Space Systems Dependability
The hybrid (modelling) necessity

Jean-Paul Blanquart

ASTRIUM Satellites

Jean-paul.blanquart@astrium.eads.net

5th Latin-American Symposium on Dependable Computing

INPE, São José dos Campos, Brazil, April 25-29, 2011

All the space you need

ASTRIUM
AN EADS COMPANY

Outline

- A few words about EADS, Astrium, …
- Dependability in space
  - Constraints, needs, solutions, achievements
- Dependability (FDIR) process
- Model Based Dependability
  - Engineering, Assessment
- Why it may become so complicated?

EADS
European Aeronautic Defense and Space

EADS

AIRBUS AN EADS COMPANY | ASTRIUM AN EADS COMPANY | EUROCOPTER AN EADS COMPANY | CASSIDIAN AN EADS COMPANY

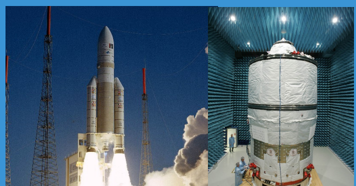
Astrium

Astrium is a global space industry leader, with world-class expertise and extensive prime contractorship experience across all sectors of the space business

3 Key domains

Astrium Space Transportation

The European prime contractor for civil and military space transportation and manned space activities



Astrium Satellites

A world leader in the design and manufacture of satellite systems

Astrium Services

At the forefront of satellite services in the secure communications, Earth observation and navigation fields

Outline

- A few words about EADS, Astrium, …
- Dependability in space
  - Constraints, needs, solutions, achievements
- Dependability (FDIR) process
- Model Based Dependability
  - Engineering, Assessment
- Why it may become so complicated?

Space systems: Constraints

- Limited mass, power
- Limited ground-board link
- Limited maintenance
- Radiations
- Knowledge, mastering of the environment
- Phased missions, critical parts
- Long lifetime

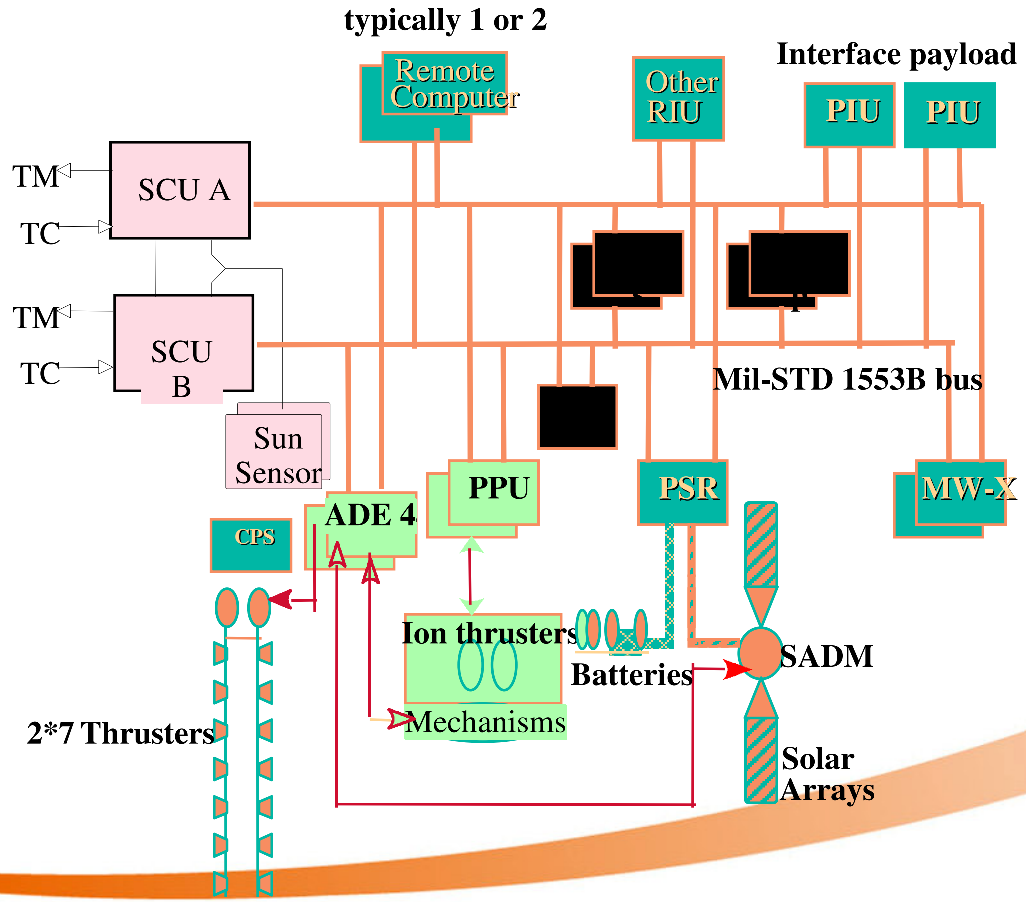
A large variety of dependability needs

- Reliability
  - Lifetime (satellites, space probes)
  - Continuity of service (launchers, rendez-vous, re-entry)
- Availability
  - Instantaneous (satellite, probes critical phases, launchers)
  - Average (Mission return)
  - Outage duration, frequency (critical or expensive services)
- Maintainability, adaptation (reconfiguration, software, procedures)
- Safety (Launchers, manned flights, rendez-vous, end of life)
- Security

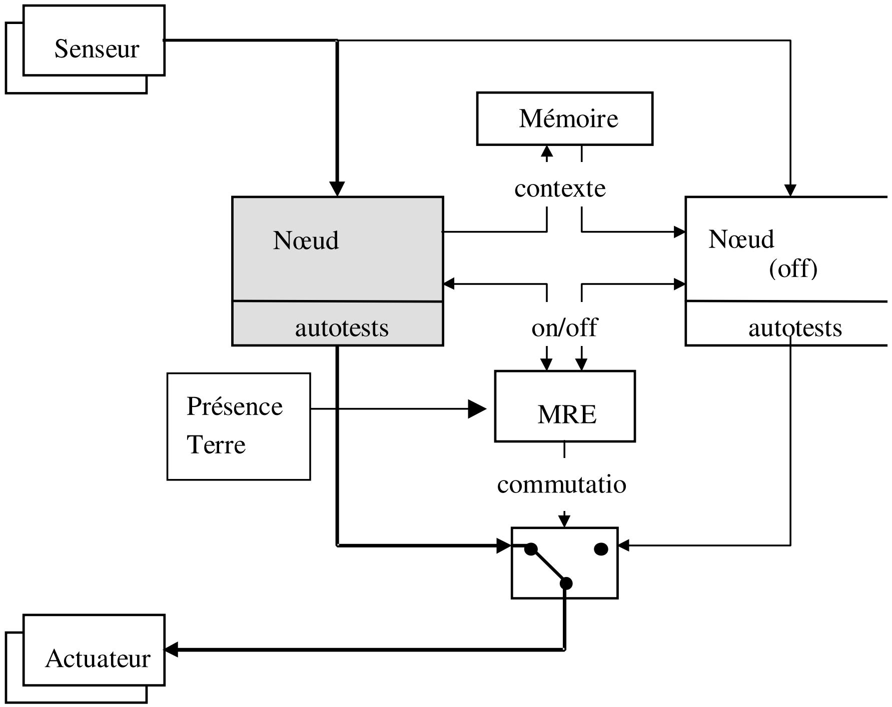
Solutions, basic principles

- Selective redundancy, hot or more often cold duplex
  - Some notable examples of comparison and vote
- Automatic detection and reconfiguration (FDIR)
- Safe mode
- “Favourite” adversaries
  - Single point of failures, common cause failures, failure propagation
  - Unpredicted situations, lack of observability, controllability

Classical satellite architecture



Computer failure (cold duplex)



