On Board Software

December 2012
OBSW Characteristics

- On Board Software Constraints
- On Board Software Dependability
- On Board Software Criticality
- On Board Software Development Lifecycle
- On Board Software Documentation
OBSW: Constraints

- **Embedded Software**
  - Cross Development Environment, bounded Memory Footprint and Processor Consumption

- **On Board Software**
  - Single Event Effects, (Upset, Latch Up), Memory Scrubbing

- **Remote Software**
  - Need for large autonomy

- **Critical Software**
  - Possible catastrophic consequences of failure

- **Dependable Software**
  - Need for high Reliability, Availability, Maintainability and Safety

- **Real Time Software**
  - Processor load, scheduling issues, soft and hard deadlines

- **Deterministic Software**
  - No Dynamic thread creation or memory allocation, budget and schedulability
OBSW: Dependability

Achieving mission objectives and ultimate mission success relies on dependability of the space Systems.

As software plays more and more a prominent role in space systems, its contribution to the overall system dependability becomes a vital aspect of system development.

Troubles

- **Error**: A wrong or missing human action or thought
- **Fault**: An incorrect step, process or data definition in a program
- **Failure**: The inability of the software to perform its required functions.

RAMS

- **Reliability**: continuity of correct service.
- **Availability**: readiness for usage.
- **Maintainability**: easiness of repair/upgrade.
- **Safety**: non-occurrence of catastrophic failure

Analysis

- **FMECA**: Failure Mode Effects and Criticality Analysis
- **FTA**: Fault Tree Analysis
- **FHA**: Functional Hazard Analysis
- **HSIA**: Hardware Software Interaction Analysis
- **SCCFA**: Software Common Causes Failure Analysis
- **SWCA**: Software Criticality Analysis

Strategies

- **Fault Prevention**: avoidance and reduction of fault causes
- **Fault Tolerance**: avoidance and reduction of fault consequences
- **Fault Removal**: removal of fault occurrences
- **Fault Forecasting**: prediction of behaviour in presence of faults

FDIR

- **Fault Detection**: e.g. through monitoring
- **Fault Isolation**: determination of the cause
- **Fault Recovery**: e.g. through redundancy
- **Certifiability**: Ability to get stamp from certification body

FDIR is imperative to guarantee a dependable and autonomous system with a minimal risk of ruinous failure

Integrity

- **Maintenance of data consistency**

Security

- **Non disclosure of unauthorized info**
OBSW: Criticality

Software that if not executed or if not correctly executed or whose anomalous behaviour could cause or contribute to a system failure resulting in:

<table>
<thead>
<tr>
<th>A: Catastrophic Consequences</th>
<th>B: Critical Consequences</th>
<th>C: Major Consequences</th>
<th>D: Minor Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of Life, life threatening, personnel injuries, permanently disabling injury or occupational illness, Loss of an element of an interfacing manned flight system. Damage to other equipment. Loss of launch site facility facilities or loss of system. Severe detrimental environmental effects.</td>
<td>Permanent or non-recoverable loss of the satellite's capability to perform its planned mission Temporarily disabling but not life-threatening injury or occupational illness. Major damage to flight system or loss or major damage to ground facilities. Major damage to public or private property or major detrimental environmental effects.</td>
<td>Negligible or minor effect on the satellite’s mission and operability A detailed definition is left on a project by project basis and reported in its risk policy. <em>Example is Mission Simulation Software</em></td>
<td>A detailed definition is left on a project by project basis and reported in its risk policy. <em>Example is Test Software</em></td>
</tr>
</tbody>
</table>

In the DO-178-C (*Software Considerations in Airborne Systems*), the Design Assurance Level (DAL) is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system. The failure conditions are categorized by their effects on the aircraft, crew, and passengers.

