Ensuring Schedulability of Spacecraft Flight Software

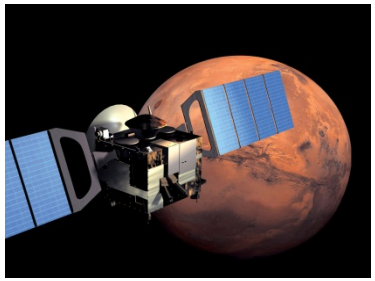
Flight Software Workshop

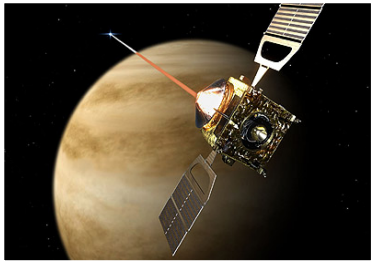
7-9 November 2012

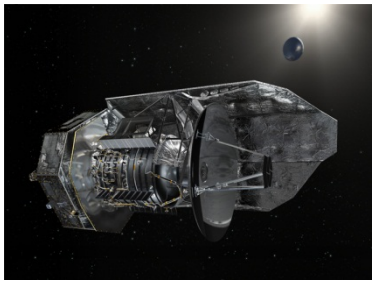
Marek Prochazka & Jorge Lopez Trescastro
European Space Agency

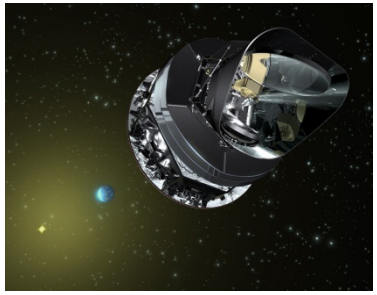
OUTLINE

- **Introduction**
- **Current approach to ensure FSW schedulability**
  - Schedulability analysis approach
  - Worst-case execution time measurements
  - Tools
- **Challenges of the current approach**
  - Analysis techniques
  - Sources of pessimism
  - Hardware
  - …
- **Challenges coming in the near future**
  - Cache
  - Multi-core
  - Integrated Modular Avionics
  - Model-Driven Software Engineering
- **Conclusions**









REAL-TIME SOFTWARE & DYNAMIC ASPECTS

- **Real-time software: Correctness is partially a function of time**
  - Failure to respond is as bad as a wrong response
- **Timing requirements**
  - High-level timing constraints come from system requirements (e.g. GNC), HW/SW interaction analysis, system needs
  - E.g. delay between reading a sensor and updating actuator
- **Schedulability concerns design and implementation**
  - System requirements are translated to tasks and their timing properties (deadlines, offsets, jitters, latencies)
  - Design constraints could come from requirements
  - E.g. “statically assigned priority scheme”
- **Verification through analysis & testing**

HARD VS. SOFT REAL-TIME

Hard real-time

- Tasks have “hard” deadline, which must be met at all times
  - Time/value function gives 1 before deadline and 0 after deadline
- This does not imply that missing a deadline has “catastrophic” consequences (that is determined by task/system criticality)