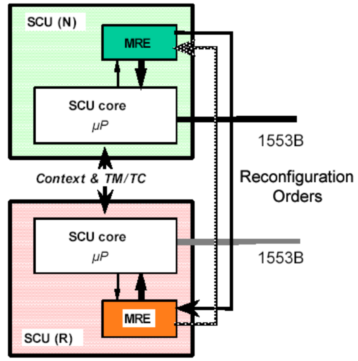


Figure 7.1.1.3/1 : SCU Architecture

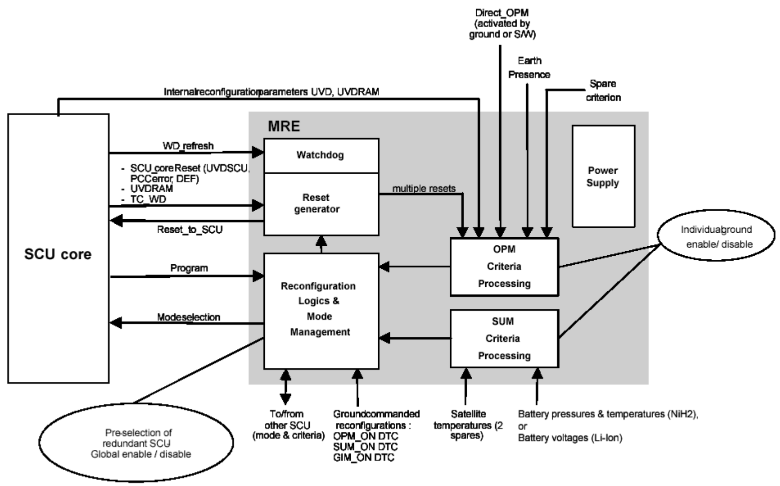
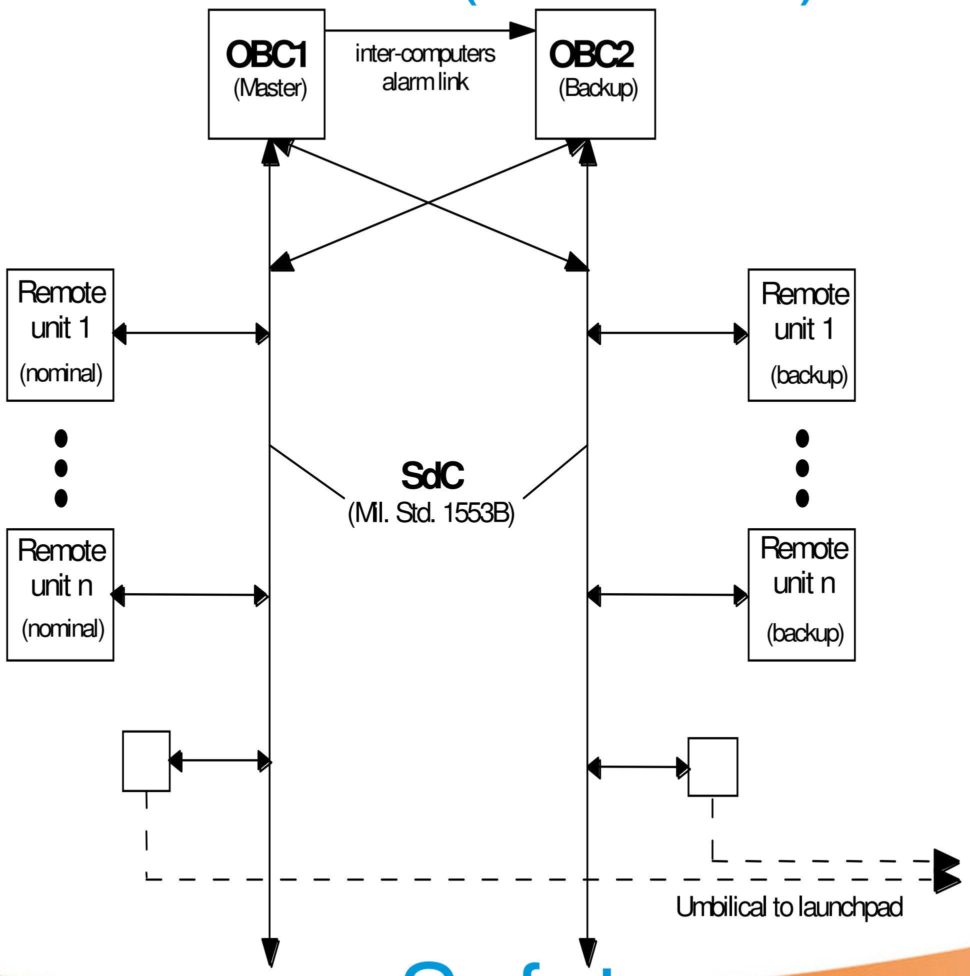


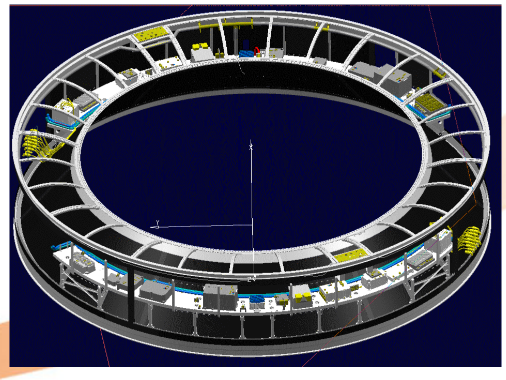
Figure 7.1.1.3/2 : MRE architecture

ASTRIUM
AN EADS COMPANY

Launcher (Ariane 5)

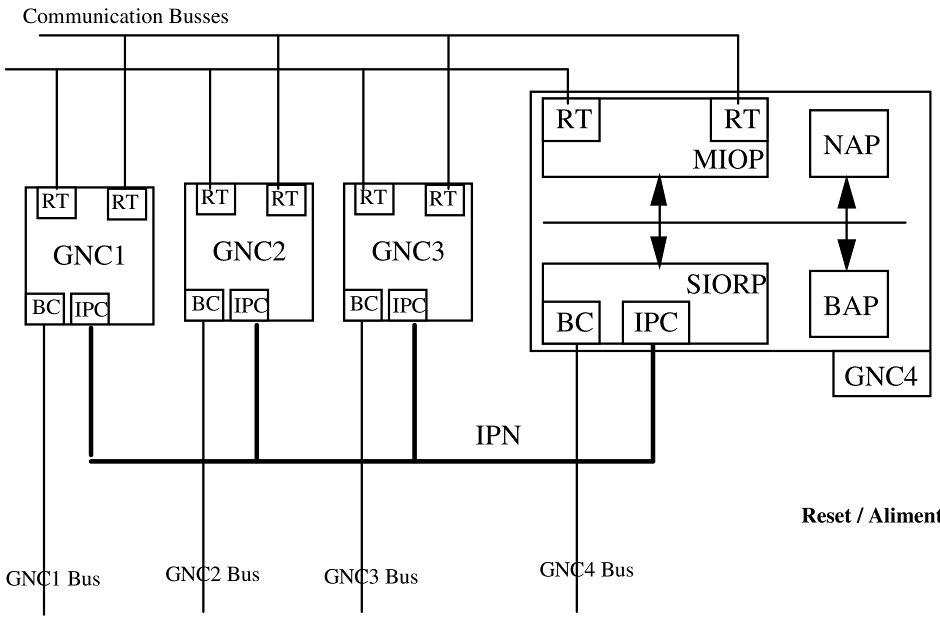


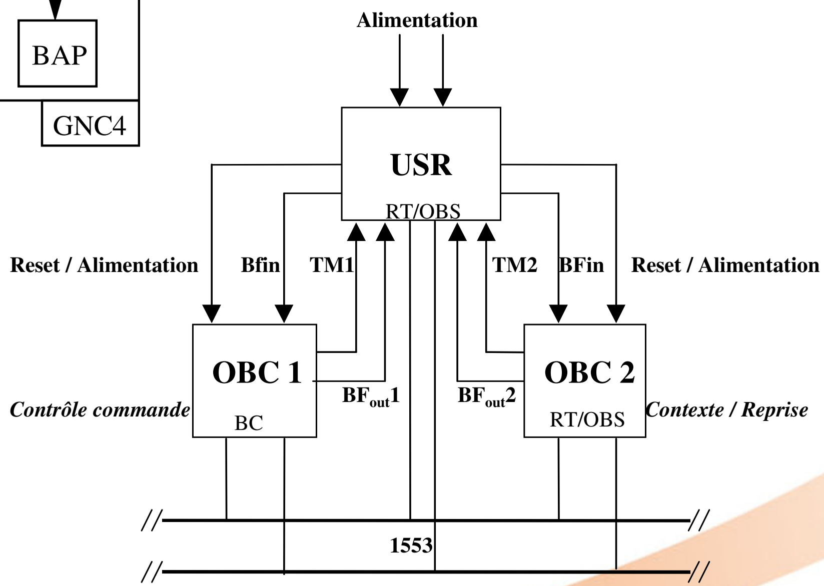
Hot-Duplex,
semi-cross-strapped,
one-shot