- **Catastrophic** – Prevents continued safe flight or landing, many fatal injuries. Failure may cause a crash. Error or loss of critical function required to safely fly and land aircraft.
- **Hazardous/Severe** – Potential fatal injuries to a small number of occupants. Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers. (Safety-significant)
- **Major** – Impairs crew efficiency, discomfort or possible injuries to occupants. Failure is significant, but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries) or significantly increases crew workload (safety related)
- **Minor** – Reduced aircraft safety margins, but well within crew capabilities. Failure is noticeable, but has a lesser impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change)
- **No Effect** – Does not effect the safety of the aircraft at all. Failure has no impact on safety, aircraft operation, or crew workload.
OBSW: Lifecycle

Phase A
- System and Mission Requirements
  - RFI, RFQ, RFP, ITT, SOW

Phase B
- Software Specification
  - PMP, PAP, CMP, SDP, SVVP

Phase C
- Architectural Design
  - SDD-AD, ICD
- Detailed Design
  - SDD-DD
- Coding
  - Source Code, Executable, Makefiles, Scripts

Phase D
- Integration Test
  - ITR
- Unit Test
  - UTR
- Acceptance Test
  - ATP
- Validation Test
  - VTP, VCD

Phase E
- Operations and Maintenance
  - AR (TRR/TRB/DRB)
  - QR (TRR/TRB/DRB)
  - CDR (TRR/TRB/DRB)

Concept of Operations
- December 2011
- ULG - On Board Software Overview
OBSW: Documentation

- Technical Notes
- Documents
- Plan
- Reports
- Matrixes

- System Requirements Document
- Mission Requirements Document

- Project Management Plan
- Software Project Development Plan
- Software Verification and Validation Plan
- Software Configuration Management Plan
- Software Product Assurance Plan
- Risk Management Plan

- Progress Report
- Product Assurance Report
- Software Budget Report

- RID: Review Item Discrepancy
- SPR: Software Problem Report
- NCR: Non Conformance Report
- CR: Change Request
- CP: Change Proposal
- CCN: Contract Change Notice
- RFD: Request for Deviation
- RFW: Request for Waiver

- Technical Specification
- Interface Control Document
- Architectural Design Document
- Detailed design Document

- Unit Test Plan
- Unit Test Procedures
- Unit Test Report
- Integration Test Plan
- Integration Test Procedures
- Integration Test Report
- Validation Test Plan
- Validation Test Procedures
- Validation Test Report

- Software User and Operation Manual

- Software Release Note
- Configuration Items Data List

- DRL: Document Requirement List
- DRD: Document Requirement Description
OBSW Architecture

• Interfaces and Data Flows
• Functional Architecture
• Static Architecture
• Dynamic Architecture
• Deployment Architecture
Interfaces & Data Flows

**Telecommands**
- High Priority commands ➔ Hardware
- Nominal commands ➔ Software
- Macro commands ➔ Expanding
- Time tagged commands ➔ Scheduling

**Telemetries**
- Housekeeping (Platform & Payload) Temperature, Pressure, Voltage and Current, Statuses
- Science Data (Payload) e.g. Raw or compressed images
Functional Architecture (1/2)

- Functional Architecture
  - definition of
    - **functional break down in**
      - **functional modules** and
      - **functional interfaces** between these modules
### Functional Architecture (2/2)

