Conference on Reliable Software Technologies

Guaranteeing Timing Requirements in the IXV On-Board Software

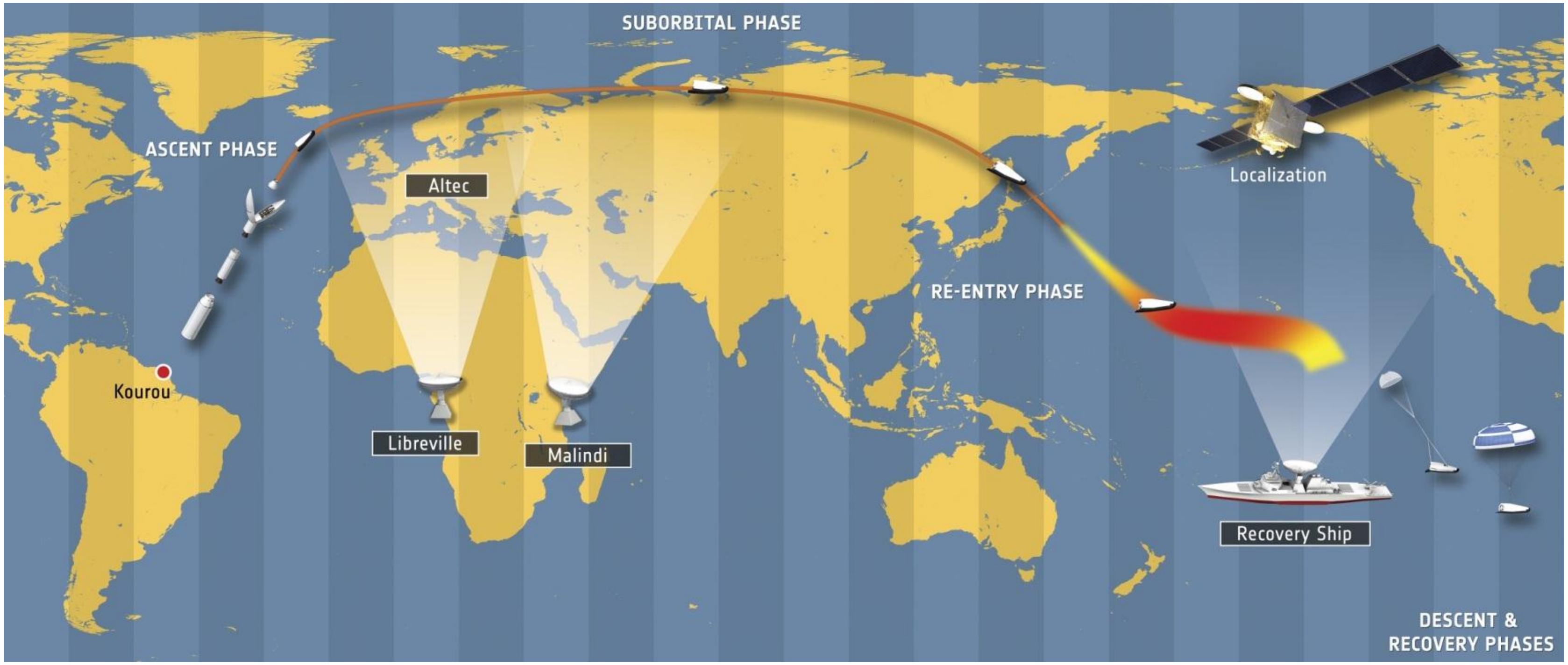
Santiago Urueña, Nuria Pérez, Bruno N. Calvo,
Carlos Flores, Andreas Jung

INNOVATING SOLUTIONS

IXV OVERVIEW
WHAT IS THE INTERMEDIATE EXPERIMENTAL VEHICLE?