Soft real-time

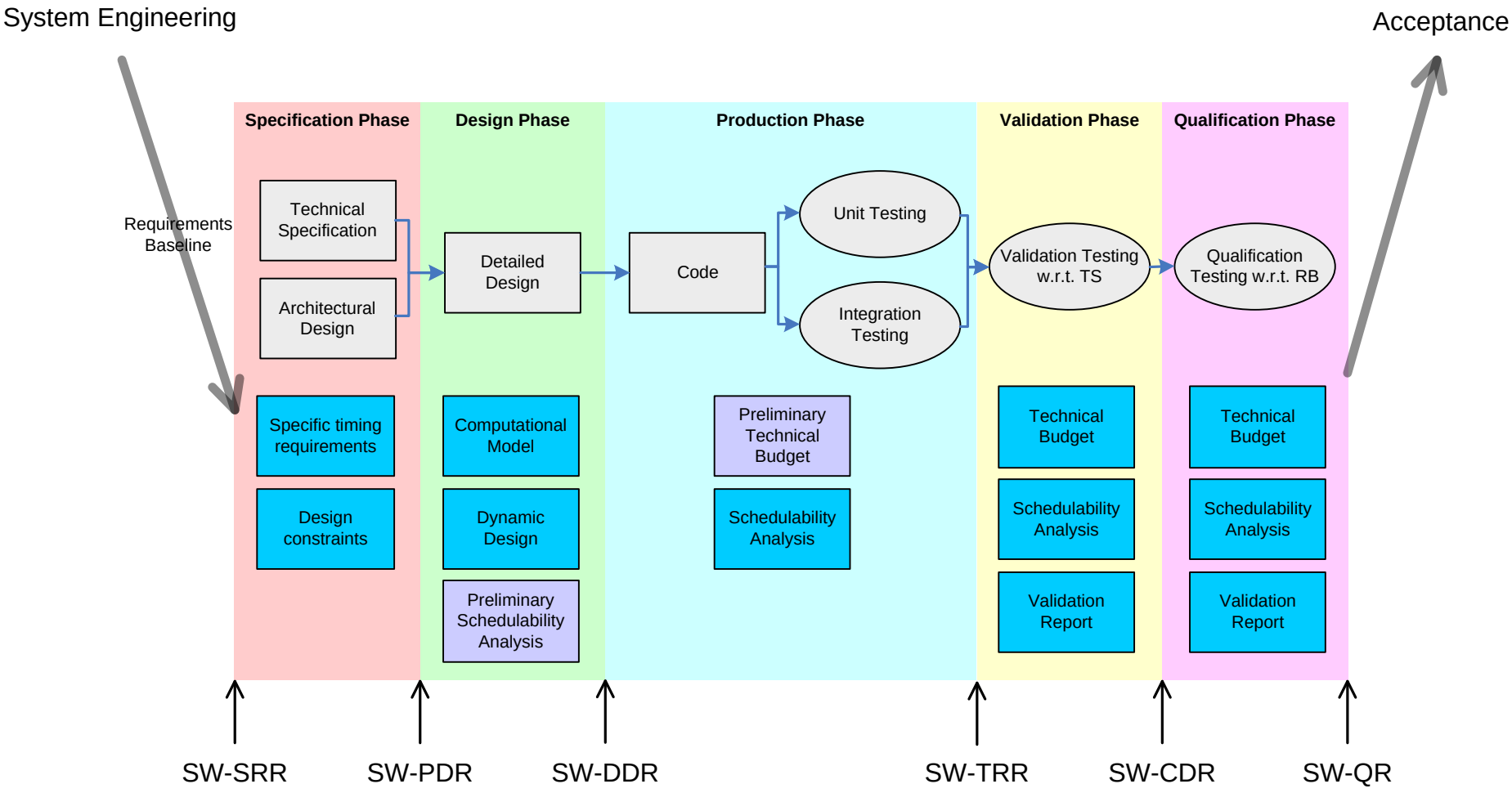
- Approximate deadline
- Utility of answer degrades with time difference from deadline (time/value function is a curve) or number of missed deadlines
- Stochastic methods, probability of meeting a deadline
- “Hard real-time is hard, but soft real-time is harder”
- But having no deadline at all makes things easier indeed

MIXING HARD AND SOFT REAL-TIME

- **Mixing tasks with hard deadlines with tasks with soft deadlines in a single system is possible**
  - Error-prone due to shared resources
    - Temporal, spatial and I/O partitioning could be needed
  - Analysis of such system could become more complex

- **We consider FSW a hard real-time system**
  - We do not analyse consequence of overruns, instead we make all possible to avoid them (by analysis)
  - In some cases missing a deadline would not cause a serious problem
    - I.e. not all deadlines should be defined as “hard”
    - Robustness analysis needed
  - Recently we see some systems with tasks having no deadlines, but analysis not always fully adjusted

SCHEDULABILITY THROUGHOUT THE SOFTWARE LIFECYCLE



DYNAMIC DESIGN & COMPUTATIONAL MODEL

- **All system tasks**
  - Their activation signal (external signal/event, processor clock)
  - Restrictions on what they can and cannot do
- **Task communication means** (shared memory, mailbox/message queues, signals, rendez-vous)
- **Handled and non-handled interrupts** (frequency, priority, resources)
- **Scheduling type** (cyclic executive, fixed-priority preemptive)
- **Task synchronisation mechanisms** (including mutual exclusion)
- **Resource protection mechanisms**
- **Inter-node communication and distribution**

