VEGA

“First Application of the Generic Emulated Test Software, GETS, in the LISA Pathfinder Operational Simulator”

SESP 2008, 8th October 2008, ESTEC

Joachim Ochs	VEGA
Michael Irvine	VEGA
Mehran Sarkarati	ESA/ESOC
Mariella Spada	ESA/ESOC

Independent Programme and System Assurance
Technical Excellence . Pragmatic Solutions . Proven Delivery

Presentation Overview

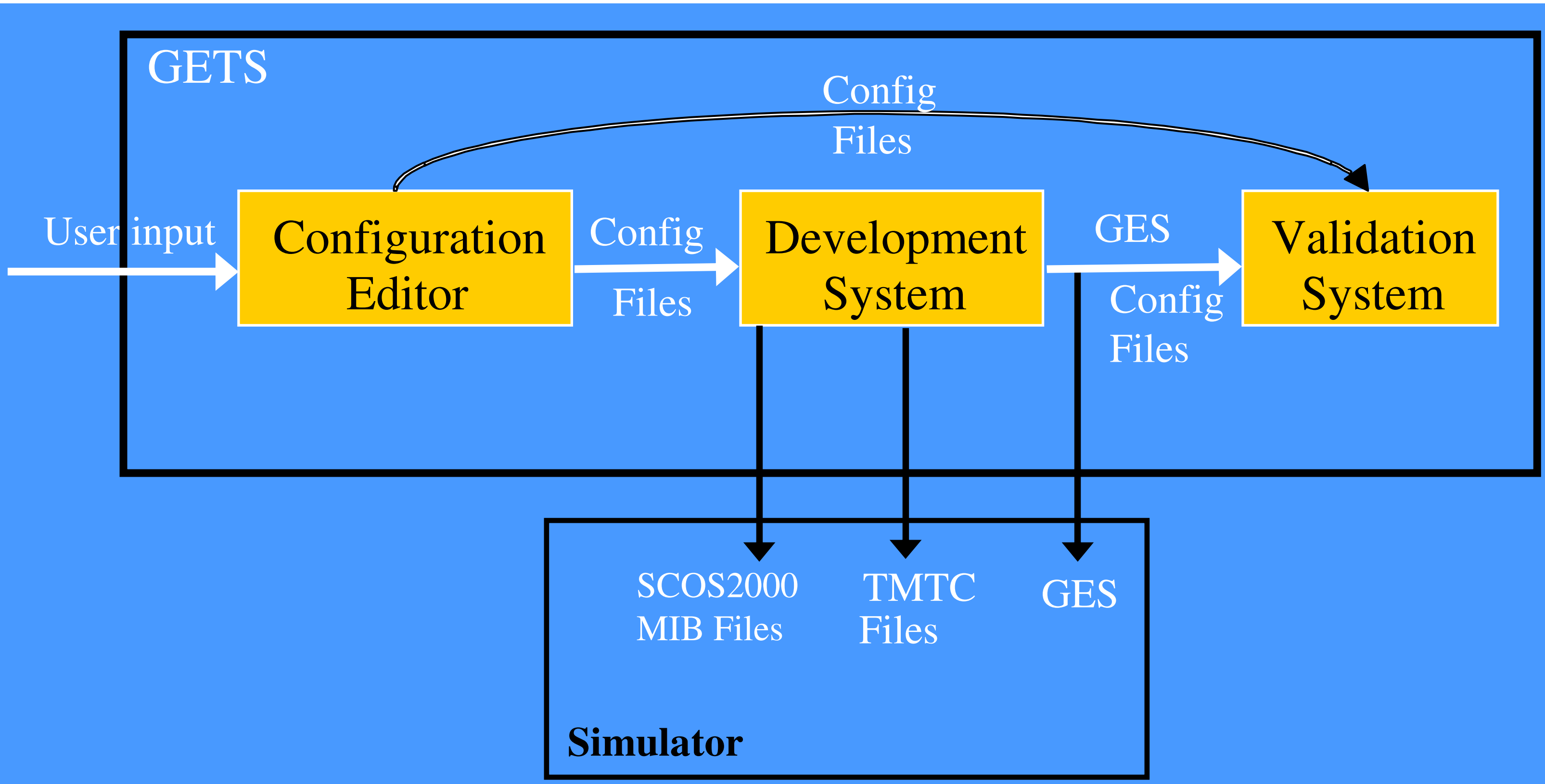
- GETS – Overview and Background
- Application on the Lisa Pathfinder Operational Simulator
  - Objectives
  - Technical Approach
  - Lessons Learnt
- Future considerations for GETS
- Summary and Conclusions
- Questions….

