MODEL AND SIMULATION BASED SATELLITE ENGINEERING

Gustavo WIMAN, Leonardo HANDSZTOK
Introduction

• INVAP´s approach to modeling and simulation based engineering for satellite projects is presented.

• It is based on INVAP experience in the aerospace business and reflects organizational learning and culture build up.

• Standards may guide the process, but learning-by-doing is irreplaceable.

• Simulation and EGSE for space projects are closely related and constitute core parts of an engineering support system.

• In relation to this presentation, there was another one about “satellite engineering support system” in session 2 and a poster showing simulators development at INVAP.
Spacecraft Context

- Ground station
- Environment
- Launcher

Spacecraft
- Service Pl.
- Payload

Ground support system
"V" development process

- **Domain**
  - Feasibility analysis
- **System**
  - Requirements
  - Architectural design
- **Subsystem**
  - Detailed design
- **Components**
  - Construction / Programming

- **Integration**
- **Unit testing**
- **Acceptance**
- **Qualification**
- **Utilization**

V&V (Verification & Validation)
Model Aided Systems Engineering
Model Based Systems Engineering

- Structural views
- Behavioral views
- Requirements views
- Parametric views
- Model (DB)
- Requirements
- System
Model Based Systems Engineering

- Systems models, using modeling languages like SysML, are developed at different levels of detail/decomposition/integration:
  - System
  - Subsystem
  - Components
- Present tools can’t execute these models
- But include the necessary information to:
  - Build the modeled elements
  - Simulate the modeled elements
  - Generate documentation
Model Based Systems Engineering

1. Structure

2. Behavior
   - interaction
   - state machine
   - activity/function

3. Requirements

4. Parametrics
Present use of simulators

- Feasibility analysis
- Power
- Integration
- Unit testing
- Construction / Programming
- Detailed design
- SVF
- Structural & thermal
- Arquitectural design
- Requirement
- AOCS
- V&V
- Acceptance
- Detailed S/C sim
- Qualification
- Utilization
- Mission
- Time

Levels of abstraction:
- Domain
- System
- Subsystem
- Components
What is not simulated (early)

- Black-box functionality
- Operations
- Performance
- Usage of resources:
  - Power
  - Memory
  - CPU time
  - Message traffic
- Other non-risky(?) stuff

Partial tests are made on breadboards and EM

Some things are analyzed by mental experiments
Lessons learned

• Late writing of tests procedures leads to
  • Nonobservability of variables that prevents verification and validation of requirements and specifications
• Inability to debug test procedures leads to
  • Bugs found in test procedures when run
• Inability to run tests on simulators leads to late problems identification, like:
  • Schedulability and performance problems
• Costs can be reduced by early testing on simulators of technologically risky elements and concepts
Executable Models - Simulation

• The model must be a full specification of the elements to be built and simulated.
• The model must contain all the information needed for simulation.
• Tools generate input for simulators from the model database.
• Different kinds of simulators are integrated in a co-simulation environment (SMP, FMI).
• Executable models / simulators will be tested with the same test support system used for flight components.
• The test support system tests black boxes using
  • virtual interfaces for simulators
  • physical interfaces for physical components
Executable Models - Simulation

- Executable model
- System
- Test

- Model
- Simulator
- Test

Tools with model simulation capability
Separate tools for modeling and simulation
Properties to be modeled/simulated

- Attributes
- Functions with execution times
- State machines
- Communications

UML & MARTE example

From www.omg.org
Models Evolution

- Models/simulators are developed at different levels of abstraction.
- When passing to lower levels of abstraction (from coarse-grain to fine-grain):
  - functions are decomposed.
  - interfaces are refined.
- Coarse-grain models allow analyzing system concepts and validating user requirements.
- Fine-grain models allow:
  - architectures comparison and trade-off analysis
  - analyzing detailed functionality and performance
  - developing and debugging test procedures
Test Driven Development

- Tests for covering every requirement are “executable requirements and specifications”.
- Test procedures are defined at the beginning of the project and are refined at the same time as the design is decomposed into finer grain elements.
- These system tests are run at different levels of system decomposition/integration (both sides of the “V”).
- Tests are first run on system simulators and are debugged interactively with the system model/simulator.
- The system simulator allows early involving of users in the development process.
Testing triple-V

From Using V Models for Testing
Software Engineering Institute
Carnegie Mellon University
Concurrent Engineering

- M&SBSE enables concurrent engineering by integrating multiple disciplines that share a common model.
- Using concurrent engineering in the initial phases of the project to build a rapid prototype allow a consistent, less expensive and lower risk conceptual/basic design.
- The necessary infrastructure includes:
  - Network of interconnected simulators (co-simulation)
  - Component models libraries
  - Environment simulators
  - Mission analysis tools
The development of complex software like a satellite simulator requires the work of a multidisciplinary group of people and the use of diverse tools to help carry out the project. For the different stages of the simulator design, MBSE methodologies are used with SysML as language. It is important to recognize that at this stage not only must a software architect participate, but also someone with a well-developed systemic vision that can correctly interpret the interaction of the systems that are part of and that interact with the satellite.

The simulator development follows a V model, starting testing tasks from an early stage. According to the level of testing to be designed and executed, different tools are used. CppUnit is the one chosen for unit testing, Python for the development of test scripts and Jenkins for system and subsystem regression tests.

As programming language C++ is choosen under a linux platform, the configuration control of the code is done using git and for the tracking of tasks and bugs Jira is used.
Questions?
Thank you!

guswiman@invap.com.ar
lhandsztok@invap.com.ar