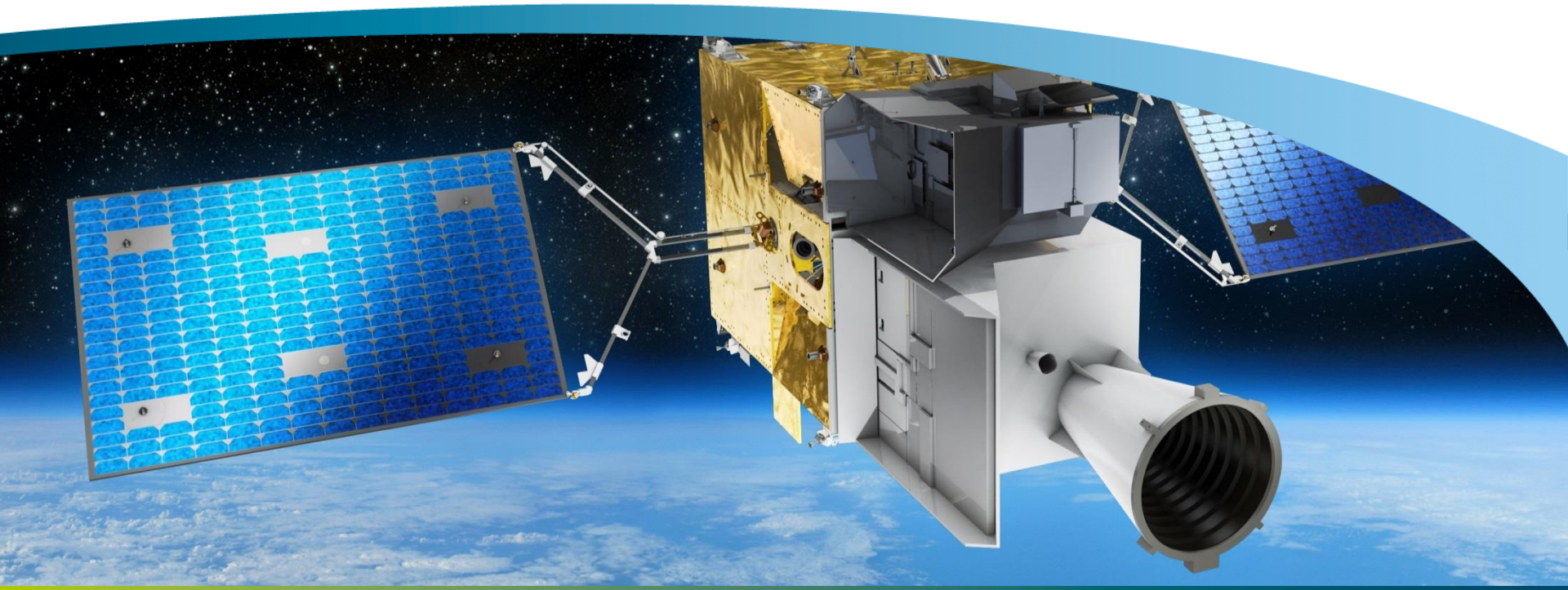
OHB System AG
Andreas Wortmann
19-21 June 2018, Lisbon, Portugal



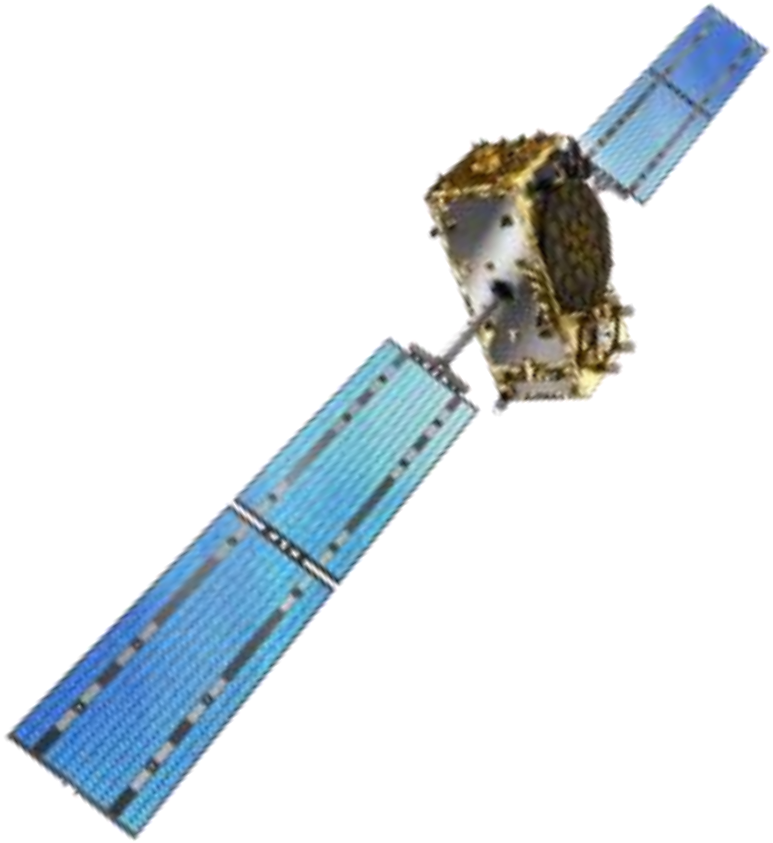
SPACE SYSTEMS

APPLYING FORMAL TIMING ANALYSIS TO SATELLITE SOFTWARE

We. Create. Space.