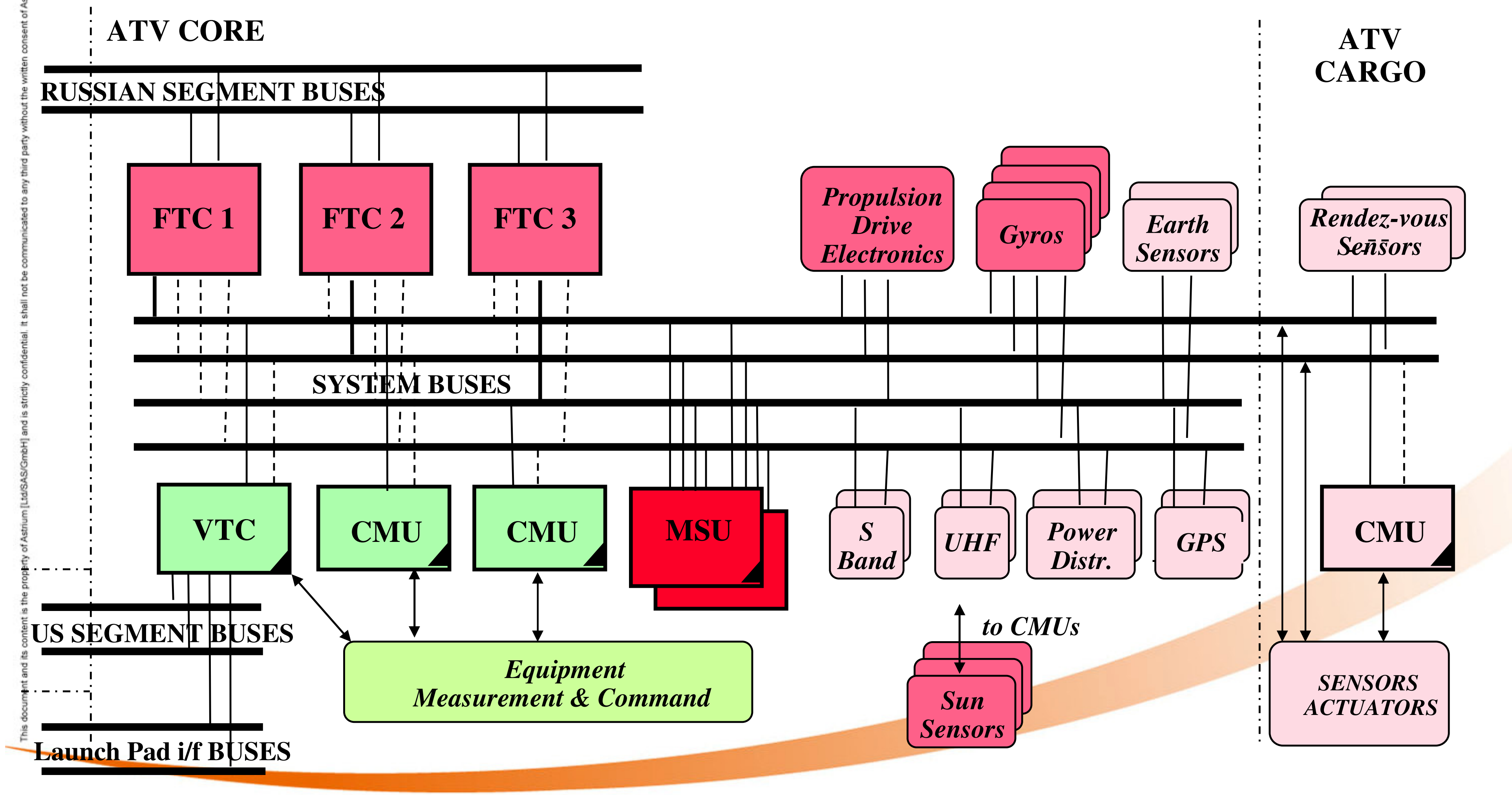
+ Safety

Manned automatic vehicles (projects)

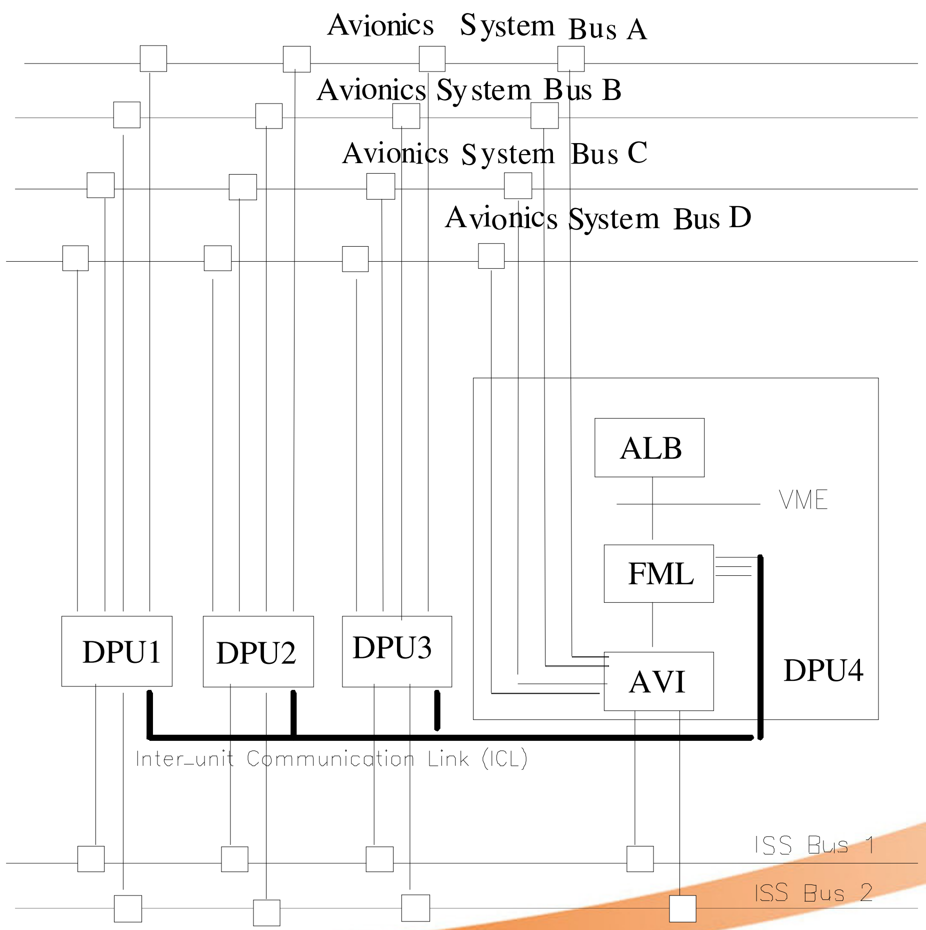




ATV (Automatic Transfer Vehicle)



ATV Fault Tolerant Computer

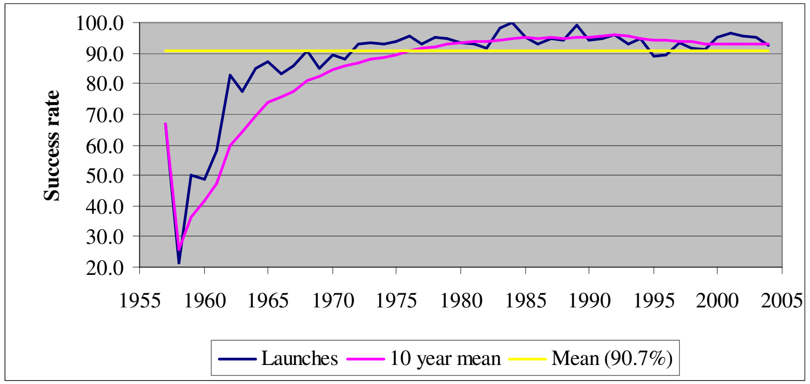


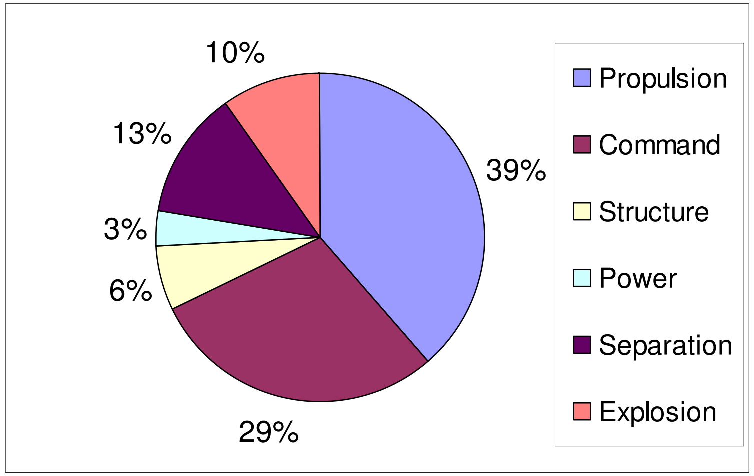
And it works…

Indicative values, from public data (up to 2005)