GETS – Overview and Background

GETS Overview

- Study performed by VEGA and Dataspazio for ESOC
- Presented at SESP 2006
- Focuses on supporting emulator-based simulator development
- Aims at reducing simulator dependency on availability of mission OBSW
- Helps to de-couple simulator and mission OBSW schedules
- But how does GETS support this?

The GETS System



The GETS System

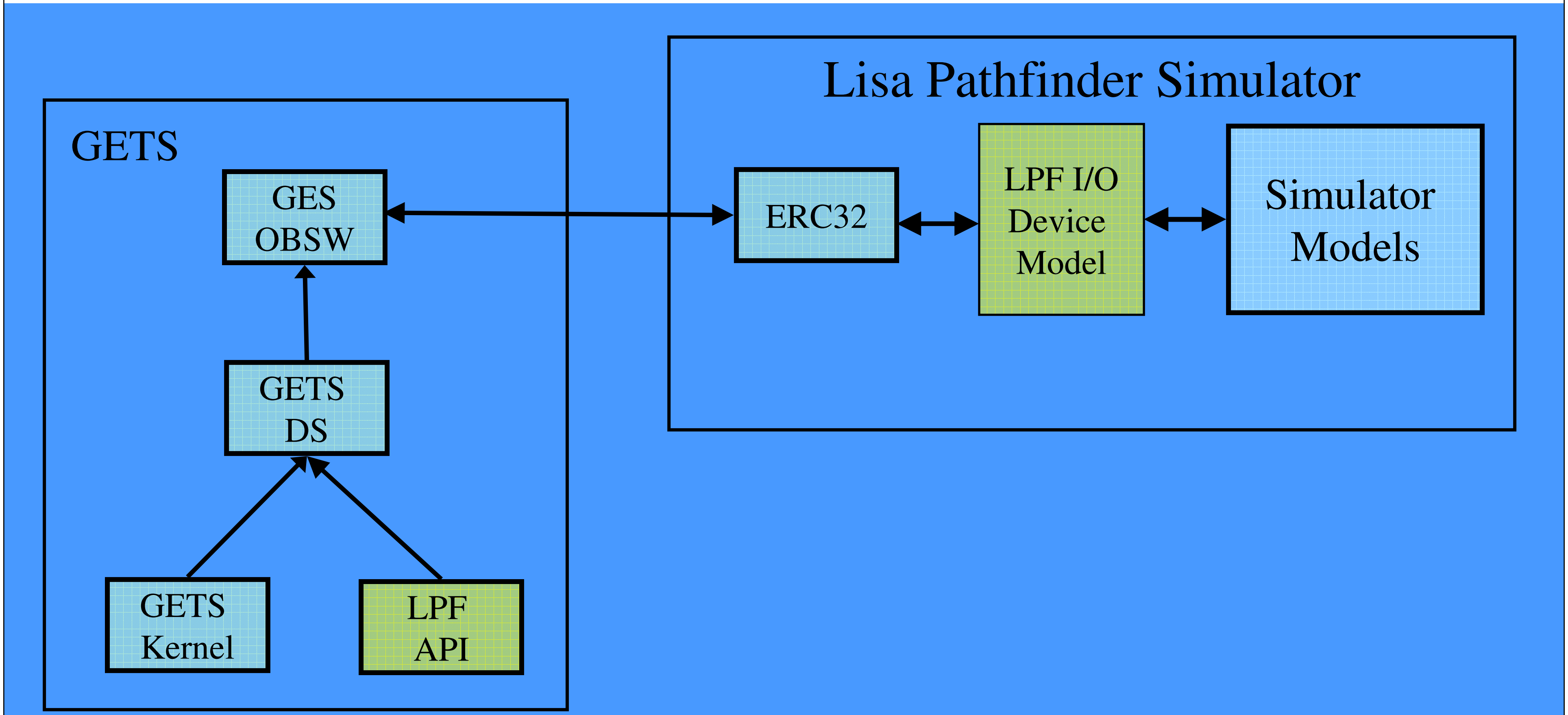
- Configuration Editor
  - Enter mission parameters (e.g. RT addresses, Virtual channel etc.)
  - Create TM/TC definitions –via GUI or import from SCOS-2000 MIB files
- Development System
  - Based on ERC32 C cross-compiler
  - Builds the GES stub OBSW → creates S-record image for ERC32
- Generic Emulator Software (GES)
  - Stub OBSW focusing on core PUS services and data bus handling
  - Supports 1553 or OBDH buses
  - Split between Kernel and API functions
  - API based on GETS HW/SW ICD
- Validation System
  - SIMSAT-based simulator system – supports testing of the GES