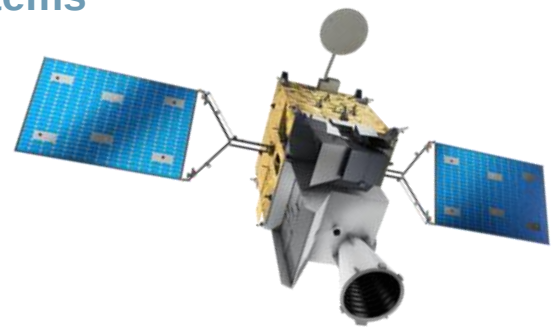
Agenda

- Introduction and Motivation
- Schedulability Analysis
- Architecture & Engaged Toolchain
- **Lessons Learned**
- Missions, Summary & Outlook



Introduction and Motivation – Satellite Software

- Central software item that glues together all subsystems
  - sensors, actuators, mechanisms, power …
- Control loops, including
  - Attitude and Orbit Control
  - Thermal Control
- Telecommand and Telemetry Interfaces
  - Full commandability and visibility
- Robustness
  - Failure Detection, Isolation and Recovery
  - Hardware redundancy
- Payload operation
  - Very large spectrum of software functions and performance needs



→ Large variety of functions with high reliability requirements

Introduction and Motivation – Mission Critical S/W

- Increasingly large projects
  - Complexity, Size, Cost, Time
- Testing software is very important but
  - Impossible to cover all cases
- Introduction of formal methods
  - Absence of runtime errors, Memory / Stack size analysis …
  - Timing Requirements

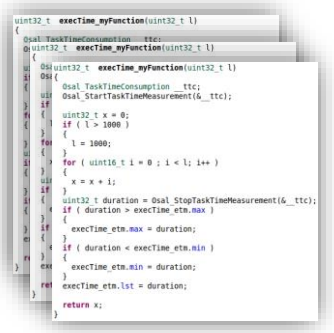
→ **Schedulability Analysis**

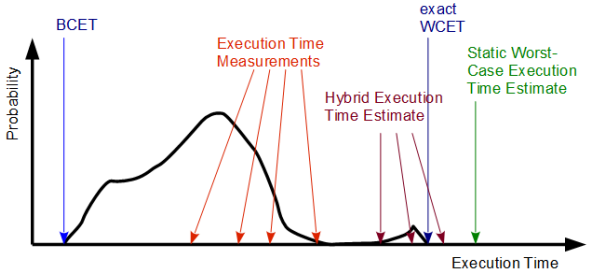
EUROPEAN COOPERATION ECSS FOR SPACE STANDARDIZATION

SA Required for flight software qualification by the [ECSS-E-40-1]

How to perform a Schedulability Analysis?

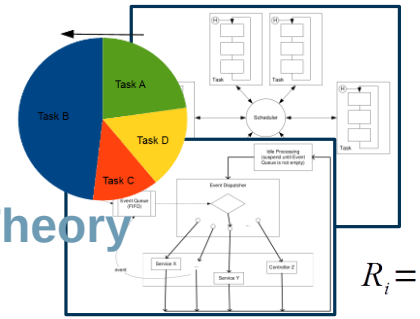
1) Timing analysis of code snippets

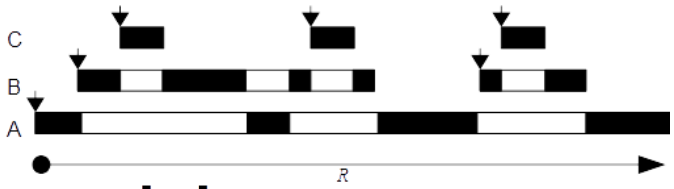




2) Schedulability calculation

- Tasking Model
- Mathematical Theory





$$R_i = CS + B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot (C_j + 2CS) + CI(R_i)$$

$$CI(\Delta t) = \sum_{r \in ISR} \left\lceil \frac{\Delta t}{T_r} \right\rceil \cdot (C_r + 2CIS)$$

3) Check with requirements

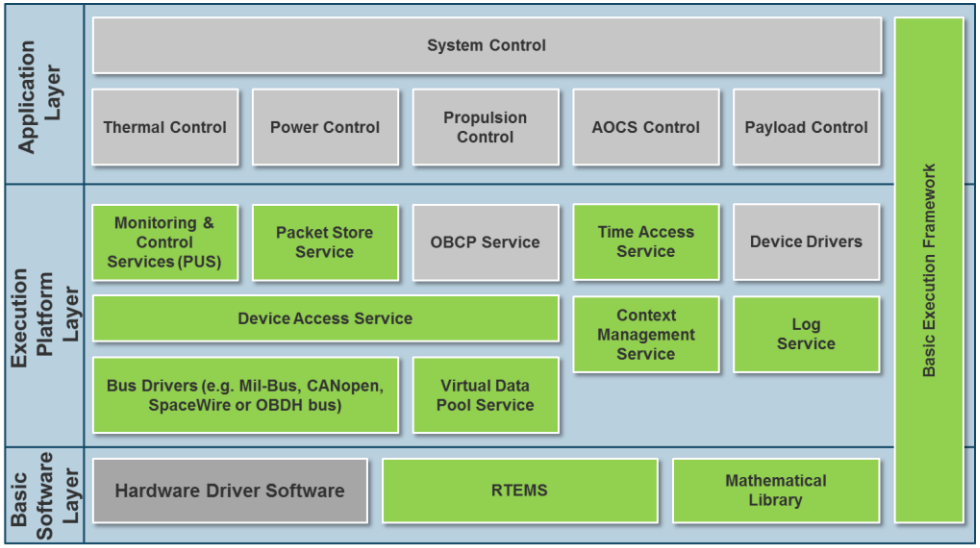
Req. AOCS-210:
Execution of the AOCS step function and its pre- and postprocessing shall be completed within 200ms after the 1Hz activation pulse.