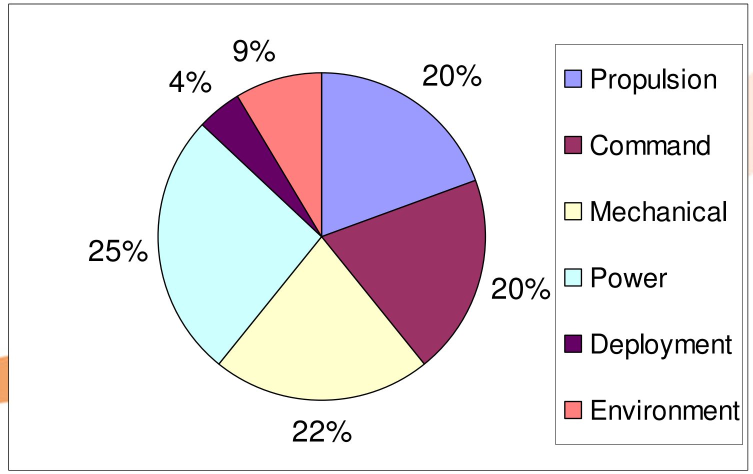
No pretention as strongly substantiated statistics

- Launchers





- Satellites
  - "~10^{-6}/h" 2xlifetime, 90%>
  - But:
    - Launch: 6-7%
    - In-orbit installation: 4-5%
    - Early phase: 1.5 10^{-6}/h
    - Life: 0.5 10^{-6}/h

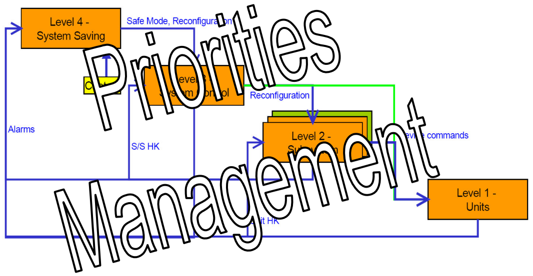


ASTRIUM
AN EADS COMPANY

Outline

- A few words about EADS, Astrium, …
- Dependability in space
  - Constraints, needs, solutions, achievements
- **Dependability (FDIR) process**
- Model Based Dependability
  - Engineering, Assessment
- Why it may become so complicated?

Dependability process (FDIR)

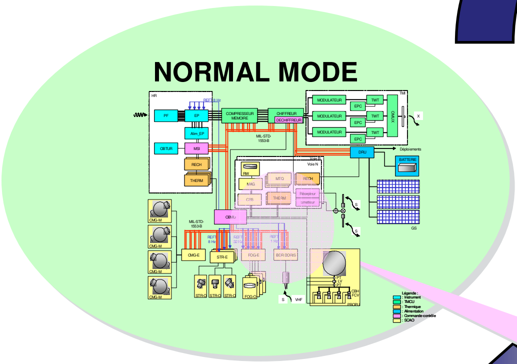


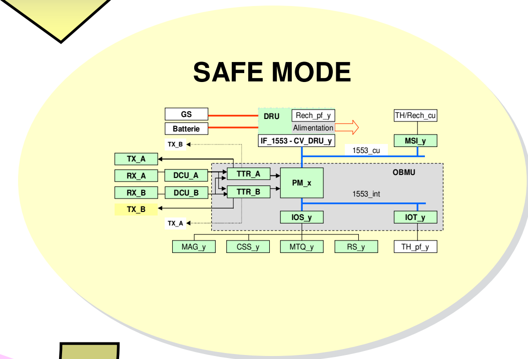
REQUIREMENTS

- One Failure tolerant design
- $R(t) \geq 0,8$
- $D(t) \geq 0,99$
- Autonomy …
- …

Events → HW ? SW ? GCC ?

Events → SW ? GCC?





Events → HW ? SW ? GCC?

Events → HW ? SW ? GCE ?

FDIR

- **REQUIREMENTS**
- **ARCHITECTURE**
- **Modes versus "failures"**
- **HIERARCHY**
- **FD/I/R Management**

GCC: Ground Control Centre

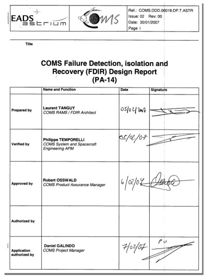
Failures Management

- **DETECTION**
How ? Which parameters ? Frequency ?...
- **ISOLATION**
Protections, Time to react ?...
- **RECOVERY**
Which actions ? Who executes ? …

ASTRIUM AN EADS COMPANY

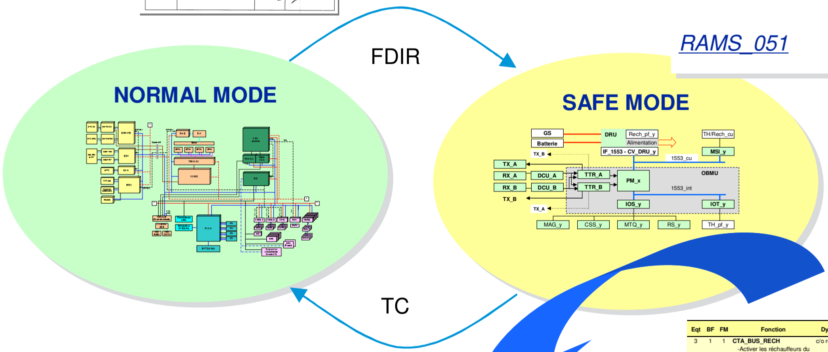
FDIR analysis

REQUIREMENTS



RAMS_001 « Every failure likely to propagate shall be detected in appropriate time in order to avoid propagation to another reliability block (as defined in teh reliability block diagram) »

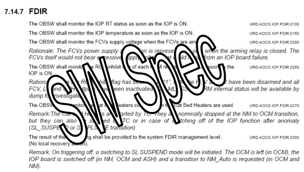
STRATEGY

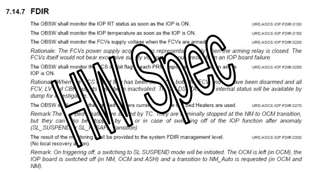


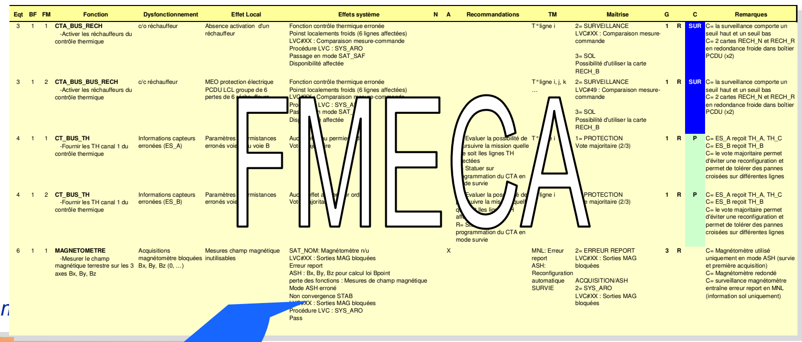
RAMS_051 « Every failure with criticality 1 shall trigger a safe mode »

