Formal Verification in Early Mission Planning

Philipp M. Fischer

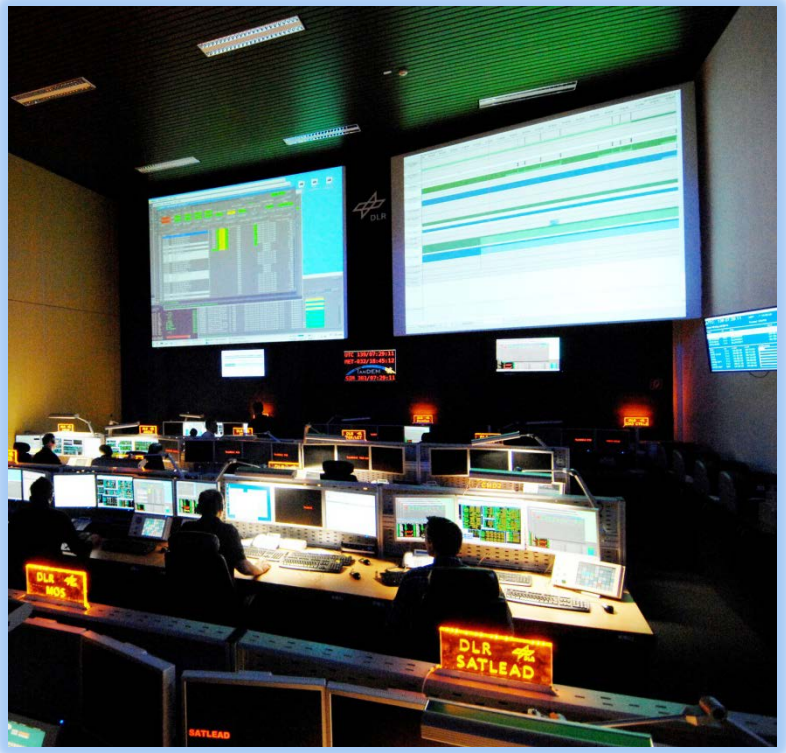
Simulation in European Space Programmes 2012

ESTEC - Nordwijk/Netherlands

DLR

Formal Verification in Early Mission Planning

- Planning a spacecraft and the importance of verification.
- From data model towards a verifiable presentation.
- Example study for early design verification.
- Performance and future evolution considerations.

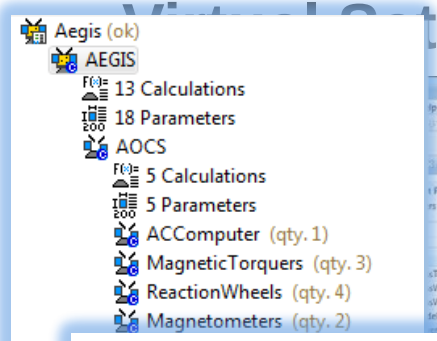


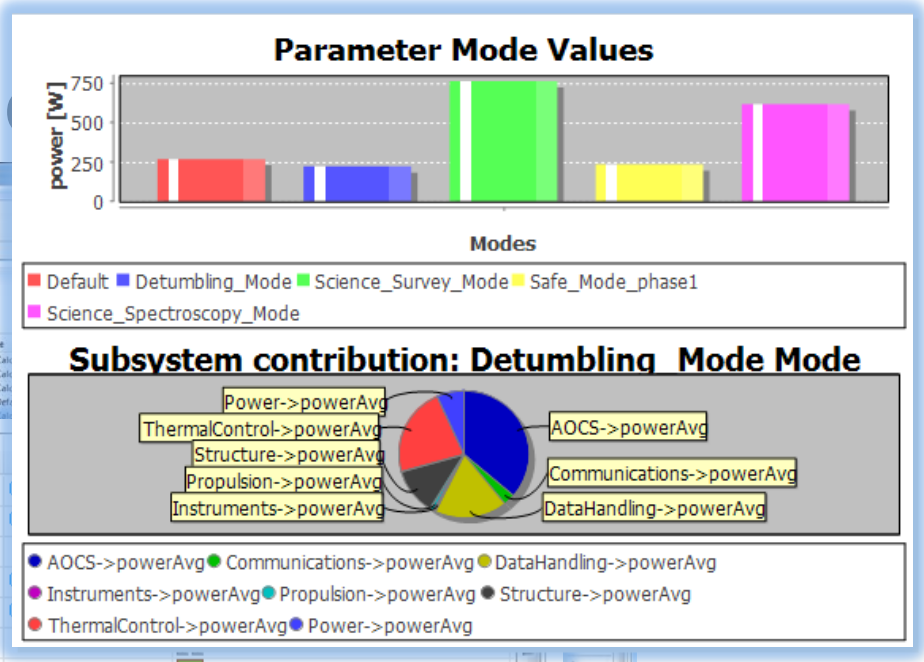
Concurrent Designing of a Spacecraft

- **Early design goals**
  - Estimate design of spacecraft
  - Evaluate feasibility of the mission
- **Supporting tools and processes**
  - Concurrent Engineering Facility
  - Studies of around two weeks
  - A common data source
- **Does spacecraft fulfill requirements?**