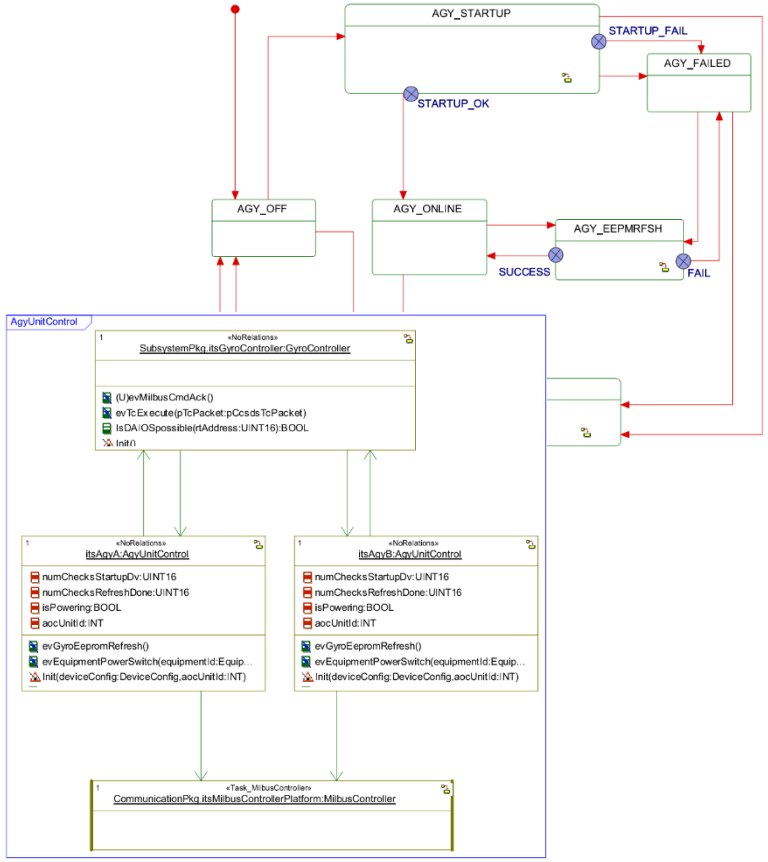
$$\forall i \in \tau : D_i \leq R_i$$

→ Analysis Model Complexity increases with Software/System Complexity

Static Architecture

Static layered set of interactive components





Model-Driven Approach /w design pattern and partial code generation from UML

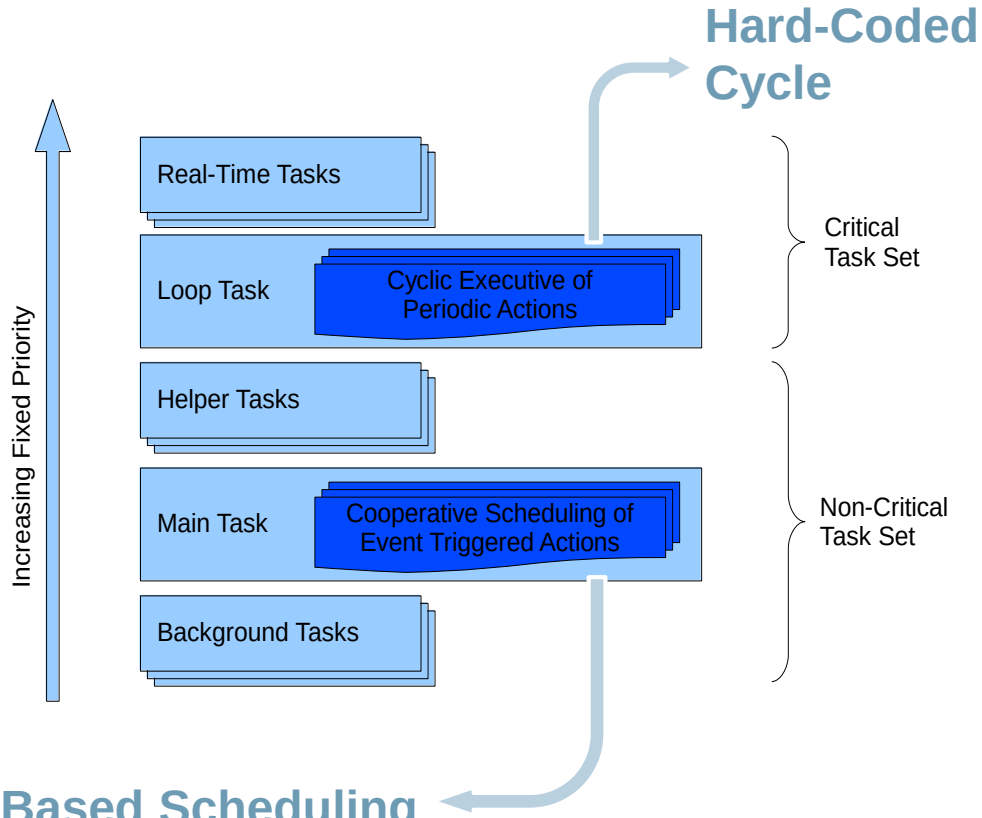
→ C language implementation

Real-Time Architecture

Single Core
Fixed Priority Preemptive Multitasking RTOS (RTEMS)
Immediate Priority Ceiling Protocol