Space vehicle from the European Space Agency (ESA) to **experiment on atmospheric re-entry** successfully launched on February 11th, 2015

IXV OVERVIEW
MISSION



- Around 100 min suborbital flight
- Representative return missions from Low-Earth Orbit
- Fully automated flight, unmanned vehicle
- No telecommanding after launch
- Monitored from ground stations

IXV ON-BOARD SOFTWARE
HIGHLIGHTS

Control Software in charge of **autonomously** flying the IXV vehicle following a predefined mission timeline

- Safety-Critical Software (DAL-B)
  - C99 & MISRA-C:2004
- Fault tolerance
  - 50 MHz LEON2-FT CPU (radiation hardened SPARCv8)
- Hard real time requirements
  - Caches enabled & code optimized
  - RTEMS operating system
  - Ravenscar profile

OBSW Application & Service layer:	
- **OBSW MVM:**	55 kLOC
- **OBSW GNC:**	22 kLOC
OBSW Basic layer:	
- **BSW (drivers):**	7 kLOC
- **RTEMS (subset):**	16 kLOC

RAVENSCAR PROFILE
INTRODUCTION

- Ada run-time profile
  - Subset of concurrency features to allow schedulability analysis

- Advantages:
  - Timing predictability, strict deadlines, low jitter
  - Small run-time to enable certification to high-integrity levels
  - Low resource consumption, high performance

- Defined in IRTAW 1997 (held in English village of same name)
  - 8th International Real-Time Ada Workshop

- Part of Annex D (Real-Time Systems) since Ada 2005

```
pragma Profile (Ravenscar);
```

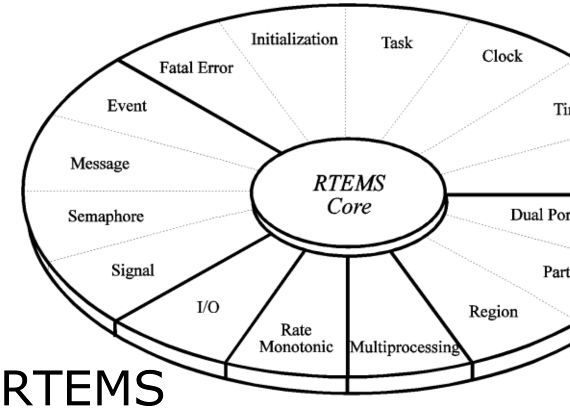
RAVENSCAR PROFILE

ARM 2012 DEFINITION

```
pragma Task_Dispatching_Policy (FIFO_Within_Priorities);
pragma Locking_Policy (Ceiling_Locking);
pragma Detect_Blocking;

pragma Restrictions (
                  No_Abort_Statements,
                  No_Dynamic_Attachment,
                  No_Dynamic_Priorities,
                  No_Implicit_Heap_Allocations,
                  No_Local_Protected_Objects,
                  No_Local_Timing_Events,
                  No_Protected_Type_Allocators,
                  No_Relative_Delay,
                  No_Requeue_Statements,
                  No_Select_Statements,
                  No_Specific_Termination_Handlers,
                  No_Task_Allocators,
                  No_Task_Hierarchy,
                  No_Task_Termination,
                  Simple_Barriers,
                  Max_Entry_Queue_Length => 1,
                  Max_Protected_Entries => 1,
                  Max_Task_Entries => 0,
                  No_Dependence => Ada.Asynchronous_Task_Control,
                  No_Dependence => Ada.Calendar,
                  No_Dependence => Ada.Execution_Time.Group_Budget,
                  No_Dependence => Ada.Execution_Time.Timers,
                  No_Dependence => Ada.Task_Attributes,
                  No_Dependence => System.Multiprocessors.Dispatching_Domains);
```

RAVENSCAR PROFILE
ADAPTATION TO RTEMS



- Success of Ravenscar profile, not just for Ada
  - Adapted to Java (A. Wellings & J. Kwon)
  - No previous publications to enforce Ravenscar in RTEMS

- Analysis of RTEMS documentation and source code internals
  - Qualified version of RTEMS by Edisoft in IXV
  - API compatibility (e.g. scheduling and synchronization policies)
  - Find primitives requiring memory allocation/deallocation
- Classification of RTEMS primitives
  - Unrestricted use (e.g. rtems_semaphore_obtain)
  - OBSW initialization only (e.g. rtems_interrupt_catch)
  - Completely forbidden (e.g. rtems_task_delete)
- Creation of RTOS wrapper (just allowed RTEMS primitives)
  - Periodic / sporadic tasks, mutexes, and interrupt service routines
  - Ported in new OBSW projects to different RTOS / CPU

IXV OBSW VALIDATION
OVERVIEW

- System validation tests
  - Open-loop and closed-loop tests
  - Emphasis in fault injection test (OBSW, OBC, avionics devices…)