**On Board Software** > **Architecture** > **Functional Architecture** > **Satellites**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TM/TC</strong> Telemetry &amp; Telecommand</td>
<td>space/ground communications or communications between spacecraft</td>
</tr>
<tr>
<td><strong>HK</strong> Housekeeping</td>
<td>Gathering, filtering and reporting of on board acquired data</td>
</tr>
<tr>
<td><strong>MON</strong> Monitoring</td>
<td>detection of on board events based on ranges or thresholds or trends</td>
</tr>
<tr>
<td><strong>FDIR</strong> Fault Detection Isolation and Recovery</td>
<td>for the on board software and system dependability</td>
</tr>
<tr>
<td><strong>MMGT</strong> Mission Management</td>
<td>for the execution of the mission timeline</td>
</tr>
<tr>
<td><strong>SMGT</strong> Spacecraft Modes Management</td>
<td>handles the on-board system through the different mission phases defines the level of autonomy,</td>
</tr>
<tr>
<td><strong>GNC/AOCS/ACNS</strong> Attitude and Orbit Control Software</td>
<td>Computes the actuators commands from the sensors measurement in order to control the spacecraft attitude and position. Control the propulsion system.</td>
</tr>
<tr>
<td><strong>THERM</strong> Thermal Management</td>
<td>for the temperature control of the spacecraft e.g. through thermal heater lines,</td>
</tr>
<tr>
<td><strong>PWR</strong> Electrical Power Supply Management</td>
<td>distributes the power coming from the battery and the solar arrays manages battery charge / discharge,</td>
</tr>
<tr>
<td><strong>RDP/HRM</strong> Release, Deployment and Pyrotechnic Activation Management</td>
<td>for solar arrays/antenna deployment, for propulsion arming sequence before firing, or for specific needs like shield jettison or spacecrafts composites dispatching,</td>
</tr>
<tr>
<td><strong>OBT</strong> On-Board Time Management</td>
<td>for the synchronization of the on-board clock with ground time,</td>
</tr>
<tr>
<td><strong>ACQ</strong> Data Acquisition</td>
<td>Acquisition of on board data e.g. according to polling sequence table and depending on mode</td>
</tr>
<tr>
<td><strong>SM</strong> Storage Management</td>
<td>for the storage of TM in case of Earth link unavailability</td>
</tr>
<tr>
<td><strong>EQPT</strong> Equipment Management</td>
<td>for the maintain of the equipment table as the reflect of the actual equipment status, and the management of equipment interface,</td>
</tr>
<tr>
<td><strong>RF</strong> Radio Frequency Management</td>
<td>for the communications with ground or with other spacecrafts e.g. through S-Band or X-Band links.</td>
</tr>
<tr>
<td><strong>PL</strong> Payload Management</td>
<td>for scientific payload or additional units (very mission specific)</td>
</tr>
<tr>
<td><strong>CM</strong> Context Management</td>
<td>Saves the context (failure history buffer, GNC states, OBSW patches, equipment table, mission timeline) in case of processor failure.</td>
</tr>
<tr>
<td><strong>ETC ... And Many Others</strong></td>
<td>And many other functions depending on the platform bus and on the mission</td>
</tr>
</tbody>
</table>

Note: interactions between functions are not depicted.

---

ULG - On Board Software Overview 11
Static Architecture (1/4)

- Static architecture deals with
  - Functional Properties
  - Component Model
  - Static Decomposition

- Software is broken down
  - in software components/modules/units/objects
  - with clear interface between them

- Components are organized (see slide 2/4)
  - horizontally in variability layers with coherent abstraction level and
  - vertically in functional chains with coherent functionality

- Component are functional units (see slides 2-3-4/4)
  - That encapsulate functional services,
  - that expose these services to other components through well defined interfaces, and
  - that can be assembled together to build a software product

- Components are made up of (see slide 3/4)
  - Algorithms, State Machines
  - Data Structures

- Main design decision is (see slide 4/4)
  - Central Data Pool vs
  - Distributed Data Flows

- Main programming paradigms are
  - Procedural programming vs
  - Object Oriented programming (not used in on board software)
Static Architecture (2/4)

On Board Software > Architecture > Static Architecture > Generic Structure

Coherent Functionality

December 2011

ULG - On Board Software Overview
Static Architecture (3/4)

COMPONENT
MODULE
UNIT
OBJECT

Functions
Procedures
Methods

Provided Interface

Data Structures

State Machines

Algorithms

Required Interface
Static Architecture (4/4)

Central Data Pool

Distributed Data Flow

VS
Dynamic Architecture (1/3)

• Dynamic Architecture deals with
  • Non Functional Properties
  • Computational Model
  • Dynamic Behaviour

• Main concepts are (see slide 2/3)
  • Tasks, Threads and Interrupts
  • Scheduling and Priorities
  • Communication and Synchronisation

• Main design decision is (see slide 3/3)
  • Periodic/Cyclic/Synchronous/Polling/Time Driven vs
  • Sporadic/Asynchronous/Event Driven
Dynamic Architecture (2/3)

On Board Software > Architecture > Dynamic Architecture > At A Glance

Several interrupts and periodic and sporadic tasks with different periods and priorities leading to possibly complex schedulability issues.