Hard Real-Time
- Sporadic functionality
- Periodic functionality

Soft & Non Real-Time
- Majority of Code



Cooperative Non-Preemptive Event-Based Scheduling

Timing Analysis

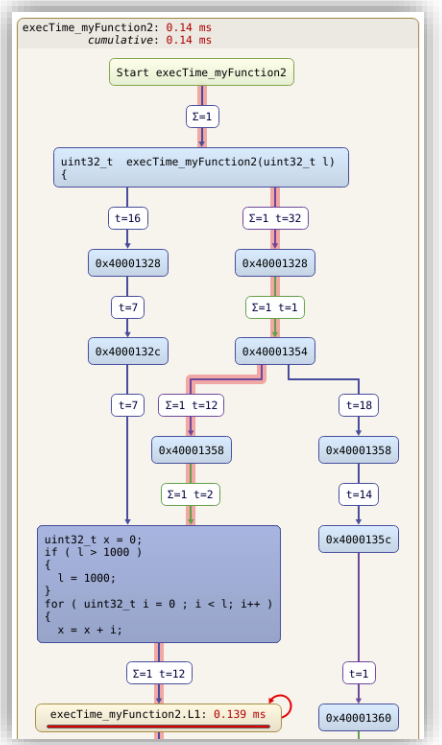
WCET Analysis of Code Snippets (subject to hard real-time constraints)

Alternatives:
Rapita Systems
Bound – T
…

- Static code analysis using AbsInt a³ suite
- Safe upper bounds for the WCET
- Manually added annotations

```
#compiler
compiler "sparc-gcc" ;

#clock rate
clock exactly 65.536 MHz;

#analysis precision
interproc flexible, max-length=inf, max-unroll=inf, defa

#includes
include "../../../ResponseTimeAnalyses/Annotations/DHSW.

# section properties
area section ".text" is read-only;
area "hkAccessTable" contains data and is read-only;

# memory accesses
global accesses restrict to 0x40000000 .. 0x5fffffff;

# function specific annotations
instruction "HandleThermostatAlert@static@TcsController.
area "itsHkRepositoryAccessService" + 0x024 bytes .. "it
loop "SwThermostat_SetMode" + 1 loop default-unroll = 6;
loop "SwThermostat_HasActiveSets" + 1 loop default-unrol
flow "EventReportingServicePUS05_SendErrorEvent" <= 1;
loop "GetMinMaxFloat" + 1 loop max 7;
snippet routine "log" is infeasible;
```

```
@staticWCET
exported uint32 myFunction2(uint32 l) {
  uint32 x = 0;
  if (l > 1000) {
    l = 1000;
  } if
  for (uint32 i = 0;i < l; i++ ) {
    x = x + i;
  } for
  return x;
} myFunction2 (function)
```

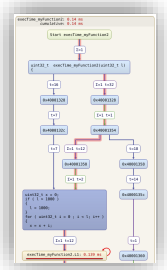


Schedulability Calculation and Check

Alternatives:
MAST, Cheddar
Symtavision
…

Response Time Calculation

- For simple tasking models: MS Excel and VB scripts
- For more elaborate models: In-house DSL