- **Weak computational model ➔ complicated schedulability analysis**

- **Design not detailed enough ➔ complicated schedulability analysis**

SCHEDULABILITY ANALYSIS

- **Analysis method**
  - Rate-Monotonic Analysis (with blocking)
  - Response-time analysis
  - Proprietary (offsets, adjusted multi-frame model)
- **Task table with all details**
  - Periodic/sporadic, WCET, period, priority, blocking time, deadline
- **All shared resources**
  - Semaphores, usage, critical sections, priority inheritance
- **Interrupt handlers**
- **System overheads**
  - Preemption, interrupt latency, access to semaphores, interrupt locks, message queues, etc.
- **Use of operational scenarios** (i.e. realistic TM/TC traffic, operations)

SCHEDULABILITY ANALYSIS

Task Id (Name)	Subsystem	Priority	Type	Frequency	Period	Deadline	WCET	Max Blocking	Max Interferen	Utilisation	Response Time	Deadline Margin
				Hz	ms	ms	ms	ms	ms	%	ms	%
Time Management	System Control	10	Periodic	100	10	10	0.44	0.090	0.000	4.40%	0.53	94.70%
1553 Bus Handler	BSW	20	Periodic	50	20	20	4.70	1.120	0.440	2.66%	6.26	68.70%
AOCS Main Loop	AOCS	25	Periodic	10	100	40	9.70	0.260	27.900	9.70%	37.86	5.35%
TC Handler	Data Handling	61	Sporadic	5	200	100	6.22	1.120	47.300	3.11%	54.64	45.36%
Thermal Control	System Control	66	Periodic	5	200	200	11.25	0.850	53.520	5.63%	65.62	67.19%
OBCP	Data Handling	67	Periodic	2	500	200	68.40	0.850	78.020	13.68%	147.27	26.37%
Housekeeping	Data Handling	68	Periodic	1	1000	400	40.40	0.850	31.820	4.04%	73.07	81.73%
MTL	Data Handling	91	Sporadic	1	1000	400	24.30	0.160	271.120	2.43%	295.58	26.11%
System Log	Data Handling	97	Sporadic	1	1000	1000	126.90	0.160	569.020	12.69%	696.08	30.39%
Scrubbing	System Control	98	Periodic	1	1000	1000	114.00	0.000	744.000	11.40%	858.00	14.20%
Total										**69.73%**		

CURRENT CHALENGES

- **Multiple sources of pessimism in schedulability analysis**
  - Requirements (e.g. telecommand upload rate)
  - Architectural and detailed design (lack of detail)
    - All tasks are considered hard real-time
  - Analysis method (for preemptive systems)
    - Combining all worst-case scenarios in a single scenario with probability to occur close or equal to zero
    - Example: Period of sporadic task is set as minimum of its inter-arrival time
    - Example: Blocking time on a set of semaphores assumes that they are all acquired
      - Not if priority ceiling is used
      - Not if more subtle analysis of resource usage is performed
- **Typical requirement is 25% margin on timing**
  - However we often see 20-25% nominal CPU load on satellites

FURTHER CHALLENGES

- **Observation**
  - Preemptive scheduling allows you not to concern about timing when designing your fligh software
  - This is not possible with cyclic scheduling (where timing is defined for your task from the beginning)
  - This is fine, as long as you end up passing the schedulability test … but what happens if you don't ?

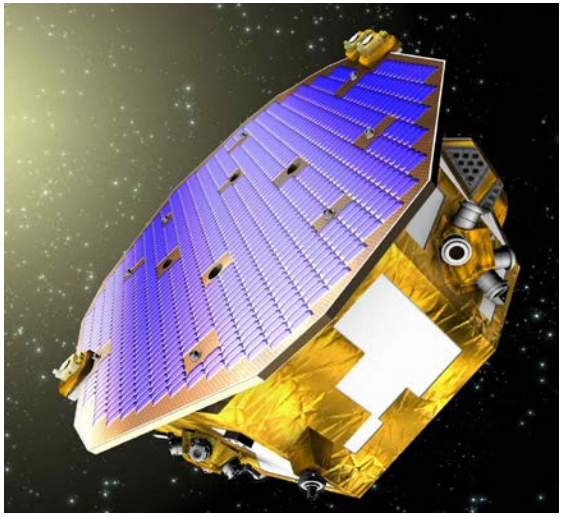
- **Hardware not so fast comparing to other domains**
  - ERC32 (SPARC v7) clocked 20 MHz
  - LEON2 (SPARC v8) clocked 80-120 MHz
  - LEON2 and LEON3 with cache