DESIGN

F / D / I / R





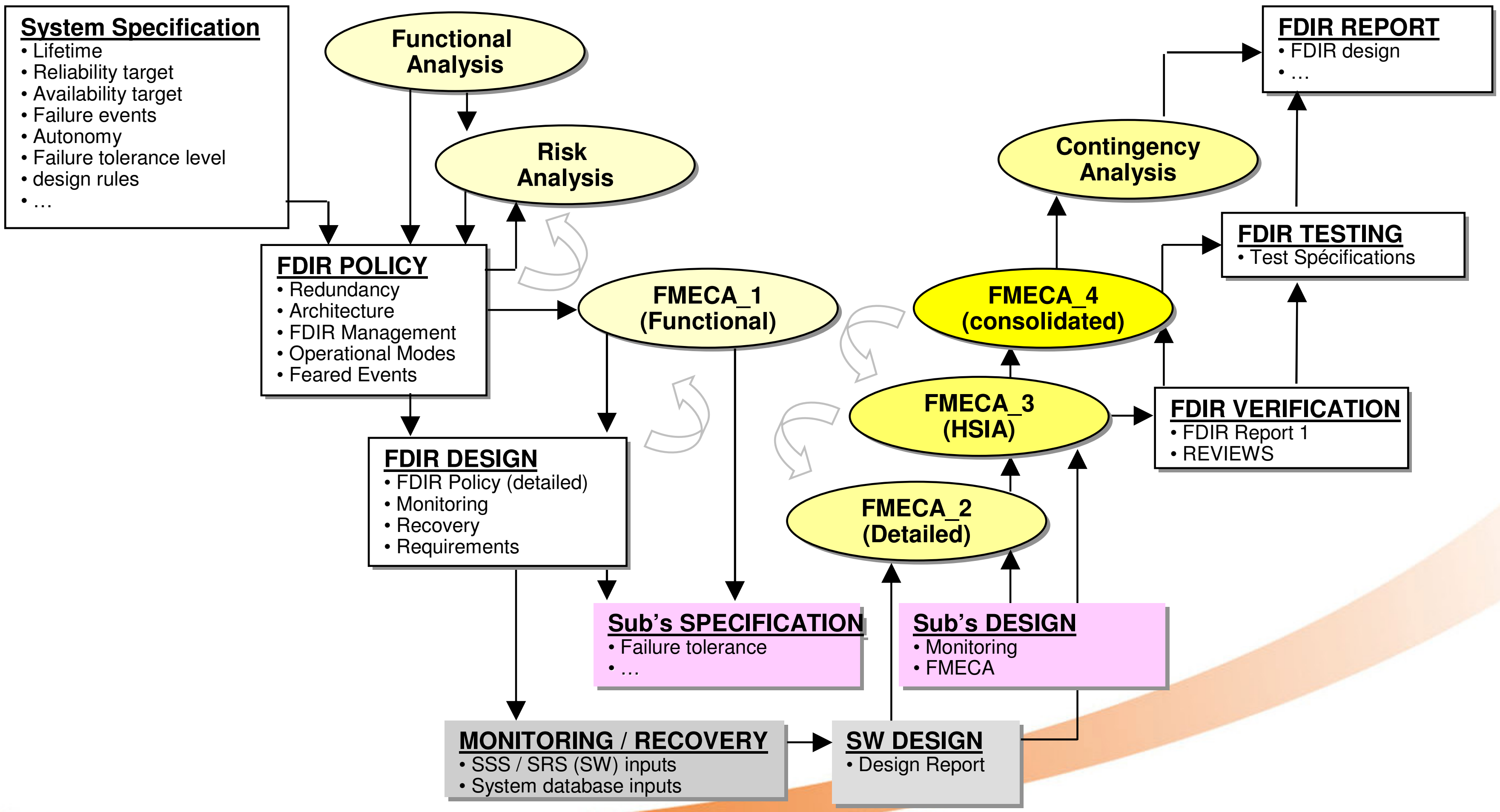


RAMS_101 « An electrical protection shall protect the spacecraft from any short-circuit down-stream

RAMS_151 « In order to detect ASH control failure during stabilization phase, the OBSW shall monitor the duration of the stabilization phase. Triggering of this surveillance shall lead to ARO. (URD.AOCS.ASH.FDIR.0100)

V&V

FDIR lifecycle



ASTRIUM
AN EADS COMPANY

Outline

- A few words about EADS, Astrium, …
- Dependability in space
  - Constraints, needs, solutions, achievements
- Dependability (FDIR) process
- **Model Based Dependability**
  - **Engineering, Assessment**
- Why it may become so complicated?

Types of models (for dependability) determined by the objectives

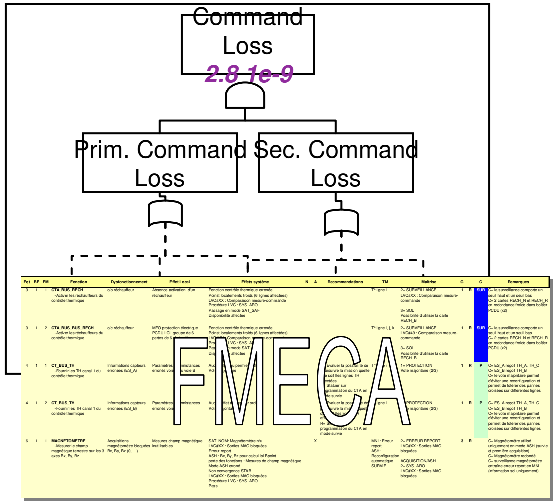
- Quantitative (probabilistic) analysis of RAMS properties
  - Modelling the impact of faults/failures on the level of performance
    Markov, RBD, …
- Qualitative analysis of faults/failures propagation
  - Modelling the impact and propagation of faults/failures on on the architecture (physical, functional, …)
    Structural models, AADL, …
- Assessment (correctness, performance) of FDIR
  - Modelling the behaviour of the FDIR
    State machines, temporised automata, model-checking, …
- Soundness, completeness of the safety, dependability arguments
  - Modelling the dependability and safety argumentation
    Logical formulas, GSN, …

Main types of dependability models

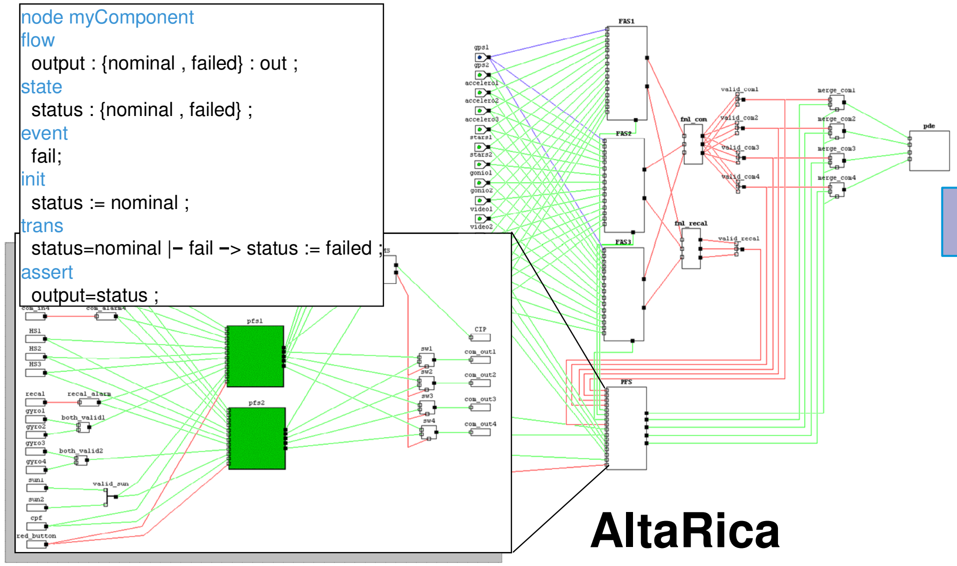
- Behaviour (in presence of faults)
  - Needs for an appropriate abstraction of the behaviour
  - Behaviour of the system (functional, « dys-functional »)
  - Behaviour of the fault processing mechanisms
  - Assembly of functional blocks (on, off, failed, …)
- Architecture and fault propagation
  - Explicit representation of fault propagation, with an abstraction of the behaviour
- Coupling behavioural and structural models
  - At least for FDIR mechanisms
  - Impact of faults on behaviour
  - Impact of reconfigurations on behaviour

Quite an old story
with limitations and solutions

Fault Trees

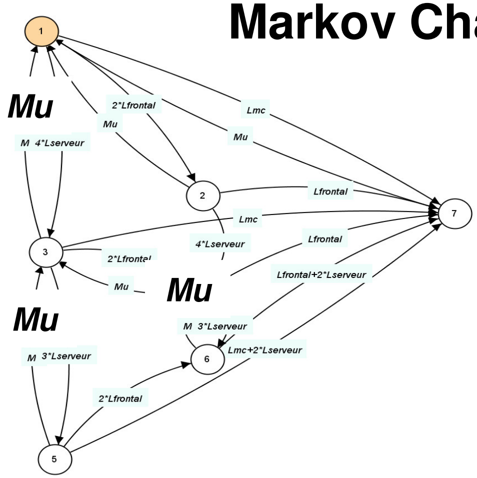


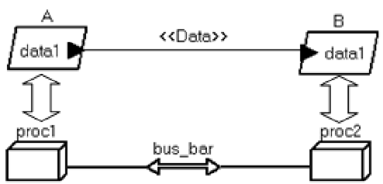

```
node myComponent
flow
  output : {nominal , failed} : out ;
state
  status : {nominal , failed} ;
event
  fail;
init
  status := nominal ;
trans
  status=nominal |- fail -> status := failed ;
assert
  output=status ;
```