DLR

Virtual Satellite and the

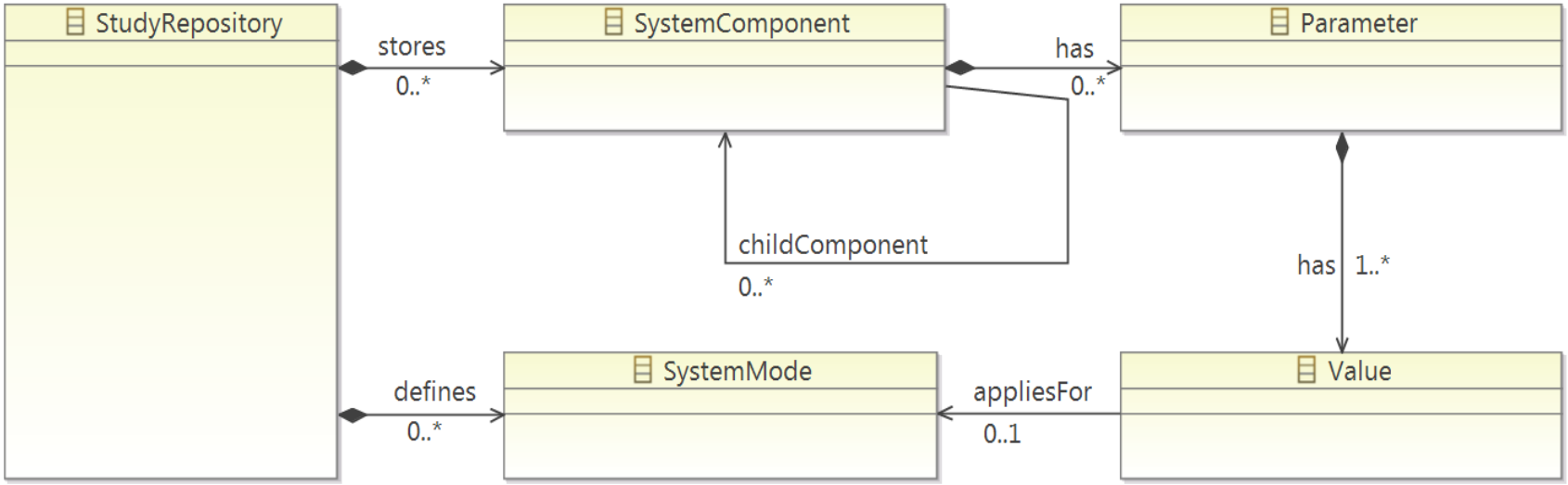




Name	Current Value	Unit		
massTotalDryMassWithSystemMargin	3171.40	kg		
massWetMassContrib	263.20	kg		
massWithMargin	2642.83	kg		
modeDuration	3600.00	s		
powerAvg	267.21	W		
Mode: Detumbling_Mode	221.20	W		Default Value
Mode: Safe_Mode_phase1	232.90	W		Default Value
Mode: Science_Spectroscopy_Mode	615.96	W		Default Value
Mode: Science_Survey_Mode	761.45	W		Default Value
powerAvgMinusMargin	160.32	W	yes	Calculated Value
powerAvgWithMargin	374.09	W	yes	Calculated Value
powerPeakPower	1165.78	W	yes	Calculated Value
powerPeakWithMargin	1632.10	W	yes	Calculated Value

Add Edit Remove

Storing the Data Model



Static Overview and Verification

	Mass w/o margin [kg]	Margin [%]	Margin [kg]	Mass with margin [kg]
Total dry mass:	2596.07			2642.83
System margin:		20.00		528.57
Total dry mass with system margin:				**3171.40**
Propellant:				263.20
Adapter mass:				
Launch Mass:				
Max launcher capacity:				
Buffer to launch mass:				