- **Tools for schedulability analysis and WCET measurements**
  - No unified set of tools for different ESA projects
  - Often spreadsheet based approach

CHALLENGES IN THE NEAR FUTURE

CACHE

- **Cache is good to speed-up in average, at the cost of more variable execution time**
  - Less predictability
  - Difficult to analyse
  - Aggravated worst-case execution path at the level if microinstructions (not in LEON2/SPARC-V8 RISC architecture)
- **Impact of architecture & design on the use of cache**
  - Small changes to memory map could have impact on cache miss ratio and consequently on timing
- **Possible solutions**
  - Cache-aware schedulability
  - Cache locking
  - Cache partitioning
  - Cache-aware coding style
  - Cache-aware linker



MULTI-CORE PROCESSORS

- **Multi-cores offer better performance per watt than single-core processors**
  - Expected technology trend also in time-critical systems
- **Classical approach is not efficient for multi-core (WCET per task, fixed priorities)**
  - Multiple tasks execute at the same time (one per core)
  - WCET harder to analyse due to inter-task interferences accessing shared resources
    - Arbitration mechanism
    - WCET depends on workload!
  - For two or more processors, no deadline scheduling algorithm can be optimal without complete a priori knowledge of deadlines, computation times and process start times
- **Dynamic priority scheduling theory is regarded as having potential advantages**
  - Higher CPU utilisation
  - Separation between truly hard real-time tasks (missing a deadline is not acceptable) and soft real-time tasks

INTEGRATED MODULAR AVIONICS

- **Logical partitions with strong spatial and temporal isolation**
- **Inter-partition Communication (IPC) mechanism respects space partitioning and real-time determinism**
  - Static scheduling of communication partition
- **Reduced integration effort**
  - Modular verification
- **Co-hosting applications with different criticality levels**
- **Partitioned design is a good way to migrate to a multi-core system**
  - Task parallelism better suited than data parallelism for data processing on multicore
- **Scheduling policy could be chosen per core**
  - Fixed-priority preemptive scheduling for hard-real-time
  - Dynamic scheduling for soft-real-time and event-driven tasks
    - Example: IRQ handlers execution, as I/O is mostly non-deterministic
    - Example: FDIR mechanisms
  - E.g. 3 cores using fixed cyclic scheduling, 1 core using dynamic scheduling

MODEL-DRIVEN SOFTWARE ENGINEERING

- **Schedulability analysis should be embedded in the model-based development**
- **New programming model: Non-functional properties used to define timing and concurrency control**
  - Non-functional properties related to timing (control flow, timing, deadlines, communication budgets, etc.) and concurrency (reentrancy) must become an integral part of a software component description
- **ASSERT / TASTE**
  - TASTE is a set of tools dedicated to the development of embedded, real-time systems, developed by ESA
  - Allows to easily integrate heterogeneous pieces of code produced either manually or automatically by external modeling tools
  - Provide facilities for automatic schedulability analysis (connection with CHEDDAR schedulability analysis tool)
- **Challenges**
  - No experience from real projects
  - Programming model is restricted by a tool
  - Difficult to apply legacy systems

CONCLUSIONS

- **ESA approach to schedulability analysis**
  - Overview of FSW development lifecycle
  - Current challenges
- **(Near) future challenges**
  - Cache, multi-core, IMA
  - Model-based FSW engineering
- **Scheduling on multi-cores is still a research topic**
  - IMA could make it easier to control
- **Model-driven engineering**
  - From schedulability point of view still an open issue

- **ESA is trying to unify the current approach as well as look into the future**
  - Research studies
    - WCET for LEON with cache
    - Schedulability for Integrated Modular Avionics
    - Schedulability for multi-core processors
    - Model-based engineering

THANK YOU

Marek.Prochazka@esa.int

Jorge.Lopez.Trescastro@esa.int

European Space Agency