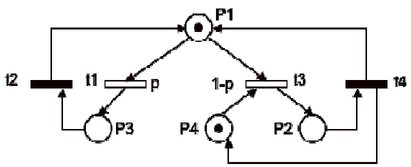
AltaRica

Markov Chains





AADL



GSPN

Cf. Ana-Elena Rugina PhD, 2007

Main advantages

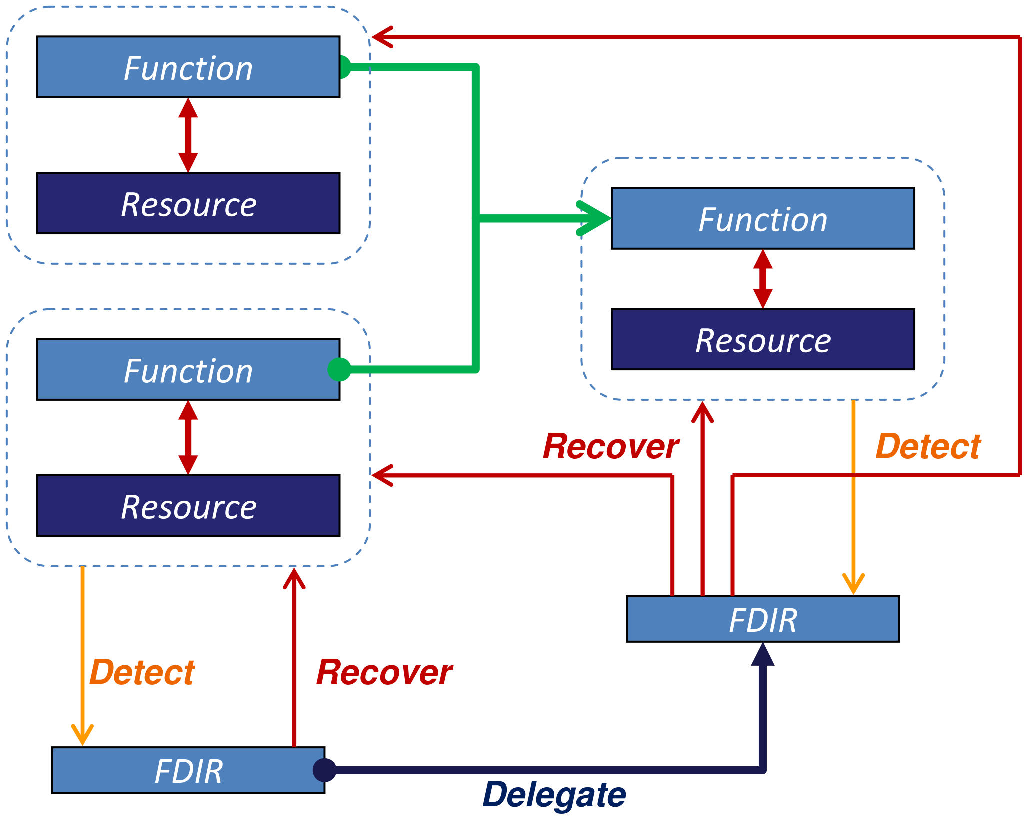
- Support to model creation, discussion, validation
- Additional capabilities (property validation, …)
- Easier updates
- Coupling with other models
- Could/should we go further?
  - What about correctness of fault tolerance mechanisms?

Objectives & Constraints

<u>Need</u>: Model the system dynamics in the presence of faults

- Define a modelling technique
  - Express faults and failures propagation
  - Specify Fault, Detection, Isolation and Recovery mechanisms
- Demonstrate properties on Dependability/FDIR
- Appropriate modelling languages & tools

Requirements on modelling techniques (1/2)



- Deployment on resources
- Functional dependencies
- Spread FDIR functions
- FDIR Hierarchy

Functional & Dysfunctional time constraints

Requirements on modelling techniques (2/2)

Design objectives

- *Simple (compositional) modelling method*
- *Close to the engineering model*

Architecture/Behaviour (AADL)

- Deployment on resources
- Functional dependencies
- Spread FDIR functions
- FDIR Hierarchy
- Time constraints

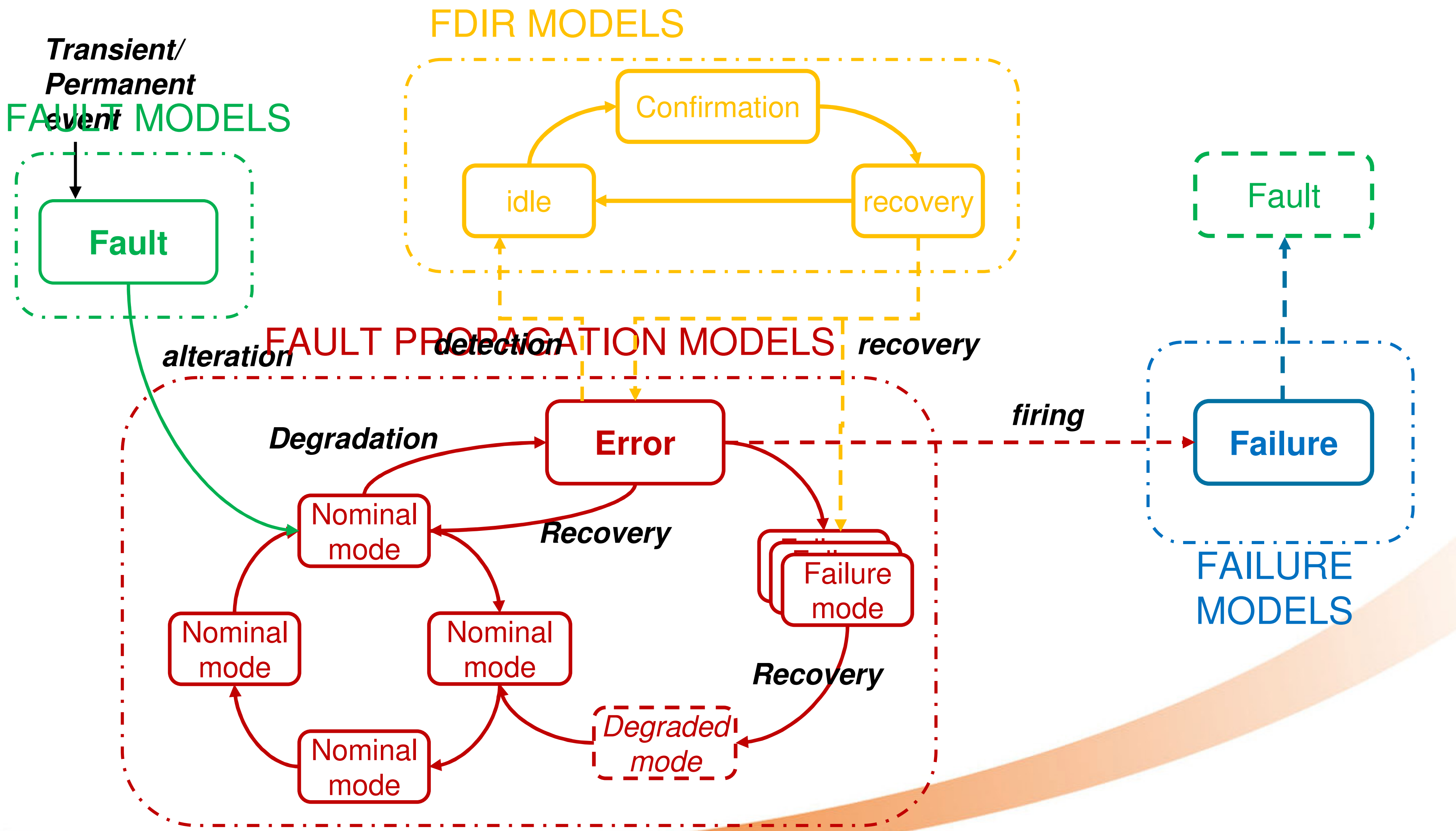
Transformation

Validation objectives

- *Demonstration*

Timed automata (UPPAAL)