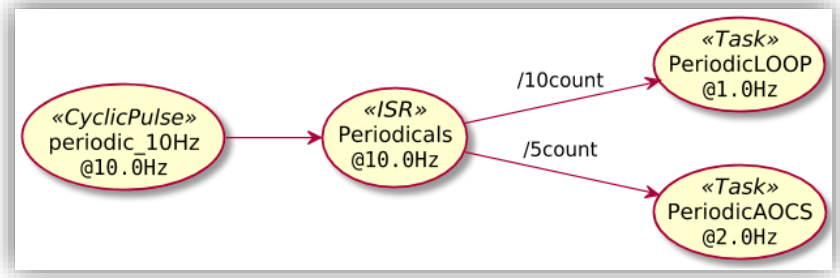


Determined WCET

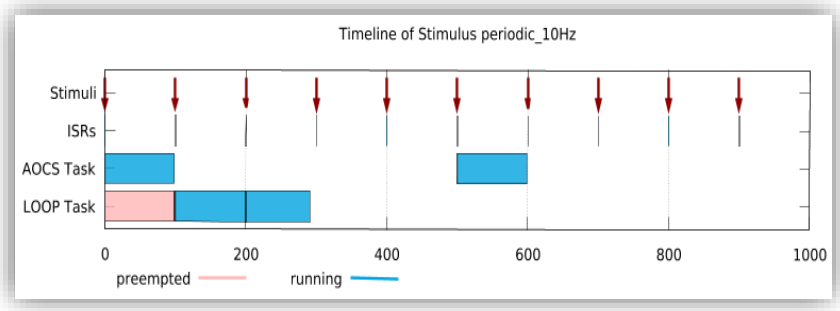
Timing Requirements

Req. AOCS-210:
Execution of the AOCS step function and its pre- and postprocessing shall be completed within 200ms after the 1Hz activation pulse.

```
Tasking Model : SCSW
-------------------------------------------------------------------------------
Analysis:
RTA1 : Response Time Analysis (Immediate Ceiling Priority Protocol)
            Context Switch Penalty 0.0001 ms

-------------------------------------------------------------------------------
// Protect shared memory from concurrent access.
Monitor DataReportingServicePUS03  -> ceiling Priority: 60 (AOCS)

// External cyclic stimulus from timer
Cyclic Stimulus periodic_10Hz T= 100 ms -> 10.0 Hz

// ISR is triggered by the timer and divides frequency for releasing periodic tasks
ISR Periodicals Stimulus: periodic_10Hz                          -> 10.0 Hz
                Offset  = 0 ms
                WCET    = 0.158 ms
                Releases Job:
                PeriodicLOOP with period of 10 counts and an offset of 0 counts
                PeriodicAOCS with period of 5 counts and an offset of 0 counts

// Periodic AOCS step function
Task AOCS Priority: 60 -> 2.0 Hz

// Execute AOCS step function
Job PeriodicAOCS Task     : AOCS                                 -> 2.0 Hz
                 Priority = 60
                 WCET     = 98.68 ms
                 Monitors used:
                 DataReportingServicePUS03 : wcet= 5.322 ms
                 Deadlines:
                 internal : WCET of 62.245 ms until deadline of 70 ms
                 end : WCET of 98.68 ms until deadline of 500 ms

// Main loop of the software for reporting and checks
Task LOOP Priority: 50 -> 1.0 Hz


// Execute reporting and checks
Job PeriodicLOOP Task     : LOOP                                 -> 1.0 Hz
                 Priority = 50
                 WCET     = 192.0 ms
                 Monitors used:
                 TmPacketPool : wcet= 0.016 ms
                 Deadlines:
                 end : WCET of 192.0 ms until deadline of 1000 ms
```



Calculation, Visualization and Documentation



Experiences and Lessons Learned

- Software Development Lifecycle
- Software Requirements
- Software Architecture
- Software Coding / Analysis Tools

Experiences and Lessons Learned
-- Software Development Lifecycle

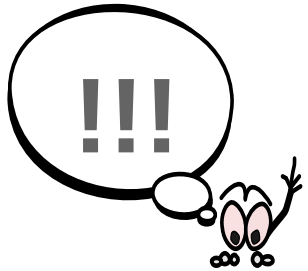
“ Here is my software,

it’s ready,

it’s tested,

and we can’t change it anymore,

→ please prepare the timing analysis report until tomorrow EOB !”



Ok, that’s a bit exaggerated, but:

It’s about the mindset:
people are focused on function rather than on timing

Experiences and Lessons Learned -- Software Requirements

Non-functional (timing) requirements are essential.

→ **You have to know your needs early to make the correct design decisions**
→ **You have to be able to verify/validate/test your system in the end**

From experience:

- **Timing requirements are usually not of high quality and incomplete.**
  - **Requirement documents**
  - **User manual and ICDs of used physical devices**

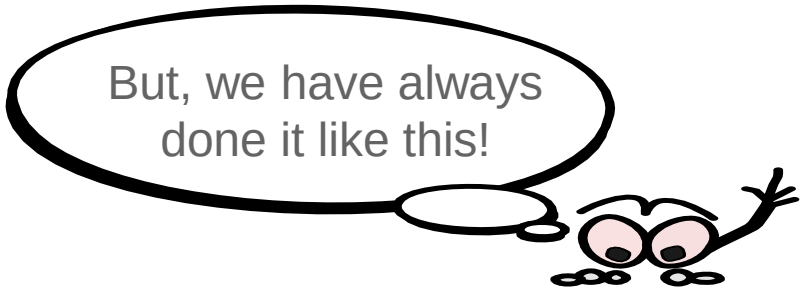
→ A complete set of high quality formal timing requirements is essential

Experiences and Lessons Learned
-- Software Architecture

Software Architecture ←→ Computational Model, Mathematical Theory (Response Time Calculation)

Some common design/implementation pattern are in conflict with assumptions or efficient application of the theory !

- **Great source for discussions**
- **Know the assumptions**
  → And show that they are obeyed !
- **Design for Analyzability**



→ Design the architecture according to timing needs and keep it simple !