Application on the Lisa Pathfinder Operational Simulator

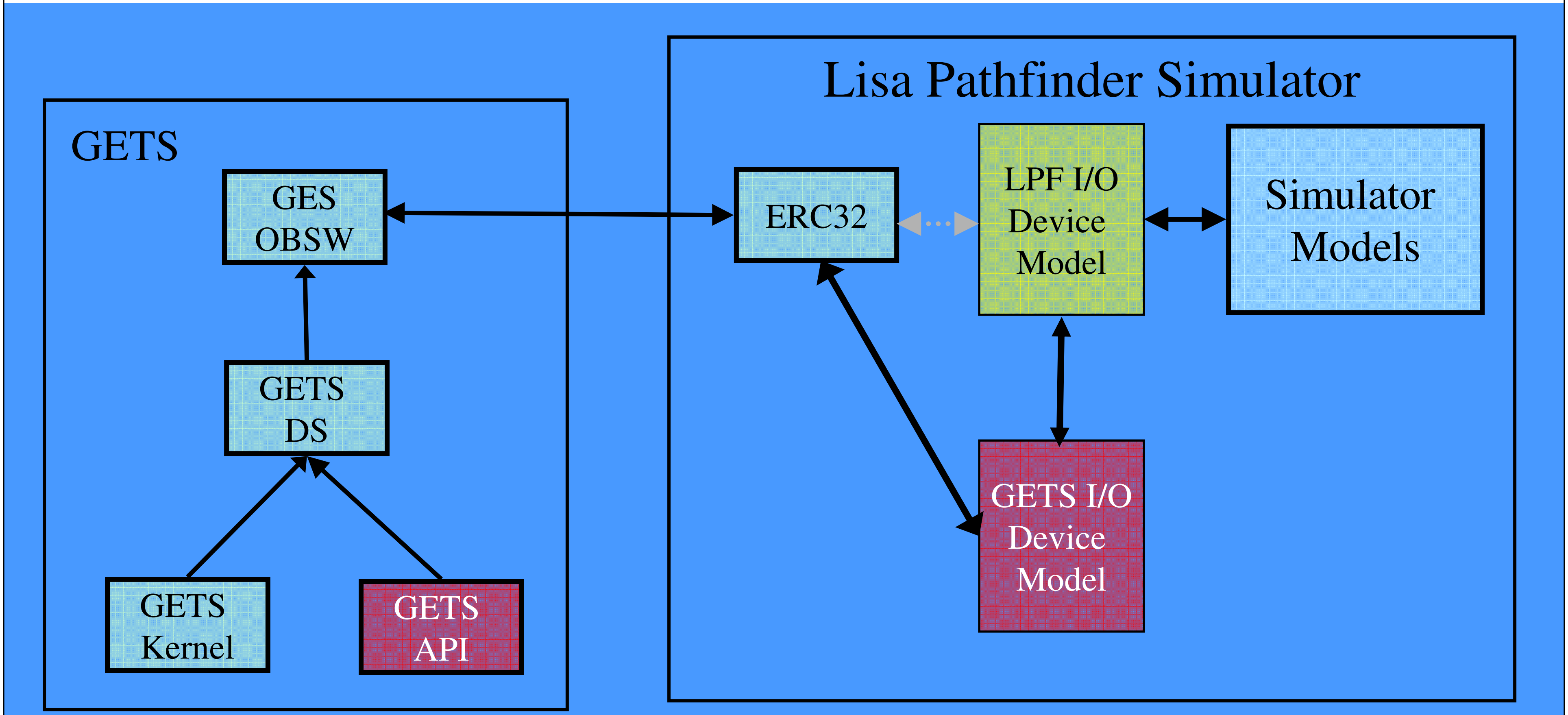
Aims and Objectives

- Adopt GETS in the LPF Operational Simulator
- No need for a switching/functional model of the CDMU in the first delivery of the simulator
- Implementation of a high fidelity CDMU model already in the first delivery of the simulator
- Validating the I/O interfaces of all simulation models (HW/SW interface)
- On-going use of GETS for regression testing of I/O device model interfaces and mission API OBSW.
- Provision of the GES-based simulator to the flight control team for early validation, training and familiarisation purposes
- Streamlining the integration of the real OBSW, once it becomes available