Modelling method – Dependability pattern



Modelling method

F4 :Data1 absent

=> {Data/ absent/…}

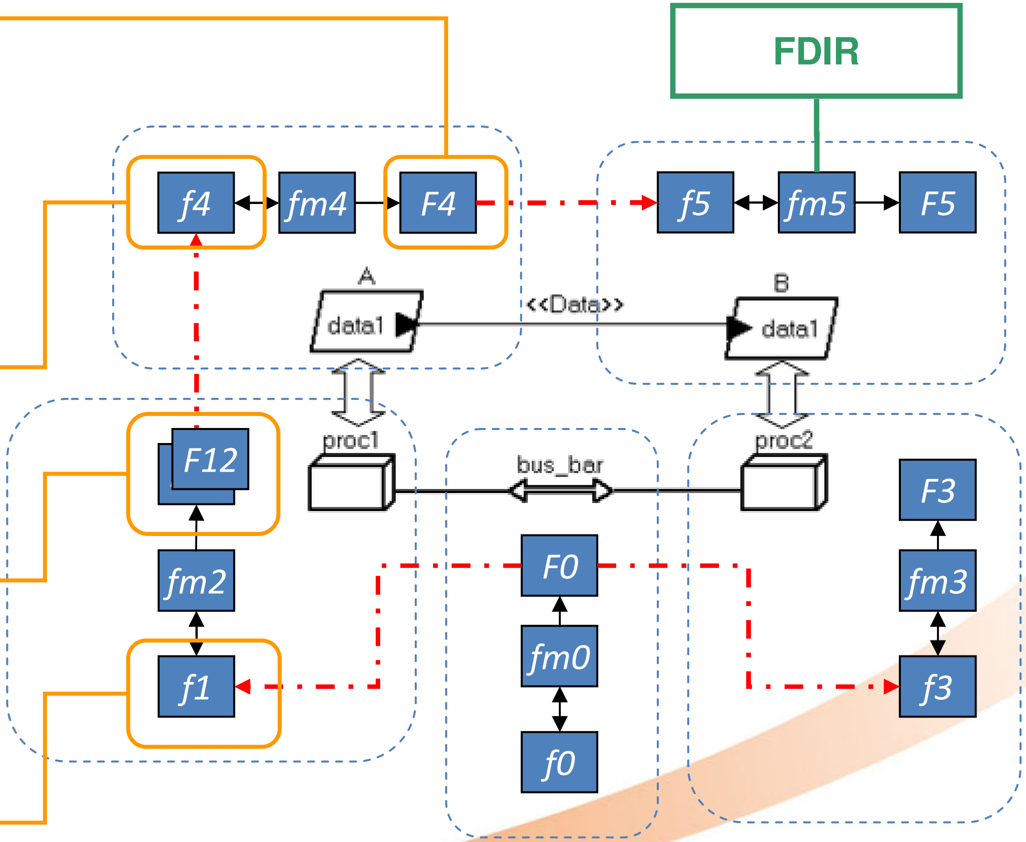
f4: => {resource/processor/computing/…}

F1 : total loss of computing power

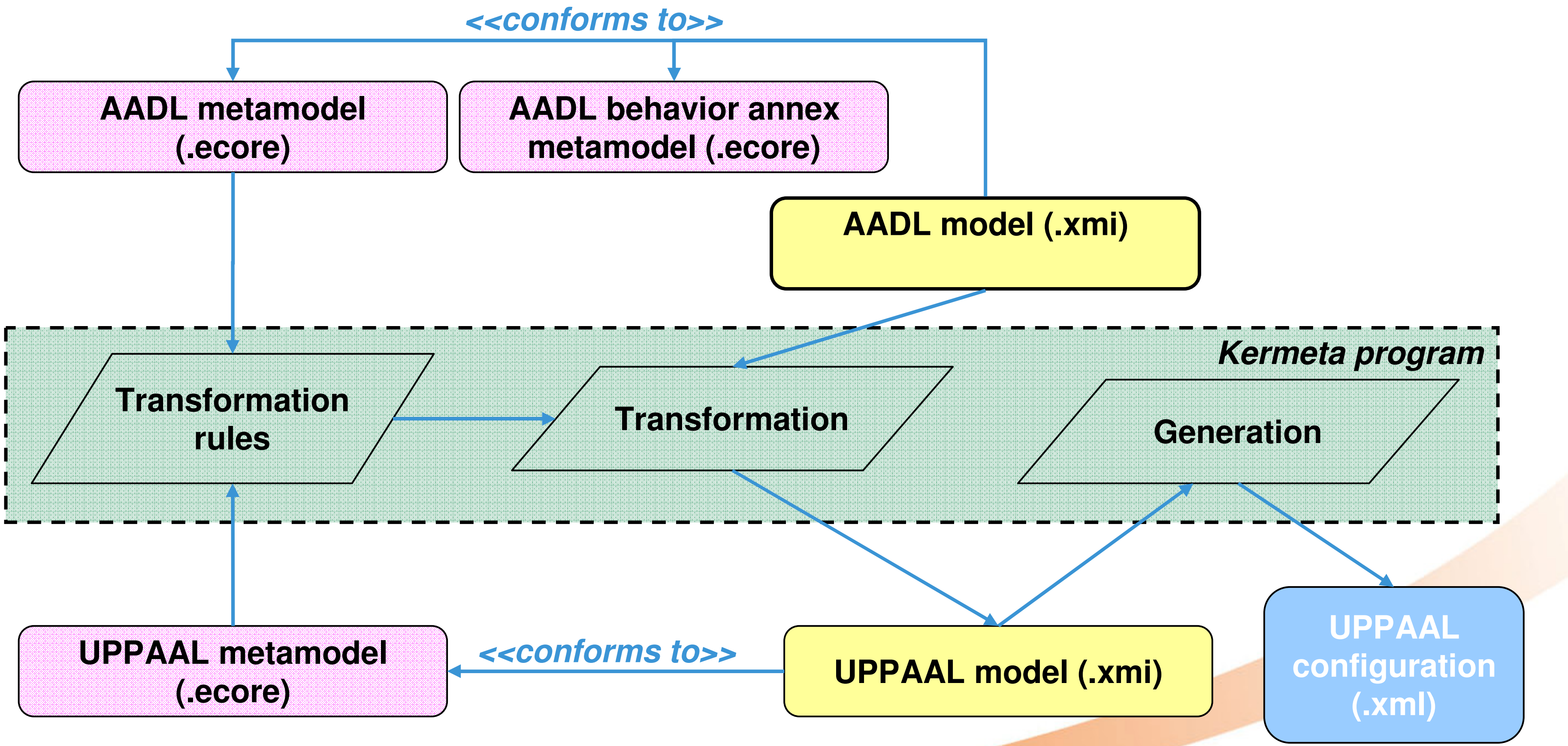
=> {resource/processor/computing/…}

f1 : CPU internal fault

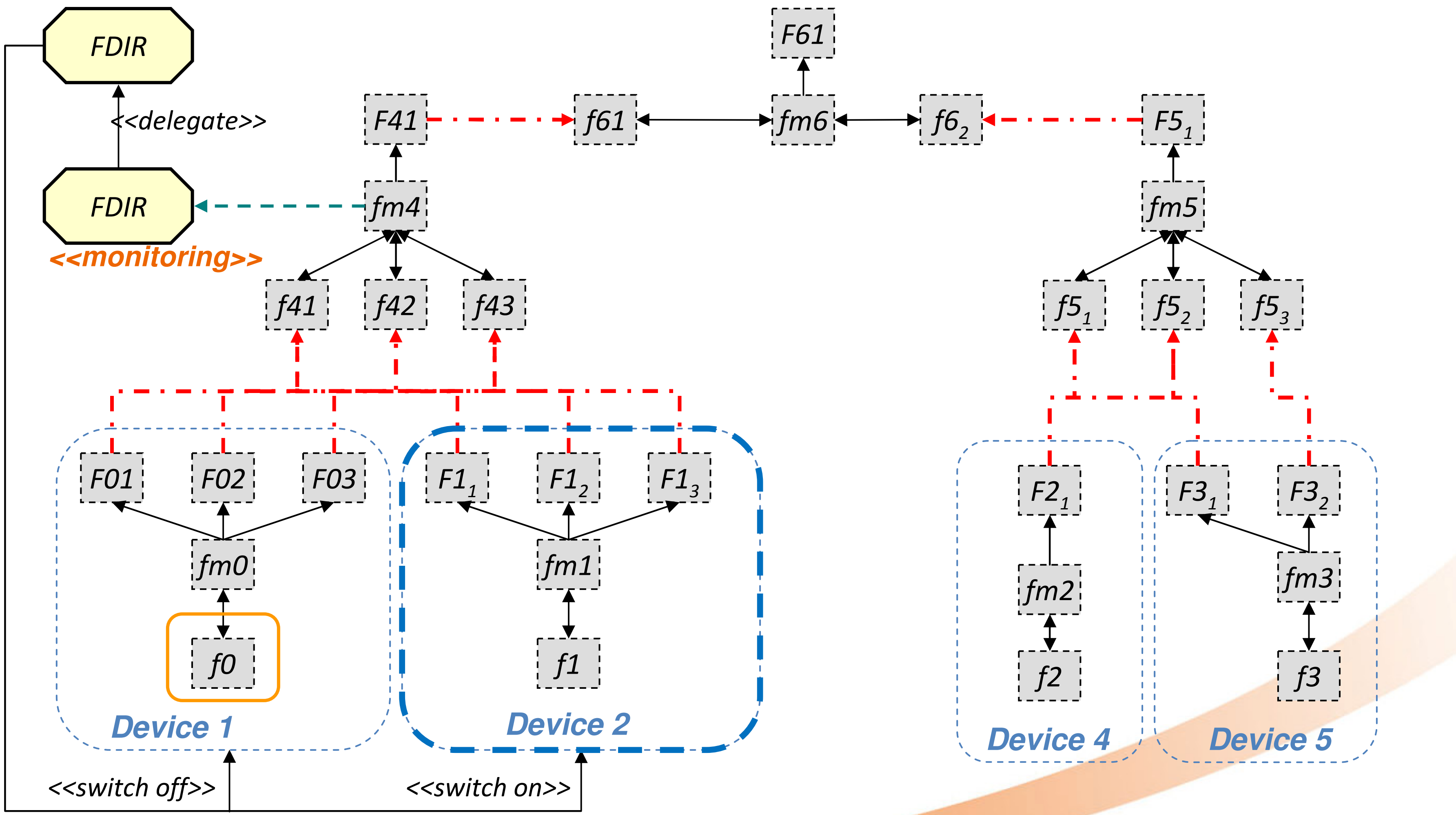
=> {resource/ processor/ computing/…}



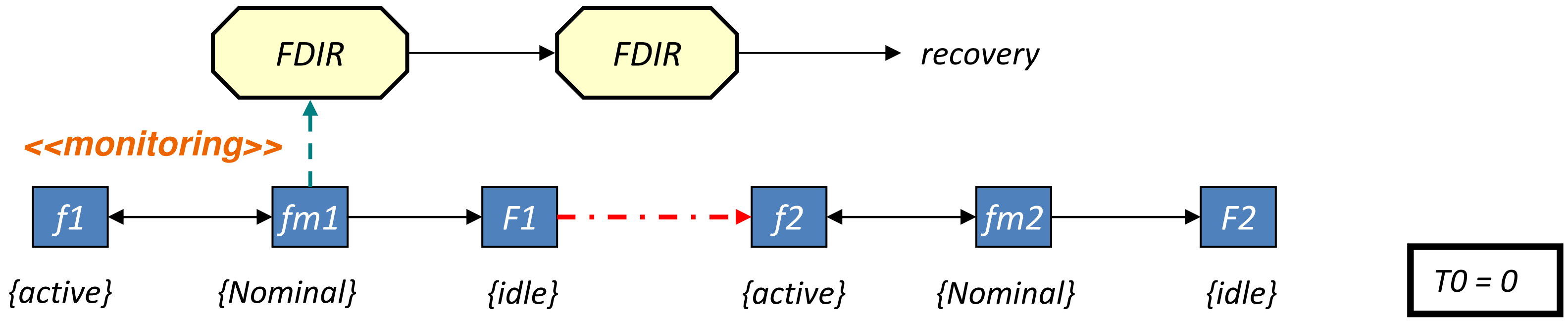
Model transformation



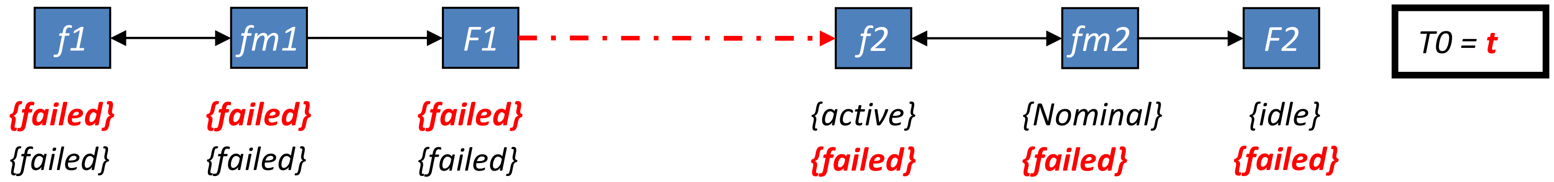
Model reduction – fault isolation



Model reduction – Step by step validation

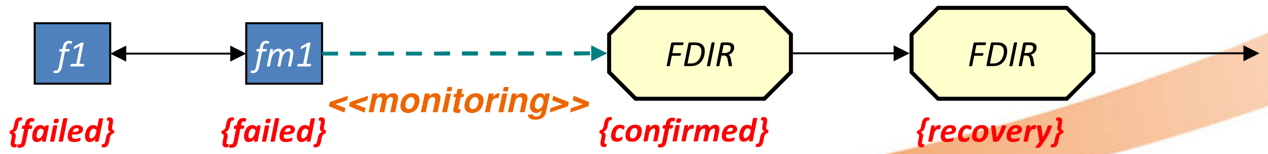


Find the earliest *F such that* / Find the earliest *t1 such that* is true at *t*



Slowest recovery time(t2) < Fastest propagation time(t1)

Find the latest *t2 such that*



To summarize…

- Systematic modelling method for dependability and FDIR
  - Compositional (dependability pattern)
  - Incremental (functional, dependability and FDIR layers)
  - Extensible (enhancement of deduced propagation paths)
  - Demonstrable (model simulation and model checking)
  - Complex propagation, common mode faults, external events, …
- Method improvements
  - Model reduction techniques, possibly resolution techniques
  - Complete libraries (fault models, failure modes, propagation rules)
- Integration into company's processes
  - Articulation of FDIR, Dependability, Engineering processes
  - Place of simulation, formal validation
  - Tools, methods benchmarking, training

Outline

- A few words about EADS, Astrium, …
- Dependability in space
  - Constraints, needs, solutions, achievements
- Dependability (FDIR) process
- Model Based Dependability
  - Engineering, Assessment
- Why it may become so complicated?

Hybrid models

- Structure / Behaviour