Experiences and Lessons Learned
-- Software Architecture

Software Modularity

Common Pattern:

- Define modules according to functionality
- Modules are independent from each other
- Support module reusability

→ Each module defines a set of tasks

Efficient Application of Theory:

- Keep task interaction scenarios simple

→ Limit the set of tasks

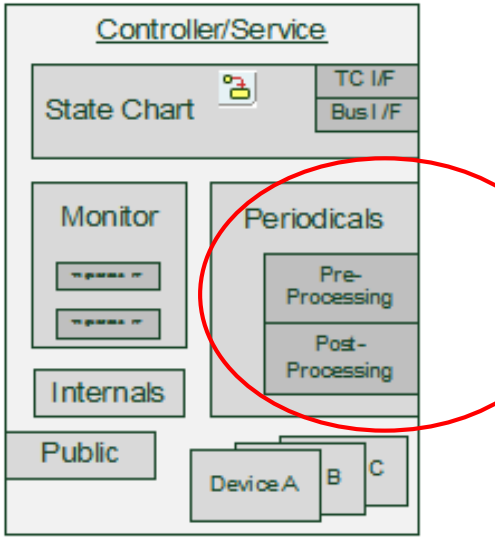
Reduce number of tasks and their communication

Experiences and Lessons Learned -- Software Architecture

Software Modularity

- When first establishing Timing Analysis, we
  - Reduced the number of tasks from 50 to 10 (5 being subject to hard constraints)
  - Reduced the monitors (to protect critical sections) from 50 to 15
- Functionality is assigned to tasks according their timing needs

Temporal Characteristics		
	Aperiodic	**Periodic**
Hard	TC reception, …	AOCS & Thermal control loops, …
Soft	TM generation, …	Monitoring of parameter values, …



- Software Modularity (for reuse)
  - Passive modules only according to pattern
  - Specific tasks that invoke operations

Experiences and Lessons Learned
-- Software Architecture

Self-Suspending Tasks

Common Pattern:

- Tasks suspend during resource accesses
- Suspensions may be hidden to provide a “user-friendly” API (Libraries, Legacy Code, Operating System …)
- Suspensions to implement temporal sequences (wait statement)

→ Invisible delays and additional (hidden) ISR invocations
→ Unknown (but hopefully bounded) delays

Assumption of Theory:

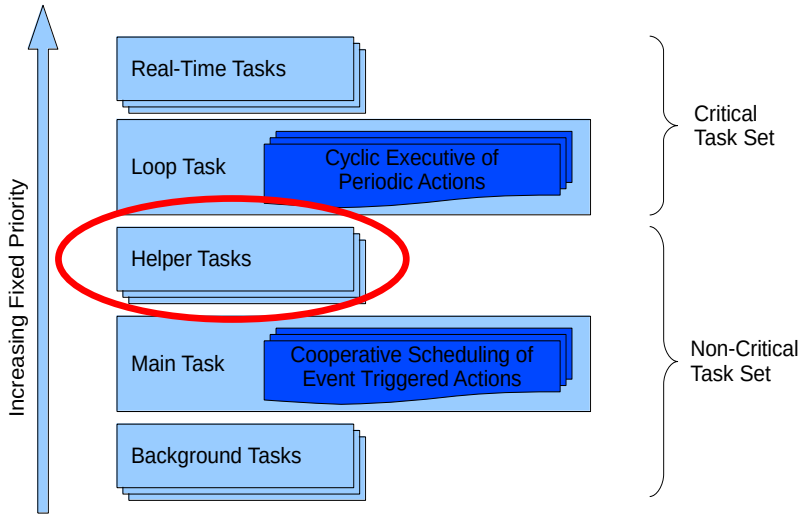
- Each task has a single point of suspension

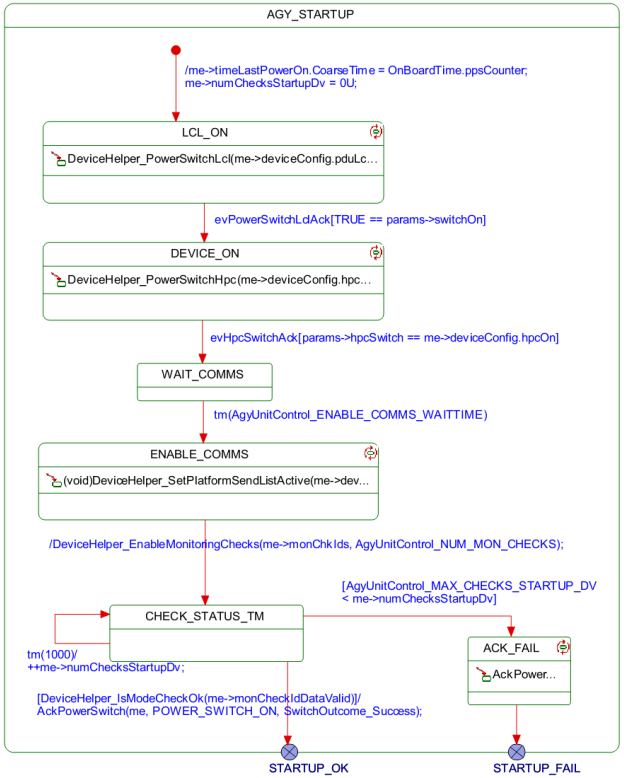
Eliminate task self-suspensions in tasks subject to hard real-time constraints.