Name

Name: powerPeak

Value

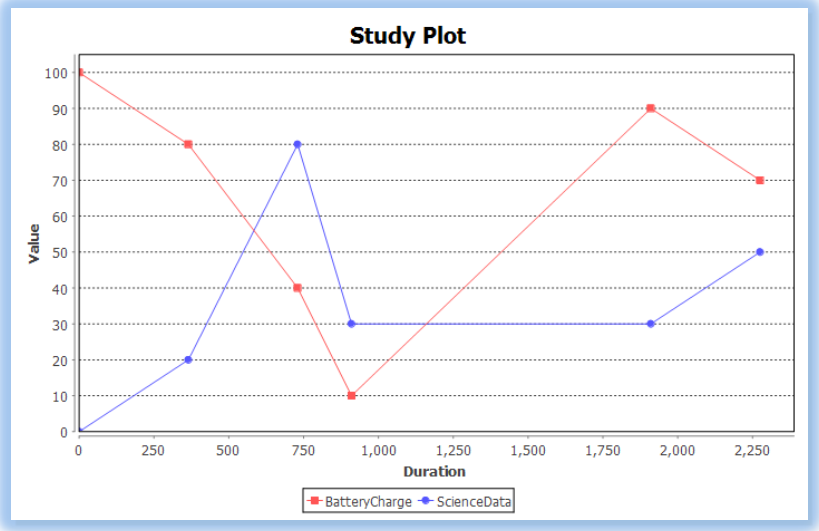
*Repository Manager | Problems

0 errors, 1 warning, 0 others

Description	Resource	Location	Type
Warnings (1 item)			
Parameter is leaving its bounds: powerPeak 200 < 510.0 < 420		UVSurveyorHUGE Parameter: powerPeak	Parameter Out of Bounds

Aspects Concerning Mission Time

- **What if we schedule modes**
  - Switching to Science
  - Switching to Charge
  - …
- **Can we fulfill the mission in time**
  - 200 GB scientific data
  - 5 years mission time
- **Many mode combinations**
  - Is there a way for verification?



The Idea: Modelchecking of State Machines

- **Creating a spacecraft model and its specification**
  - Transferring early design model to checkable representation
  - Transferring mission requirements to specification
- **This approach allows for**
  - Quick verifications of spacecraft changes
  - Quick verifications of requirement changes

An Overview to Model Checking

- **Verifying that a model complies to its specification!**
  - Applied to bus protocols
  - Common practice in semi conductor industry
  - Of the shelf tools: NuSMV, Spin, Prism
- **Special representation of the model**
  - State machines like Kripke Structures
  - Process interactions
- **Special representation of the specification**
  - Temporal Logic

Formal Verification in the Space Domain

- **Java Pathfinder Project by JPL**
  - Verification of Java Source Code
- **Code Command Sequence Verification by JPL**
  - Verifying safety of command sequences
- **Compass Project by ESA**
  - Pool of methods applied in parallel to an ESA mission

Planning Games Solved by Modelcheckers

- **Puzzles like the ferryman with the dog, goat and the cabbage**
  - How to cross the river without dog biting goat eating the cabbage?
  - Model of boat, dog, goat and cabbage
  - Specification: G (Safe -> !G (Safe & Goal))
- **Planning Competitions showed Satellite Scenarios**
  - Modelcheckers were compared to Planning tools
  - Comparable result and performance
- **Modelcheckers provide counter example to specification**

From Early Design towards a Checkable Model

- **Transformation of design model**
  - Mode dep. parameters >> Variables
    - *EnergyCharge*
    - *DataCollected*
    - *Time*
  - Operational Modes >> States
    - *Science*
    - *Downlink*
  - Mode dep. Values >> Change of Var.
    - *DataCollected +10 for Science*
    - *DataCollected -5 for Downlink*

```
MODULE main
VAR
    state: {SCIENCE1, SCIENCE2, CHARGE, DOWNLINK};
    time :   0 .. 1000;
    charge : 0 .. 100;
    ...
ASSIGN
    init(data)   := 0;
    ...
    init(state)  := SCIENCE1;

    next(state) := case
        TRUE : {SCIENCE1, SCIENCE2, CHARGE, DOWNLINK};
    esac;

    next(time) := case
        state = SCIENCE1 : time + 2;
        state = SCIENCE2 : time + 2;
        state = CHARGE   : time + 4;
        state = DOWNLINK : time + 1;
        TRUE             : time;
    esac;

    next(charge) := case
        ...
    esac;
```

What is a Safe Design

- **The satellite is safe if**
  - The battery is not overcharged
  - The battery is not depleted
  - The data storage is not filled

```
DEFINE
   safe :=   charge > 5   & charge < 100
        &    data >= 0    & data <= 100;
   goal := time > 50;
```

- **And it is safe if**
  - The propulsion tank is not empty
  - The solar panels are extracted after launch

- **These are mission constraints !**

What is the Mission Goal

- **The mission is successful if**
  - The satellite orbits for 4 years
  - Collects 200 GB Scientific Data

```
DEFINE
   safe :=   charge > 5   & charge < 100
        &    data >= 0    & data <= 100;
   goal := time > 50;
```