Tasking
- Periodic Tasks

Scheduling
- Periodicity
- Priorities
- Preemption

Shared Resources
- (memory, inputs/outputs, ...)

Shared Executable
- (reentrance of code ...)

Execution Flow

Communications
- Pipes, sockets
- Message queues, mailboxes, ...

Synchronisation
- Semaphores
- Mutexes
- Critical Sections

Interrupts
- Hardware Interrupts
- Software Traps
Dynamic Architecture (3/3)

- Computational Models
  - RMA: Rate Monotonic Algorithm
    Static priority preemptive scheduling. Applies to cyclic jobs. Shorter cycle get higher priority
  - DMA: Deadline Monotonic Algorithm
    Static priority preemptive scheduling. Applies also to sporadic jobs. Shorter deadline get higher priority
  - EDF: Earliest Deadline First Algorithm
    Dynamic priority preemptive scheduling. Process closest to its deadline get highest priority
  - RCM: Ravenscar Computational Model
    Fixed-priority preemptive system with tight restrictions on tasking and synchronisation such as priority ceiling that optimally bound priority inversion, to achieve lock-free mutual exclusion and to avoid deadlocks. Warrants static analysability of the source code and predictability of execution
  - TSP: Time and Space Partitioning
    Hierarchical superposition of two computational models: round robin scheduling of partitions and fixed priority scheduling of tasks within partitions

See also Schedulability Analysis
• Deployment Architecture deals with the Mapping
  • of a logical architecture (software components)
  • to a physical environment (hardware resources).

• Main concepts are
  • Distribution
  • Communication

• Main design decision is
  • Centralised Architecture vs
  • Distributed Architecture

Consider also
Time and Space Partitioning
A deployment architecture depicts the mapping of a logical architecture (software components) to a physical environment (hardware resources).

The physical environment includes the computing nodes, processors, memory, storage devices, and other hardware and communication devices.
OBSW Modeling

- Modeling Exemples
- Modeling Definition
- Modeling Objectives
- Modeling Characteristics
- Modeling Methods
Software has the rare property that it allows us to directly evolve models into fully-fledged implementations without changing the engineering medium, tools, or methods.

The software model may evolve into the system it was modeling in a seamless process.
Modeling: Definition

• What is a Model?

  • A model is a complex structure that represents a design artifact such as a relational schema, and interface definition (API), and XML schema, a semantic network, a UML model or an hypermedia document (Phil. Bernstein. « A Vision of Management of Complex Systems »)

  • A model captures a view of a physical system. It is an abstraction of a physical system with a certain purpose; This purpose determines what is included in the model and what is relevant. Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the appropriate level of detail. (OMG. « UML Superstructure »)

  • A functional specification of the function, structure and/or behaviour of an application or system. (OMG. « MDA Guide »)

  • A model is a simplification of something so we can view, manipulate, and reason about it, and so help us understand the complexity inherent in the subject under study. (Steve Mellor et. All « UML Distilled »)

  • A model is a description of (part of) a system written in a well defined language. A well defined language is a language with a well defined form (syntax), and meaning (semantic), which is suitable for an automatic interpretation by a computer. (Anneke Kleppe et.al. « MDA Explained »)

  • A formal representation of a function, behaviour and structure of the system we are considering. Expressed in an unambiguous language. (Chris. Raistrick et. Al. « Model Driven Architecture and Executable UML »)