Possible Technical Approach 1

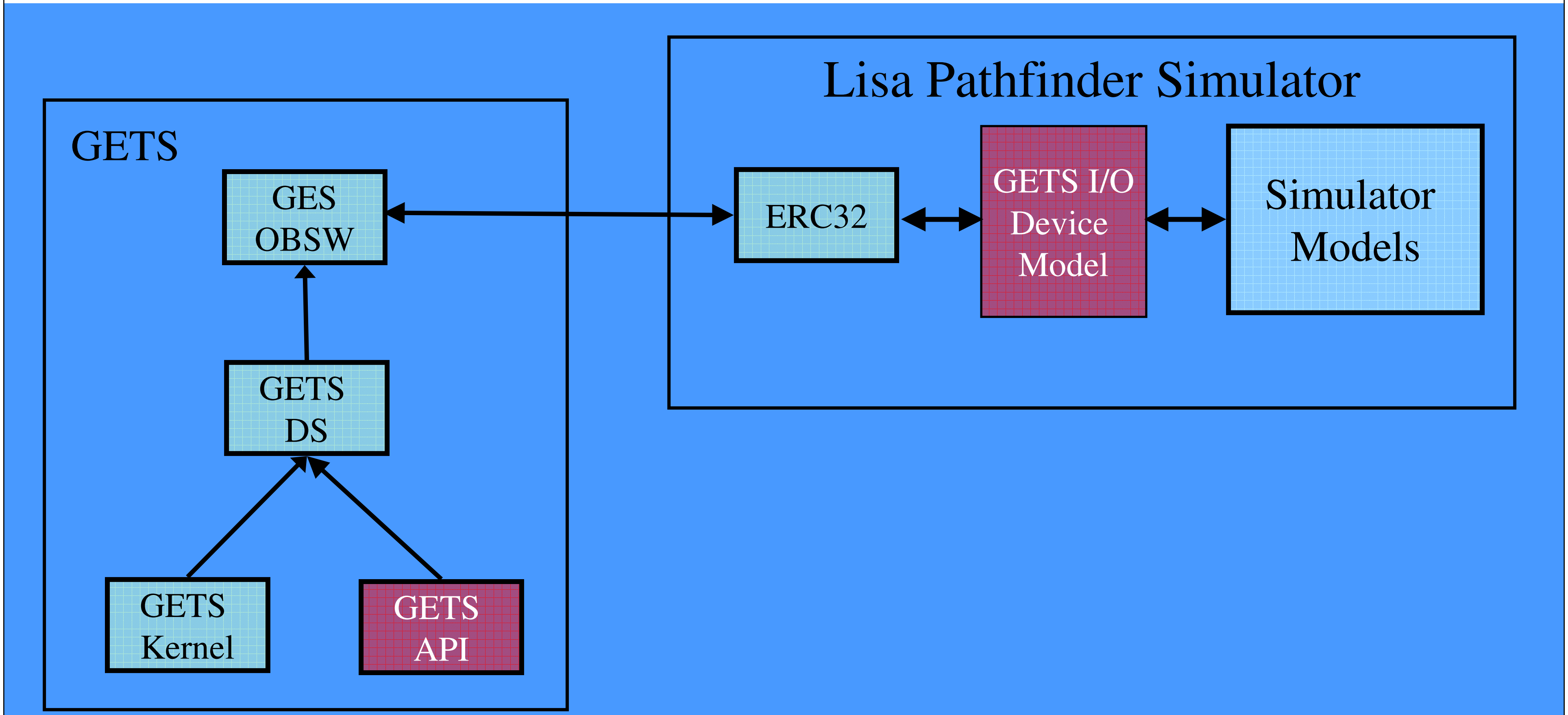


Possible Technical Approach 2



VEGA

Possible Technical Approach 3



Selected Technical Approach

- GETS provides an API based on GETS-specific interfaces (i.e. GETS HW/SW ICD).
- Approach one has been selected → Adaptation of the GETS source code to use the LPF OBSW API
- Integration of the LPF Basic Software in GETS
- Most Realistic Approach → Minimise the required changes on the simulation models after integration of the OBSW

Lessons Learnt - General

- Focus of GETS study was to support Simulator Developers

  ➔ GETS functionality was more focused on S/C onboard interfaces

- The GETS-based LPF Operational Simulator is provided to the FCT
  - What subset of the OBSW functionality should GETS solution provide?
  - Mimic the same TM/TC interface to the users
  - Manual adaptation of GES to implement the discrepancies between OBSW, Mission DB and SW/HW ICDs: Packet definitions, parameter types, calibration curves, etc.
- GETS has been developed as a prototype in a study
  - The GES OBSW was sufficient for the prove of the concept
  - GES can not replace the need for OBSW for an operational simulator
  - GETS was not developed with focus on extendibility
  - GES does not contain a Data Pool (common concept within Mission OBSW).
  - Focus on handling TM/TC for external units, not on OBSW-specific TM and TC
  - GETS does not implement the functionality of the OBSW Application layer, e.g. FDIR, AOCS

Lessons Learnt - GES

The following points impacted GETS use on LPF (highest impact first):

- Large differences between GETS API and mission OBSW API
  - Complex to replace GETS API with LPF OBSW API!
  - GETS is based on XGC compiler – single thread with interrupt handlers
  - LPF OBSW based on RTEMS – multiple RTEMS threads and messages