- 100% branch coverage (ECSS-E-ST-40C decision coverage)
- Stack analysis
  - Check maximum stack usage at different validation tests

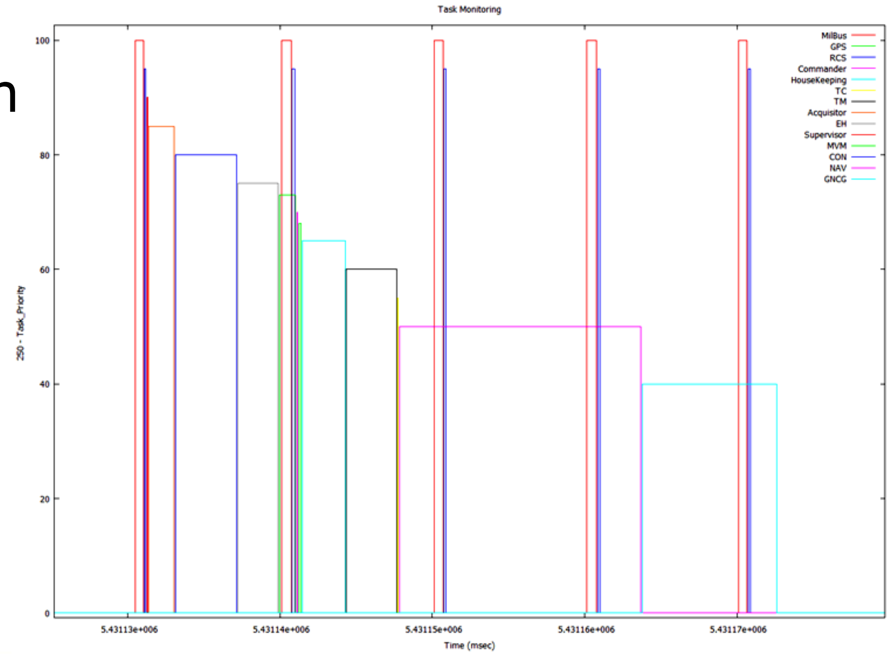
- Timing analysis
  - **Validation & Flight**: Task overruns monitored (event to Ground)
  - **Validation**:
    1. Response time analysis (dynamic analysis) → All tasks and mutexes
    2. WCET with RapiTime (dynamic & static analysis) → Critical tasks only
- Analysis of results reveals coding errors (e.g. nesting of locks)

IXV OBSW VALIDATION
TIMING ANALYSIS

- Measure the computation times of tasks and critical sections
  - Timestamps at beginning and end of each task activation
  - Timestamps at lock and unlock routines

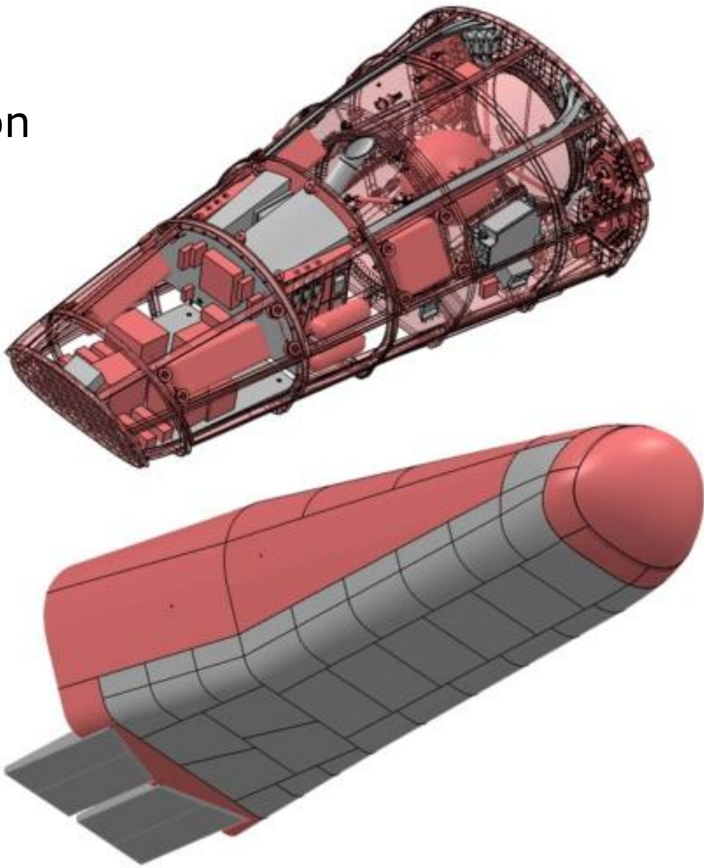
- Store all the timestamps generated in different system tests