  • A simplification of a system built with an intended goal in mind: the model should be able to answer questions in place of the actual system. (J. Bézivin & O. Gerbé. « Towards a precise définition of the OMG/MDA Framework »)

  • Models are an abstraction of the reality captured in a specific representation format i.e. diagram or language.
• Objectives of Software Modeling
  • To deal with complexity of systems development through
    • Abstraction:
      Abstract a problem to focus on some particular points of interest and to improve understandability of a problem
    • Iteration:
      Iterative modeling may be expressed at different level of fidelity
    • Separation of Concerns:
      Possible set of nearly independent views of a model (“Aspect Oriented Modeling”)
    • Domain Specific Language:
      To focus on specific domain expertise
  • To minimize development risks through
    • Through analysis and experimentation performed earlier in the design cycle
    • Enable to investigate and compare alternative solutions
  • To improve communication ...
    • to foster information sharing and reuse!
      A model is often best suited than a long speech!
Modeling: Characteristics

• Characteristics of Useful Models:

  • Abstract
    • Emphasize important aspects while removing irrelevant ones
  • Understandable
    • Expressed in a form that is readily understood by observers
  • Accurate
    • Faithfully represents the modeled system
  • Predictive
    • Can be used to answer questions about the modeled system
  • Inexpensive
    • Much cheaper to construct and study than the modeled system

To be useful, engineering models must satisfy all of these characteristics!
Modeling: Usage

• In an attempt to formalize more and more the expression of the software documentation and production, the design, architecture and requirements have moved from simple text to drawings and from drawings to models.

With the emergence of new tools these model representations can be constructed, translated and exploited in different ways:

• **Analysis**: various types of model checking e.g. completeness and consistency analysis
• **Simulation**: execution of model, behavior can be simulated
• **Design**: decomposition of system in smaller components, establishing interfaces
• **Coding**: automatic generation of source code e.g. C or Ada (auto-coding)
• **Testing**: automatic generation of tests (auto-testing)
• **Proving**: formal verification or proof
Modeling: Methods

• Modeling Methods

  • **UML** (Unified Modelling Language)
    Not a method (weak semantic) but a notation (well defined syntax)
    Can be extended by profiles (e.g. SysML, MARTE, ...)
    Can be complemented/supplemented (e.g. OCL)

  • **AADL** (Architecture Analysis and Design Language)

  • **SDL** (Specification and Description Language)

  • .... and many others

• See also Matlab/Simulink/Stateflow
try to associate the received PDU with one of the existing transactions

create_new_transaction

create_new_receiver

create_new_bit_array

destroy_not_enough_resources

delete_transaction

delete_receiver

create

create

destroy

destroy

In this example, there is still enough memory resources for cf.dp_transaction and cf.dp_receiver but not for the bit array.
State Diagram

On Board Software > Modeling > UML > State Diagram

- link available
- not locked
- locked
- buffer ready
- set to available
- set to unavailable
- non available
- not locked
- locked

- start transfer
- buffer ready
- set to available
- set to unavailable

- stopped
- started
- suspended
- running
- start
- suspend
- resume
- fired
- delay elapsed
- start
- stop

On Board Software > On Board Software Overview
OBSW: On Board Procedures

**OBOPs:** On Board Operation Procedures

- Quite Simple
- Can be considered as Ground Procedures uploaded on board
- Limited interactions with FSW, through TC and TM

**OBAPs:** On Board Application Procedures

- More Complex
- Must be considered as Flight Software
- Tight Interaction with FSW
OBSW: On Board Procedures

- **Source Language**
  - Instruction Set
  - Language Constructs

- **Target Language**
  - Interpreted
  - Pre-compiled
  - Compiled

- **Production Chain**
  - Preparation Environment
  - Execution Environment
  - Management Environment

- **Fault Containment**
OBSW : On Board Procedures

Production Environment
On Ground

- User
- OBCP Syntax
- Configuration Data Base
- OBCP Editor
- Production Program
- OBCP Compiler
- OBCP