- Behaviour: Discrete logic + continuous
  - Decoupling possible but based on a priori hypotheses which must be validated, cannot be exhaustive, hardly systematic and anyway limited by the actual interactions
  - Interest of coupling, integrating models
    - Limitations of current tools and even theoretical framework to support comprehensive and accurate simulations… and even more proofs

Why we need at least time(s)

- Explicitly used by FDIR mechanisms (implicit coupling, enforcement of the desired organisation, sequencing, hierarchy)
- Sort of "pivot" notion between the discrete logics and the continuous physical phenomena
- Explicit incorporation of the temporal characteristics of failures
  - Transient, intermittent…
  - Possibly some other interesting though speculative ideas about time and failures

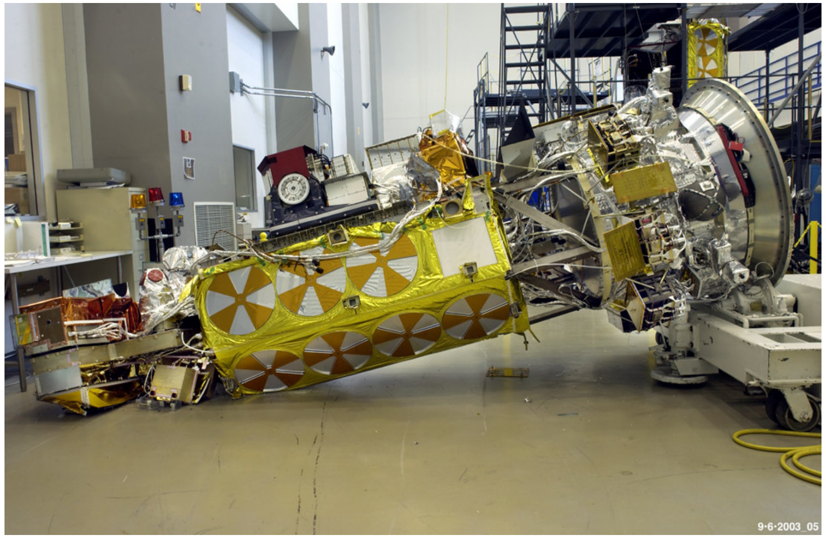
Time and failures

- Notion of resistance, during some time, to faults
  - Explicit property in security
  - Less usual though possibly interesting for accidental faults
  - See also the built-in resistance to exceptional environmental conditions
- Notion of "trajectory of failures"
  - "Mortal Byzantine", cf. Josef Widder, Martin Biely, Günther Griddling, Bettina Weiss, TU Wien, DSN 2007
- Notion of safety margin for dynamic systems
  - On-going PhD by Amina Mekki-Mokhtar (LAAS-CNRS, Toulouse, Supervised by D. Powell, J. Guiochet)

It is a long way to space

Factory,

Road…



No source of failure should be overlooked

ASTRIUM AN EADS COMPANY

Space Systems Dependability The hybrid (modelling) necessity

Jean-Paul Blanquart

ASTRIUM Satellites

Jean-paul.blanquart@astrium.eads.net

5th Latin-American Symposium on Dependable Computing

INPE, São José dos Campos, Brazil, April 25-29, 2011

All the space you need

ASTRIUM
AN EADS COMPANY