New Tools for Spacecraft Simulator Development
Why use Simulators?

• Replace the Spacecraft
• Support to design
• Support to testing
  – replacement of real equipment in destructive or expensive tests or when real equipment is unavailable
  – validation of embedded software
  – data source for control systems validation
• Support to training
• Support to failure investigation and correction
Concurrent Development

Spacecraft
  Specification
  Design
  Development
  Operations

Simulator
  Specification
  Design
  Development
  Operations

PDR

CDR

Launch

Database
  Flight s/w

Ops-G
M. Pecchioli
GSAW
March. 2007  Page 3
Development and Validation

- Spacecraft
- Development
- Simulator
- Development
- Sims Campaign
- Database
- Flight s/w
- CDR
- SVT-0
- SVT-1
- SVT-2
- Launch

Development and Validation Diagram:

- Spacecraft Development
- Simulator Development
- Flight software
- Database
- CDR
- SVT-0
- SVT-1
- SVT-2
- Launch
- Sims Campaign
Avoiding concurrent development: sim & MCS

- Historically, one of the first uses of the simulator was a data source/sink during Mission Control System development and testing
- Problem: the simulator was developed in parallel to the MCS, so if a problem occurred, difficult to isolate
- Solution: develop standard tools which are validated on previous missions and reuse them for new MCS
- Second advantage: permits delay to simulator development
Avoiding concurrent development: sim & spacecraft

• Impact from inputs from the satellite development on the simulator development. Dependency between Simulator and mission schedules for the ICDs, OBSW, SDB
• the provision of the OBSW typically end up on the critical path for the simulator
• Possible impact on Integration of the simulator
• Impossible to perform end-to-end test without OBSW
• Need to decouple from spacecraft schedule
Standards

- Standards can be *de facto* or *de jure*
  - *De jure* external standard *(CCSDS, ECSS)*
    - Written telemetry and telecommand standards, utilisation std
  - *De jure* internal standards
    - Written TM/TC database standard
  - *De facto* standard
    - Few suppliers => Hardware commonality for many units
    - Few space-qualified parts
    - Generic models (dynamics, orbit)

- Benefit from standards
  - Reusable models (ground station models, processor emulators)
Research and Innovate

• Small study to mitigate impact of late flight software
• Generic Emulator Test System - GETS
  – Standard packet format - syntax
  – Standard telemetry and telecommand packet services (semantics)
    • Configurable routing and addressing
  – Standard database to import from mission control system
  – Standard computer – ERC32 => standard emulator
  – Standard peripherals – OBDH bus & mil-std-1553
• Make own embedded software for test purposes
  – Generated source code including database and packet routing information
  – Use open source cross compiler for ERC32 (gcc )
  – Own validation simulator
Typical Simulator Design

- SIMSAT Simulation
  - Kernel
  - SMI I/F
- Ground Segment Simulation
  - SMI I/F
  - Spacecraft Model Code (C++)
  - PM with Emulator
- Mission Control System
  - TTC Streams
  - Flight Software
  - Spacecraft Database
  - Risk item
GETS Study

• Is it possible to produce a tool which allows the user to generate a replacement software?

• Which are the commonalties between OBSW for different missions?
  – The TC and TM packet structures / functionality's are defined by ECSS/PUS standard for the ESA mission -
  – The API are basically defined by the used hardware components
  – the interface types are similarly between the different missions even when the implementation is different.
GETS Overview

GETS

Configuration Editor → Config Files

Development System

Validation System

GES

Config Files

SCOS2000 MIB Files

GES

TMTC Files

Simulator

User input

Config Files

Config Files

MIB Files

TMTC Files

Simulator
GETS - Configuration Editor

- The Configuration Editor is an off-line tool used to enter the mission-specific information including:
  - TM and TC packet and parameter definitions.
  - The relationship of TM packets to virtual channels.
  - Data bus interrogations.
  - Bus addresses.
  - The relationship of Application Packet ID to data bus addresses for external units.
  - Configuration are in XML-format
  - Support import of SCOS2000 MIB files
  - Java based
GETS - Development system

- The Development System is the component of the GETS system dedicated to the building of the Generic Emulator Software (GES).
- It performs the following main operations:
  - Imports the configuration files defining the TC/TM packet structure and the bus configuration (from Configuration Editor).
  - Load its own configuration files defining the GES Memory Map I/O, the PUS constants definition and the list of telecommand APID codes reserved to high priority TC, GES and user code;
  - Inserts the configuration data into the GES source code;
  - Builds the GES onboard software image (Prototype in SREC format);
  - Exports a set of SCOS-2000 format MIB files.
  - XGC cross compiler for ERC 32 target platform used in the prototype
GETS - Generic Emulator Software

• The GES is an OBSW image generated by the Development System. It provides a subset of typical mission OBSW functionality aimed at supporting simulator integration activities.

• The main GES OBSW features are:
  – GES is configured based on user input to CE and DS tools.
  – Interfaces to external I/O hardware devices are via GES API software.
  – Distribution of TCs using external bus interfaces (via GES API).
  – Acquisition of TM using external bus interfaces (via GES API).
GETS Validation System

• Real-time SIMSAT-2003 based simulator that loads and executes the GES in an ERC32 emulator.
  – ERC32 emulator embedded within a common interface
  – I/O device model layer handling the I/O operations performed by the GES software
  – Simple bus models linking the I/O layer to bus users
  – Simple Data Source/Sink models connected to the various data buses (External Models)
GETS Validation System

The Validation System (VS) is used to support validation of the GES software image generated by the Development System.
Benefits of GETS

• Simulator developer
  – Integration of the Simulator independent from the availability of the OBSW

• End customer
  – Schedule of the simulator delivery decoupled from the OBSW schedule, reduces schedule risks

• Infrastructure
  – performance prediction of the simulator/emulator, before the real OBSW is arriving
  – Generic Emulator Software can be used as a test tool, e.g to compare processor emulators.
Pieces of the Puzzle

Standardised Interfaces to ground segment

Ground Models

Man Machine Interface

SIMAT

Model portability

Generic Models

Automation And Scripting

Previous models

Mission Control Functions

Data calibration

Standard MMI

commanding

planning

Data distribution

Standardised Interfaces to Space segment

TM/TC Database

OBSW images

Mission Control Functions

Data calibration

Standard MMI

commanding

planning

Data distribution

Standardised Interfaces to Space segment

TM/TC Database

OBSW images
Questions ?