Debugging Environment
On Ground

- OBCP Debugger
- DHS Stubs
- OBC Simulateur
- Validated OBCP

Execution Environment
On Board

- OBCP Interpreter
- OBCP Manager
- Data Handling Software
OBSW: Schedulability

- Proof of software schedulability
  - Run Time Testing (how to prove exhaustivity?)
  - Static Analysis (superior to testing)
    - Selected Computational Model (see dynamic architecture)
      - Implementation is assumed to comply to model
      - Mathematical schedulability criteria
    - Software Budget Report (estimated or measured)
      - Processor utilization
      - Worst Case Execution Times

See also Dynamic Architecture and Computational Model
OBSW: Schedulability

- Rate Monotonic Scheduling
- Deadline Monotonic Scheduling
- Earliest Deadline First
- Ravenscar Profile (See Dynamic Architectures)

Schedulability Condition

\[ U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(\sqrt{2} - 1) \]

\[ \lim_{n \to \infty} n(\sqrt{2} - 1) = \ln 2 \approx 0.693147 \ldots \]

Compliance to a selected Computational Model allows for Static Analysis and Formal Check of Schedulability Conditions, based on corresponding Mathematical Model fed by actual Measurement from execution of realistic scenarios.

To this respect, Static analysis is superior to testing, which faces exhaustivity issue in real-scale systems.
OBSW: Verification

• Methods
  • Formal Methods
  • Reviews
    • Peer Review
    • Cross Reading
    • Independent Software Verification and Validation

• Analysis
  • Static Analysis
  • Schedulability Analysis

• Test Campaign
  • Unit Testing (Code Coverage)
  • Integration Testing (Interface)
  • Validation Testing (Functionality)

• Software Validation Facilities
OBSW: Automation

- Automation
  - Formal Model Verification
    - early detection and correction of errors
  - Automatic Code Generation
    - improved productivity
  - Automatic Testing
Centralised Database

- Contains ...
  - On-board software configuration,
  - Constant values, and
  - Parameters useful for ground
  - On-board software configuration, event handling, default housekeeping, default monitoring, patch areas, Drivers commands, TC function ID’s (PUS 8, 1), Memory partitions, File partitions, On-board stores, Default on-board storage allocation, PUS services and sub-services, Types of the parameters in the PUS TC/TM, Error codes, Application ID ...

- Ensure coherency between ...
  - the different outputs

Production Tools

- Support Automatic generation of ...
  - Configuration Files and Documentation

- Allow for easy configuration
Languages

- **C:**
  procedural language, widely used, poor expressiveness but good control, well suited to system programming, efficient code but error prone

- **Ada:**
  strong typing, well suited to embedded and real-time programming, used in launchers

- **C++:**
  object oriented, based on C, less efficient due to object orientation

- **Java:**
  object oriented, interpreted language, under investigation

- **Assembler:**
  low level language, for specific usage
OBSW: Processors

- **Hardened Space Qualified processors**
  - 1990: 1750 – 16 Bits – 2 MIPS
  - 2010: LEON – SPARC V8 - 32 Bits – Cache – Pipeline - 100 Mhz – 84 MIPS
  - Future: NGMP – Quad Core LEON

- **Commercial of the Shelf**
  - e.g. Power PC 1600 MIPS but sensitive to SEEs

See Avionics Overview
OBSW: Operating System

- Real Time Operating Systems
  - VxWorks (Commercial)
  - RTEMS (Open Source)
  - Linux RT (Not widely used ... in OBSW)

- Time and Space Partitionning
  - Hypervisor
  - μKernel
OBSW Reference Architecture

- **Design Pattern**
  - « a repeatable solution ... »

- **Middleware**
  - « a connectivity service ... »

- **Common Services**
  - « shared services ... »

