understanding of simple CS or maths concepts:

- Use cases, n-ary associations (with $n > 2$), association classes, class, object, component…
- Injections, surjections, equivalence relation…

OPERATIONS

- Long lifespan w.r.t. people, organisations, technologies...
- Maintenance
  - Minor fixes
  - Requirements and design evolution
  - System updates
- Increasing interest for diagnosis, prognosis, autonomy and self-adaptation

Aeronautics

- Total lifecycle up to 80 years
- On-ground static software updates

Space

- Total lifecycle up to 30 years
- In-flight dynamic software updates

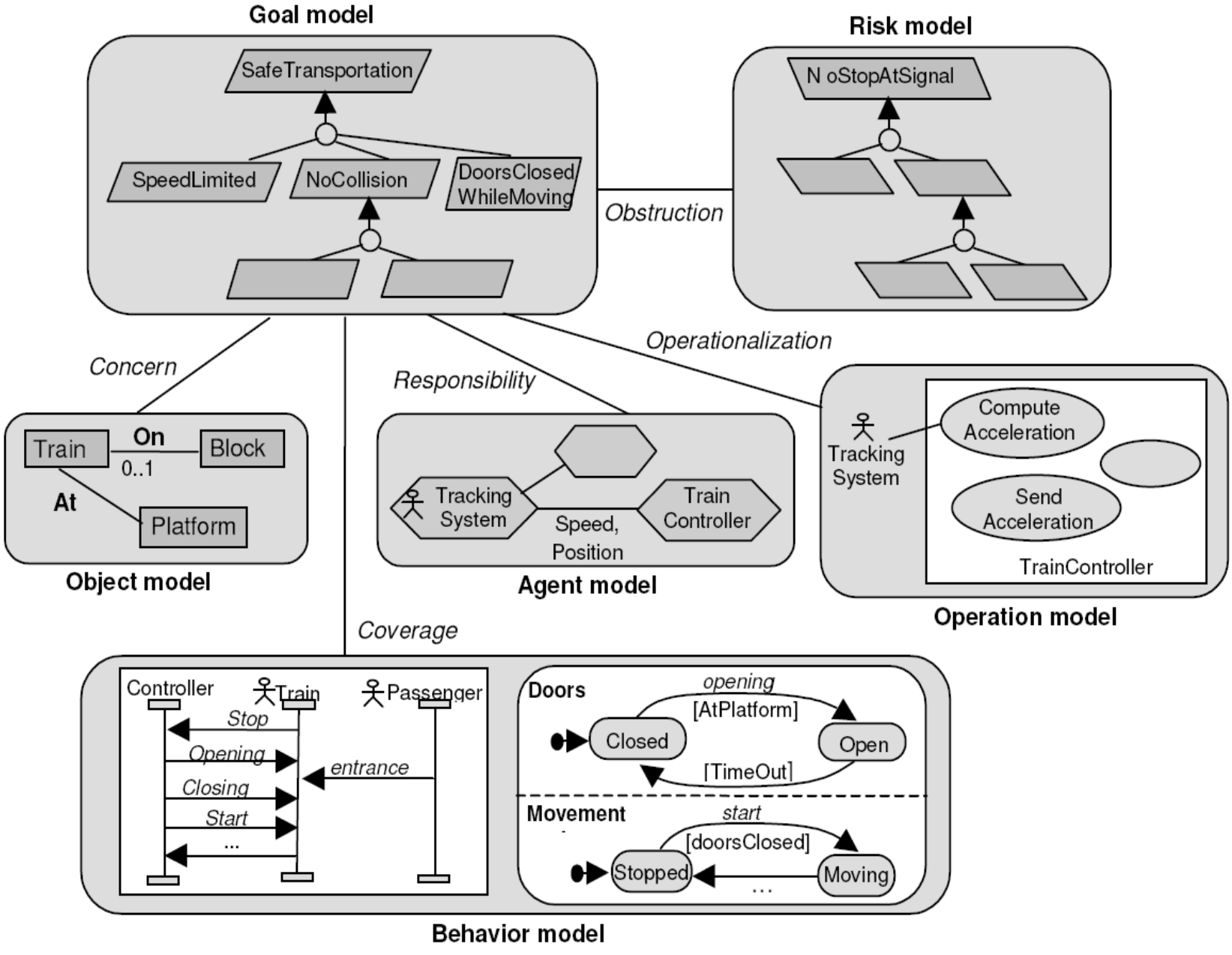
TECHNOLOGY

- Some important drivers for technological choices:
  - Numerous and skilled workforce
  - Many product and service providers
  - Standardised technology and COTS
- Increasing importance of:
  - Software size (A380 avionic software: 12MB)
  - Security
  - Space and time partitioning
  - Multi-core chips
  - Reconfigurable chips (FPGA)
  - Simulation

- Introduction
- Good news
- Bad news
- Some lines of research
- Conclusion

SOME PROBLEMS

- Focus on early system-definition activities
  - **Requirements engineering**
  - Functional analysis
  - System architecture
- Distribution & concurrency
- Fault tolerance
- Certification
- Evolution management
- Maintainability
- Sustainability
- Continuous and stochastic modeling


Goal model
SafeTransportation
SpeedLimited
NoCollision
DoorsClosed WhileMoving
Risk model
N oStopAtSignal
Obstruction
Concern
Responsibility
Operationalization
Train
On
Block
0..1
At
Platform
Object model
Tracking System
Speed, Position
Train Controller
Agent model
Tracking System
Compute Acceleration
Send Acceleration
TrainController
Operation model
Coverage
Controller
Train
Passenger
Stop
Opening
Closing
Start
...
entrance
Doors
opening
[AtPlatform]
Closed
Open
[TimeOut]
Movement
start
[doorsClosed]
Stopped
Moving
...
Behavior model

TRIVIALITIES

- Gentle learning curves
- Familiar vocabulary and notations
- (Reasonably) easy-to-explain intuitive semantics
- Important return on personal investment
- Scalable methods (industrial projects are huge)
- Integration into existing processes and with existing tools (provide a parser and an XML pretty-printer)
- Adapted to the extended enterprise/supply chain

CURRENT INTERESTS (WITH COLLEAGUES)

Inspirations:

- Logic, algebra, categories
- KAOS, Tropos, B, Event-B, Alloy, TLA...

Topics:

- Goal-oriented requirements engineering
  - ATL for KAOS goal and obstacle viewpoints
  - Extensions/modifications of KAOS, other propositions
    - Security, mobility...
- System architecture description language
  - Tom Maibaum's 2 last items of today's presentation :-)
  - Community-like? Bi-algebraic semantics?
- Evolution and reconfiguration
  - Categorical characterisation *in abstracto*
  - Reconfiguration techniques
- Viewpoints in modelling
  - Hets

- Introduction
- Good news
- Bad news
- Some lines of research
- Conclusion

CLAIM

- Most industrial problems are not of functional nature.
- However, some of them may be handled (partly) formally.
- Still, they may have not been fully explored by the formal methods community.
- Dealing with them may actually lead to a quite better adoption of formal methods in the industrial world.