- Script to analyze the computation time of every task activation and critical section



IXV OBSW VALIDATION
TEST ENVIRONMENTS

- Software Validation Facility (SVF)
  - Pure software emulation of OBC and avionics, Real-World simulator
  - LEON2-FT tsim emulator
  - Flexibility: introspection, debugging, fault injection
- Avionics / GNC Test Bench (AGTB)
  - Hardware avionics (engineering models) & software Real-World
  - OBC Functional Model (FPGA LEON2)
  - Representative avionics hardware
- Proto-Flight Model (PFM)
  - Flight hardware avionics + Real-World simulator (GPS stimulators…)
  - OBC with AT697E (ASIC LEON2)
  - Flight software



TIMING ANALYSIS
MEASUREMENT TECHNIQUES

- Measurement techniques analyzed (SVF only)

1. Low overhead traces (invalid instruction processed by tsim module)
   - Instrumented executable
   - Very easy to use, convenient just for application code
   - Some overhead (more instructions in memory, less code optimizations)
2. Breakpoints (debugger)
   - No instrumentation needed
   - Difficult to use, but very flexible
   - Initially no overhead

- Both techniques can also be used with RapiTime

TIMING ANALYSIS

COMPARISON AVERAGE COMPUTATION TIMES

Task	Priority	Period	Breakpoints	Traces	Overhead	
MILB	150	10.00	0.427	0.435	**0.0077**	**1.8%**
RCS	155	10.00	0.085	0.089	**0.0045**	**5.3%**
SUP	160	50.00	0.036	0.038	**0.0014**	**3.9%**
ACQ	165	50.00	1.634	1.657	**0.0238**	**1.5%**
GNC_C	170	50.00	1.956	1.973	**0.0168**	**0.9%**
EH	175	50.00	2.353	2.361	**0.0078**	**0.3%**
MVM	177	50.00	0.059	0.060	**0.0002**	**0.3%**
CMD	180	50.00	0.063	0.064	**0.0008**	**1.3%**
GPS	182	50.00	0.188	0.194	**0.0063**	**3.4%**
HK	185	50.00	2.061	2.066	**0.0049**	**0.2%**
TTM	190	50.00	1.190	1.124	**–0.0662**	**–5.6%**
TC	195	50.00	0.034	0.039	**0.0054**	**16.0%**
GNC_N	200	500.00	10.655	10.458	**–0.1977**	**–1.9%**
GNC_G	210	500.00	1.574	1.601	**0.0267**	**1.7%**

- Average task times comparison measured with both techniques
  - Higher overhead of traces than breakpoints as expected
  - Some tasks execute faster with traces! (cache anomalies?)