  - RTEMS is more complex than XGC
- Complexity due to OBSW “re-formatting” of acquired TM – format in DataPool different from acquired TM format
- No support for SpaceWire – had to be added for LPF GES
- Additional needs to handle Service 8 commands
- GES does not include a Data Pool – TM acquisition coupled with TM generation.

Future considerations for GETS

Future GETS Options – Tools Updates

- Configuration Editor:
  - Better performance concerning imports and modification
  - Design of the GUI to be more intuitive
- Development System:
  - File-based input mechanism should be improved
- Consolidation of both GUIs into a single GUI-based tool

Potential GES Updates

- Re-factor GES design to make it more extendable for future missions:

  + more focus on OBSW APID TC/TM handling/generation

    → assess approaches for TC decoding and TM encoding

    → possible use of simulator encoder/decoder component

  + add support for Data Pool

  + add support for multiple data buses in parallel

- Consider how to handle OBCPs within GES (if required for future simulators/missions)
- Update GES to more natively support use of RTEMS.

Summary and Conclusions

Summary and Conclusions

- It was possible to adopt GETS in the LPF operational simulator
- This was a complex task
- GETS Supports the sim developer implementing the I/O devices
- Current limitations on TM generation
- Limitation concerning complex data handling interfaces
- Standardisation of onboard I/O device would be a benefit for applying the GETS implementation
- Provision of a Reference Architecture for OBSW would allow the extension of the functionalities in Generic Emulated Software, GES

Any questions....?

VEGA

www.vega-group.com

Joachim Ochs	VEGA
Michael Irvine	VEGA
Mehran Sarkarati	ESA/ESOC
Mariella Spada	ESA/ESOC