Experiences and Lessons Learned -- Software Architecture

Self-Suspending Tasks

- Introduction of “Helper Tasks” executing self-suspending
  - I/O accesses
  - Low-Level Driver API calls
  - (+ Globally account for “hidden” ISRs)
- Temporal sequences are implemented in event-driven state machines



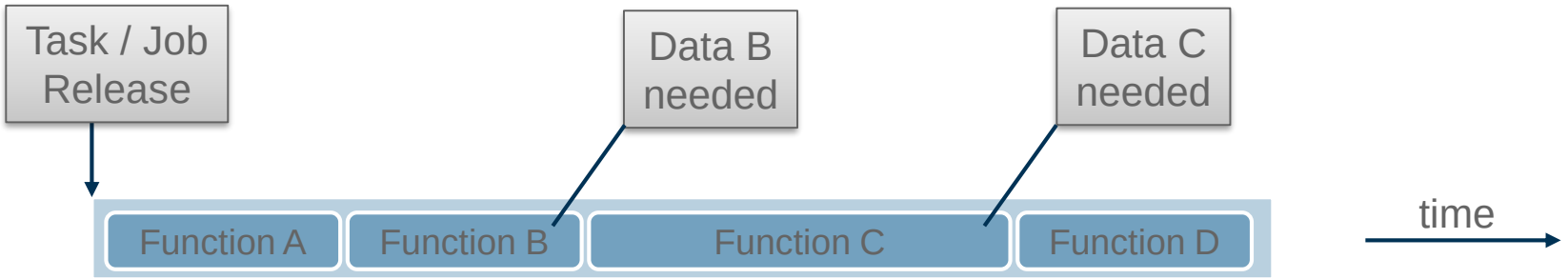


Experiences and Lessons Learned -- Software Architecture

Data flow constraints

Common Pattern:

- A single task prepares data items for different data sinks
- Some data is required before the task completes



Theory:

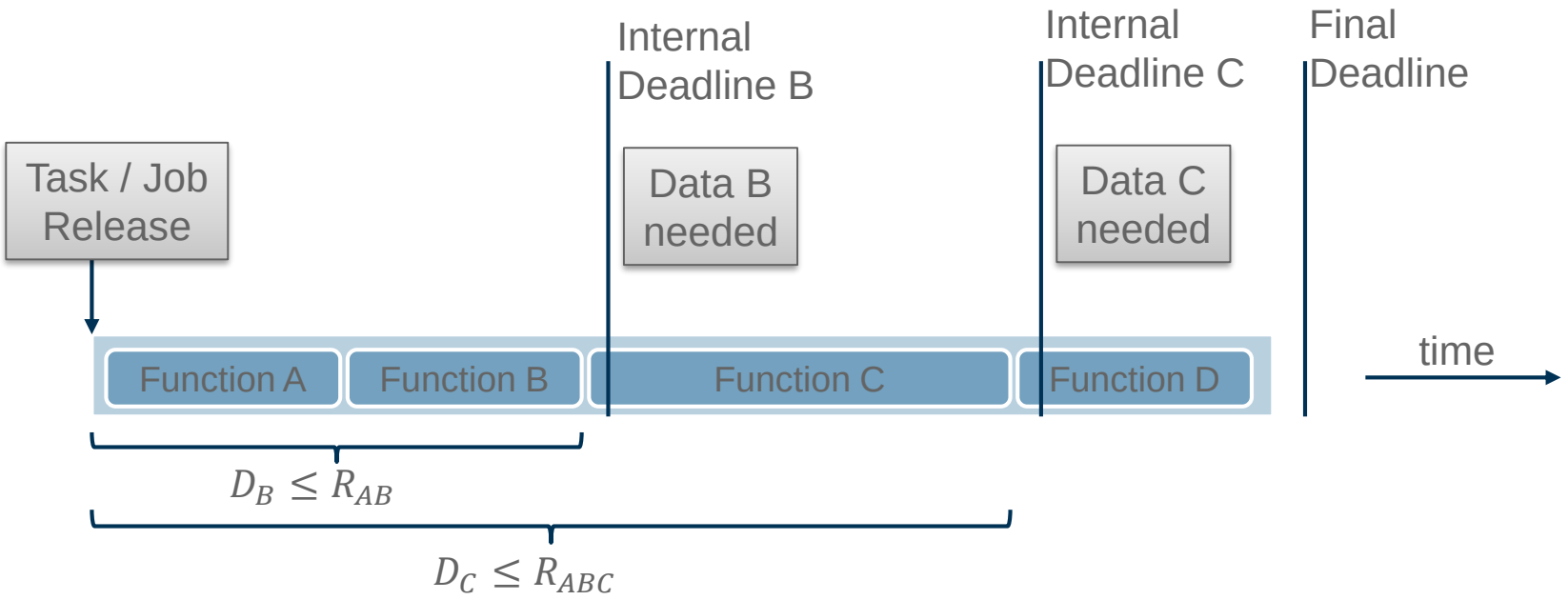
- Doesn't specifically address this need

Augment the analysis model with "virtual" tasks and "internal" deadlines.

Experiences and Lessons Learned -- Software Architecture

Data flow constraints

- Define Internal Deadlines



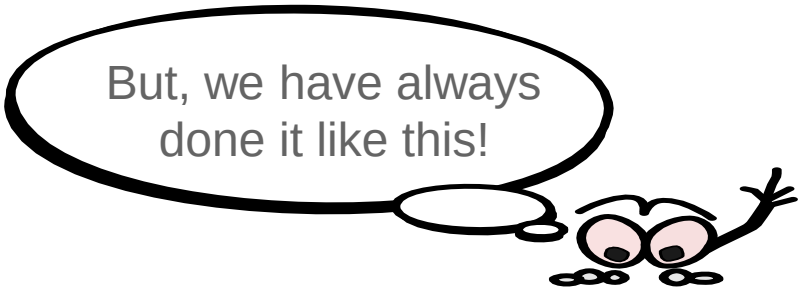
- Calculate response times of task until respective data is prepared
- And compare with respective deadline