- **Lightweight On Board Application Framework**

- **Layered and Modular Architecture**

---

**Applications**

- AOCS
- POWER
- THERMAL
- ... (PAYLOADS)

**Core Services**

- OBDS
- LDT
- SCHED
- TC
- TM
- HK
- MON
- EVT
- STAT
- HEALTH
- SCHED

**On Board File System**

**PUS services**

**Coms I/F**

**OBCP Interpreter**

**DHS Support Package**

**BSW & RTOS I/F**

- BSP
- RTOS
- Drivers

**Generic Application Services**

**Mission Specific Applications**

**Generic Component**

**Platform Specific Components**
OBSW: Middleware

- SOFA
- ASSERT (PolyORB)
- CORBA LWCCM, HR-LWCCM
- SOA (prospective)

See also Static Architecture and Component Model
OBSW: Building Blocks

- PUS (being revisited)
- SOIS (is this really new)
- OBCP (new standard)
- FDIR (on going study)
- CFDP
- FS
- ...
OBSW: Trends

- **Functionality:**
  Improved on board autonomy, on board data processing, formation flying technologies

- **Distribution,**
  Multiprocessor, Multicore
  Time and Space Partitionning, Integrated Modular Avionics,

- **Standard Architecture**
  Reference Architecture, Application Framework, Middleware, Building Blocks

- **Standard Components**
  Packet Utilisation Standard (PUS),
  Spacecraft On Board Interface Services (SOIS)

- **Methods and Tools:**
  Component Based Software Engineering, Model Driven Engineering,
  Requirement Engineering, Automatic Code Generation and Testing, Formal Verification
OBSW: Engineering

- **Methods**
  - **Requirement Engineering**
    - Formal requirement specification
    - Knowledge Based Engineering
    - Ontology
    - Traceability
  - **Model Driven Engineering**
  - **Formal Methods**
  - **Property Verification**
    - Real-time, performance, functional, dependability
SVF Overview

SVF > Overall Architecture

Simulation Controllers
- Graphical User Interface
- Script Language Interpreter
- Integrated Symbolic Debugger
- Distributed Simulation Interface

Multi Controllers Support

Drivers
- TTM
- SIM
- DAM
- RECU
- MPM

Data Handling Software
- DHS
- PUS
- DHS
- CORE

Applications
- AOCS
- FDIR
- SYS
- MGT
- RF
- PWR

Simulation Core
- Instruction Set
- Simulator Engine
- Processor Registers Simulation
- Memory Simulation
- Marker/
- Breakpoints/
- Coverage
- Discrete Event
- &Time
- Simulator Engine

Communications
- TTM
- SIM
- DAM
- RECU
- MPM

Equipments
- ST
- GPS
- MM
- MT
- RW

Instrum

Environment
- RWM

Ground

Test Conducting Script

OBSW

Generic
SVF Component

Platform dependent
SVF Component

Test Conducting Script
SVF Overall Architecture

- Application Software
- Data Handling Software
- Basic Software

OBSW
TOMS Overview

TOMS> Overall Architecture

Simulation Controllers
- Configuration
- Calibration
- Message Parsing & Formatting
- Synchronization

Multi Controllers Support
- Graphical User Interface
- Script Language Interpreter
- Integrated Symbolic Debugger
- Distributed Simulation Interface

Simulation Core
- Instruction Set Simulator Engine
- Processor Registers Simulation
- Memory Simulation
- Marker/ Breakpoints/ Coverage
- Discrete Event & Time Simulator Engine

Applications
- AOCS
- FDIR
- SYS MGT
- RF
- PWR

Drivers
- DHS
- PUS
- DHS CORE
- OBCP
- APP SVC

Data Handling Software
- TTM
- SIM
- DAM
- REC
- MPM

Communications
- TTM
- SIM
- DAM
- RECU
- MPM

Instruments
- LYRA
- ST
- GPS
- MM
- MT
- RW

Ground Equipment
- RWM
- RWM

Test Conducting Script

Ground