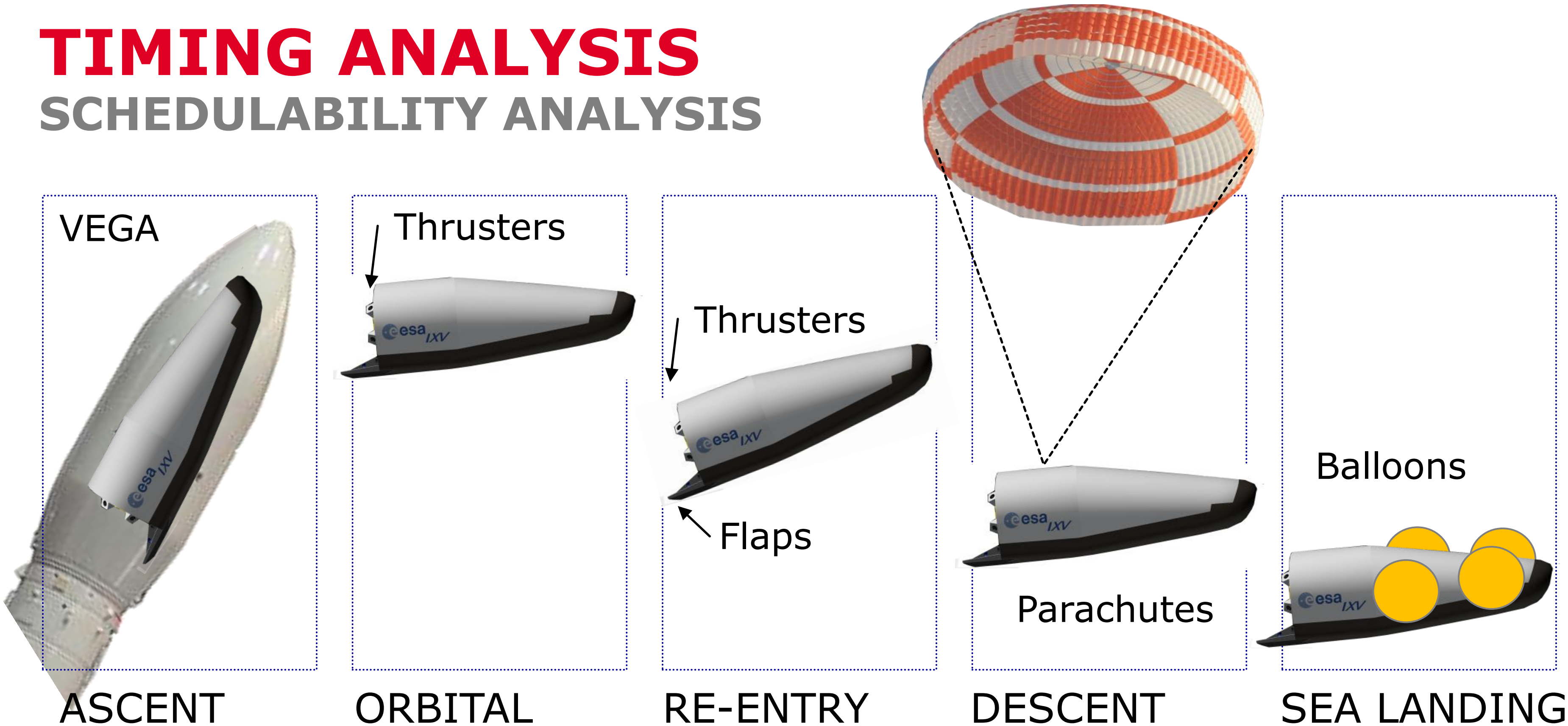
TIMING ANALYSIS
COMPARISON WITH NO TIMING MEASUREMENTS

Event message	Reference timestamp (no timing measurements)	Interference breakpoints	Interference traces
…	…		
EV_MOS_ACTION_TRIGGERED	4316.97509 second	+0.00 ms	+424.32 ms
EV_ACTION_SUCCESSFUL_EXEC	4316.97686 second	+0.00 ms	+424.38 ms
EV_MOS_TRANS_TO_REENTRY	4371.97497 second	+0.06 ms	−75.68 ms
EV_ACTION_SUCCESSFUL_EXEC	4371.97552 second	+0.06 ms	−75.62 ms
…	…		

- Timestamp comparison of transmitted packets in same SVF test
  1. Flight executable with no timing measurements
  2. Flight executable with breakpoints
  3. Instrumented executable with traces