Experiences and Lessons Learned -- Coding Rules

Coding Rules ←→ (static) WCET Analysis

Mostly WCET Analysis Tool Specific
… details not addressed in presentation

- Great source for discussions
- Know the peculiarities of the WCET tool

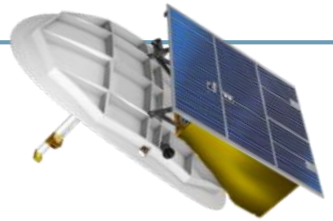


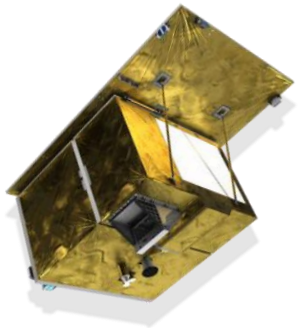
Mind-set

- Algorithms: Worst Case Time (WCET) vs Average Time (AET) → loops!
- Avoid dynamic structures and data dependent control flow

→ Code for efficient WCET analyzability!

Missions

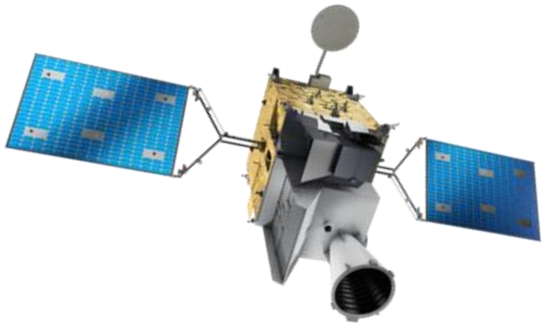


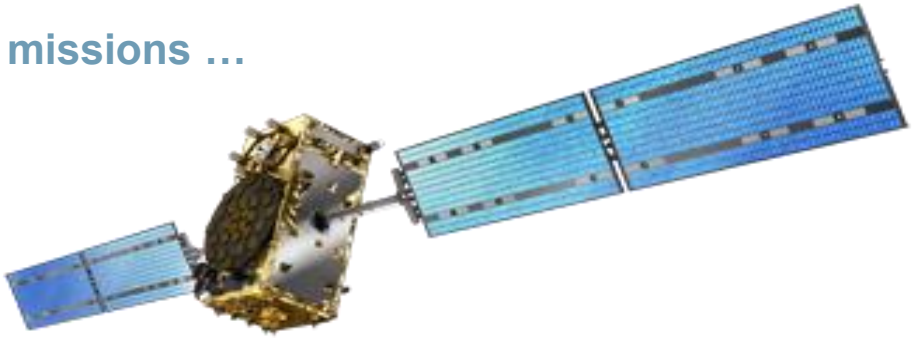


The presented analysis has been and is carried out successfully in multiple missions

- **Small Geo Platform**
  - **Hispasat AG1 Telecom Satellite**
  - **European Data Relay Satellite – C**
- **Galileo**
- **Meteosat 3rd Generation**

- **… Method will be applied to all future missions …**





→ Performing SA and WCET analysis in satellite flight software is established

Evaluation and Summary

Is it worth doing Formal Timing Analysis ?

→ **YES**

-- We have to do it as it is part of the Flight Software Qualification
-- We can do it as the technology is available

Is it worth spending all the effort and money ?

→ **Well**, we'll have to become way more efficient

-- and there are many options to improve in

Experiences

- Engineers (Requirements and Software) are not used to consider timing essential

  → Intensify training, Increase awareness

- Modern WCET analysis tools are complex to use correctly

  → Ease utilization by restricting the programming language (MBSE, pattern …)

- Tools and analysis are not (yet) well integrated in the workflow

  → We want a “Push-Button Solution”

- SA and static WCET analysis provides orthogonal view on the software

  → Nice Side-effect … helps us finding bugs early

Outlook – future processor hardware

Right now:

Leon2 single core @ 65Mhz under maximum load conditions
30% CPU utilization

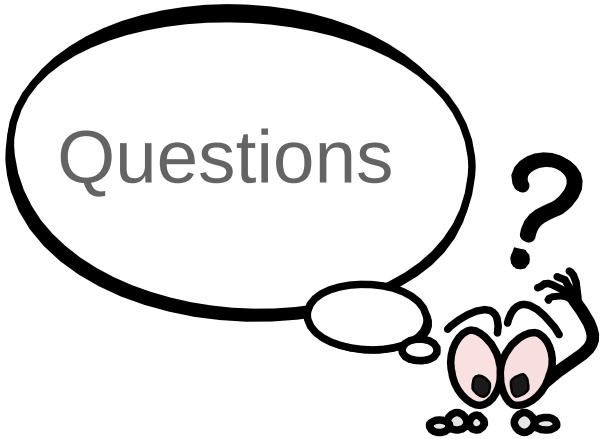
If more processing power is needed:

- Distributed Systems
  - Increasing Interest by Systems Engineering
- Multi-Core Processors
  - First hardware is on the (Space) Market

→ Flight Software Engineering should get ready to address these upcoming challenges

→ Need input from academia and tool vendors

OHB System AG
Andreas Wortmann
18-22 June 2018, Lisbon, Portugal



SPACE SYSTEMS

APPLYING FORMAL TIMING ANALYSIS TO SATELLITE SOFTWARE

We. Create. Space.