- **Further goals can be defined like**
  - Collecting 50 GB Scientific Data Type I
  - Collecting 60 GB Scientific Data Type II

- **These are mission goals!**

The Modelchecker plans our Mission

- **Using NuSMV Model Checker**
  - Use Kripke-like spacecraft model
  - Use Temporal logic:
    - *G (Safe -> G ! (Safe & Goal))*
  - Define provided Constraints
  - Define provided Goals

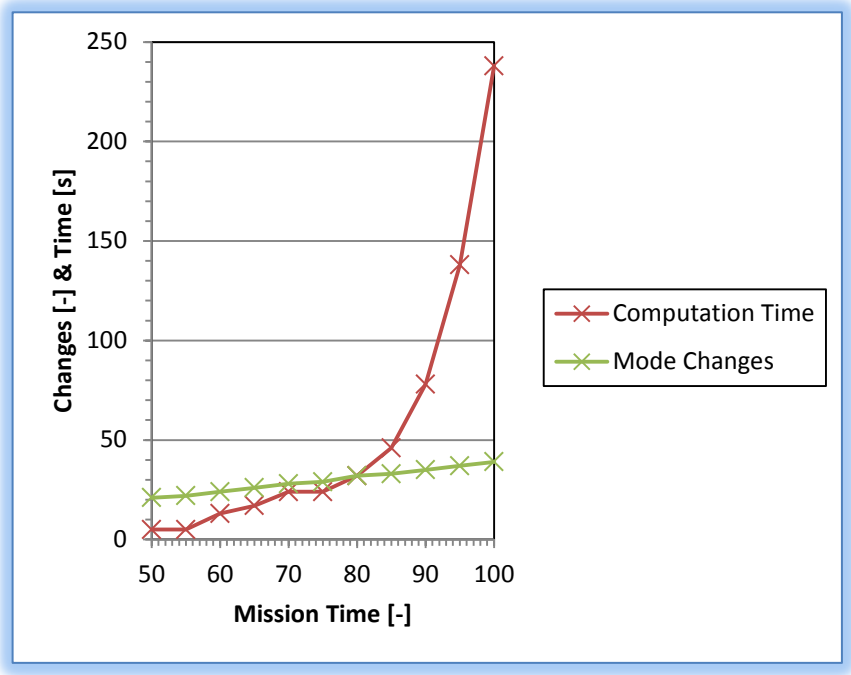
```
        state = SCIENCE2 : time + 2;
        state = CHARGE   : time + 4;
        state = DOWNLINK : time + 1;
        TRUE             : time;
    esac;

    next(charge) := case
        ...
    esac;

...
DEFINE
    safe := charge > 5 & charge < 100 & d
    goal := time > 50;

LTLSPEC
    G ((safe) -> G !(goal & safe))
```

```
charge = 30
  data = 65
-> State: 1.37 <-
  state = SCIENCE2
  time = 95
  charge = 80
  data = 30
-> State: 1.38 <-
  state = CHARGE
  time = 97
  charge = 45
  data = 80
-> State: 1.39 <-
  state = DOWNLINK
  time = 101
  charge = 95
  data = 45
  goal = TRUE
```

- **Counter example equals the mode schedule**

The End of the Universe – State Explosion Issue

- **States to evaluate**
  - Modes$^{\text{State-Changes}}$
  - Exponential Increase
- **We need quick iterations!**
  - Otherwise not useful for CEF



Focusing on the Detail or Lifetime

- **How much time evolves between state changes**
  - Seconds > fine detail useful for mission details
  - Mode duration > medium detail useful for reasonable overview
  - Orbit revs or Days > low detail useful for long term analysis

- **Difficult to chose right amount of detail**
  - State Explosion is still a major bottleneck

Level of Abstraction – The Orbit Example

- **Modeling time and data continuous aspects is difficult**
  - Needed for orbit position
  - Needed for ground contact

```
DEFINE
    orbit := time mod 10;
    downlink_constraint := (state = DOWNLINK) & orbit < 4;

    safe := downlink_constraint & ...
...
```

- **Using stochastic abstraction like**
  - 4 out of 10 orbits allow ground contact

It is Working and What Happens Next?

- **We have a quickly checkable model for CEF.**
- **We struggle with state explosion problem.**
  - Can handle it partly by certain abstractions.
- **We can hardly evaluate complex aspects.**
- **Future work and further directions**
  - Analyzing heuristic verification approaches
  - Coupling simulation models

Thank You and See You Later!

Philipp M. Fischer

German Aerospace Center (DLR)

Software for Space Systems and
Interactive Visualization

Philipp.Fischer@dlr.de