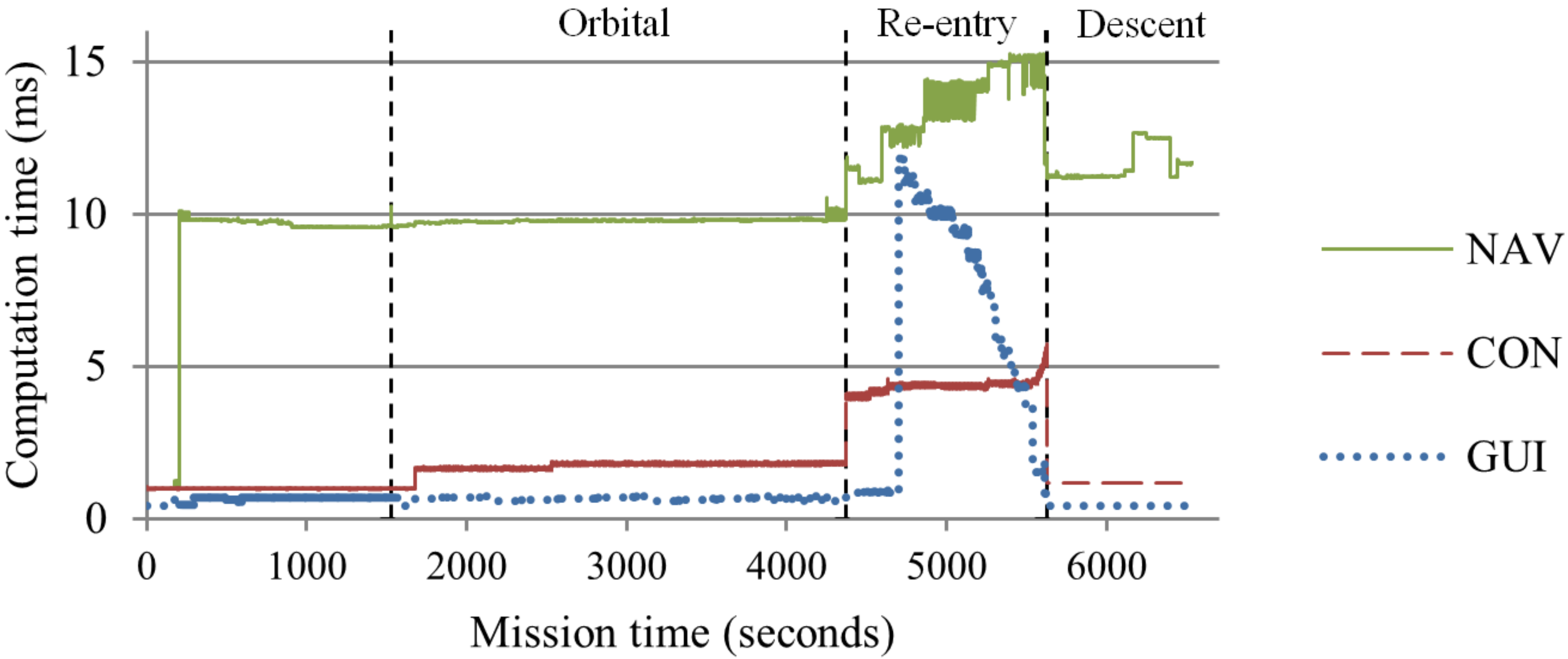
- Breakpoints also introduce some timing overhead
- Overhead of traces affects test behavior
- Similar comparison with AGTB (SVF adds less than 1 ms)

TIMING ANALYSIS
SCHEDULABILITY ANALYSIS



- Separate schedulability analysis for each mission phase
- The OBSW is schedulable in all phases, with 35% CPU margin
- No task overrun ever detected during validation or the mission

TIMING ANALYSIS
TIMING RESULTS



- Publication of response times of every thread per mission mode
- Useful in future projects (budget estimations at early phases)

TIMING ANALYSIS
CONCLUSIONS

- Ravenscar profile allows the schedulability analysis
  - IXV OBSW schedulable in all phases

- Different measurement techniques analyzed
  1. Low-overhead traces
     - Easy to use, but just application code
     - Some overhead and noticeable software interference
     - Convenient during development

  2. Breakpoints
     - Very flexible, both for application and RTOS code
     - Negligible overhead, minor software interference
     - Difficult to use manually, just for final tests

- Emulators and CPUs must provide better timing features
- Critical software should monitor its timing attributes

Thank you

Santiago Urueña

Section Head Critical Software

Software Engineering department

Email: suruena@gmv.com

www.gmv.com

INNOVATING SOLUTIONS

IXV OVERVIEW

PAST (EUROPE) & FUTURE

HERMES (1990)
(cancelled)

ARD (1998)
(successful)

X38/CRV (1999)
(cancelled)

PHOENIX (2004)
(cancelled)

USV1,2,3 (2007…)
(successful)

EXPERT (2012)
(on-hold)

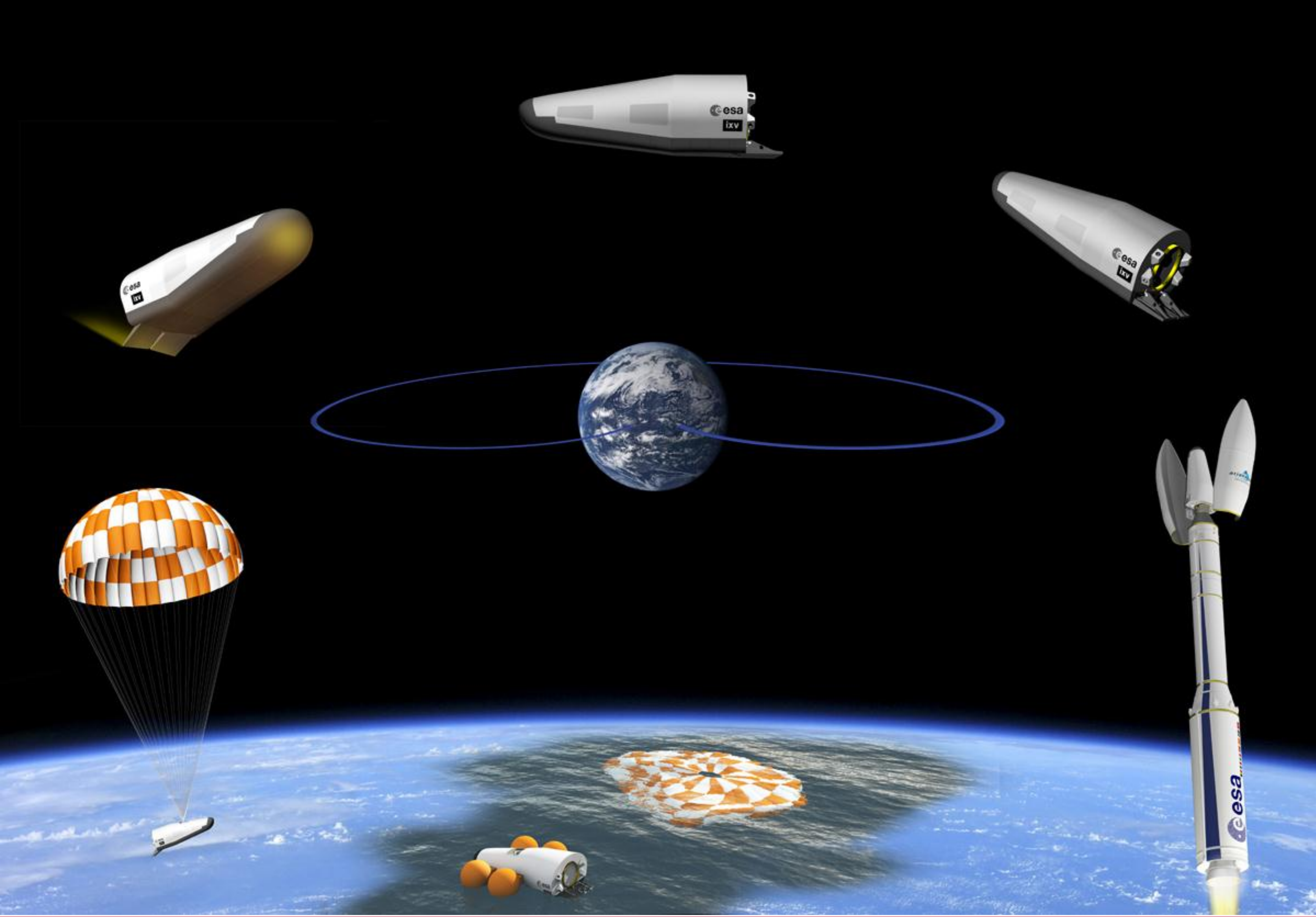
IXV (2015)
(successful)

Reentry vehicle applications:

- Servicing of orbital infrastructures
  (e.g. ISS)
- Servicing of satellites
  (e.g. refueling or disposal)
- Robotic exploration
  (e.g. sample return from Mars)
- Microgravity experiments
- Earth sciences
  (e.g. high-altitude atmospheric research)
- Earth observation
  (e.g. crisis monitoring)

Next step: **PRIDE**
(Programme for Reusable In-orbit Demonstrator for Europe)

 |


esa
ixv