Requirements on Software Frameworks for Space Robotics

14-12-2017 on ROS-Industrial Conference in Stuttgart

Martin Azkarate, Space Automation & Robotics Engineer
Contents

• Why space is different?
  • What are RAMS requirements?
  • The validation process for space systems
• Resource limitations (HW and SW)
• SW implementation: Traditional vs Where we are going
• The approach to Model Driven Software
• TASTE: a framework with V&V of RAMS in mind
• Robotics developments on TASTE (SARGON & ESROCOS)
Why space is different?

- **Cost**: 100k€/kg-of-mass put in space
  - Failures are extremely costly
- **One shot**: Impossibility to fix problems (at least HW)
  - It cannot be repaired as it is far or already dead
- **Remoteness**: Distant from Earth with limited communications
  - Autonomy
- **Environmental**: Structural stress, Thermal constraints, Radiation issues
  - Technology Gap with what available on ground
- **Limited resources**: Energy & Power, Mass & Volume, Connectivity
  - High level of Optimization
- **Confidence**: Cannot afford failure
  - Failure undermines future developments
RAMS Requirements

- **Reliability**: Work consistently (giving correct answers)

- **Availability**: System always responds when requested

- **Maintainability**: Needs to be running throughout the mission lifetime (~years)

- **Safety**: Critical systems need to adhere to highest safety standards

  ➢ Impact in the engineering process
Resource limitations: HW

Terrestrial HW cannot work in space as it is highly disturbed by environment. Space HW is often built from scratch with an expensive and tedious qualification process. Silicon pitch needs to be big to lower radiation sensitivity.

- Available components have limited performance.
  - **Computational**: most advanced OBC processor run on clocks in the order of hundred MHz (monocore).
  - **Memory**: typical RAM values are 256MB, with static allocation.
  - **Sensorial**: cannot use COTS sensors.
  - **Mass & Volume**: cannot embark as much as wanted.

Note the differences between processes of development (qualification) in space and non-space.
Resource limitations: Space systems architecture

- SPARC processor (Leon)
- 50 MHz, 1 core
- Bus: Spacewire, CAN, MIL-1553, RS422
- 256 Mb SDRAM + Flash memory

- RF link
Resource limitations: SW

Similarly, qualification process for space SW is also expensive and tedious.

**Validation & Verification process** of flight code (following **ECSS standards**) imposes constraints to SW in order to create extremely reliable SW.

Every code needs to be checked against:

- **Code coverage**: no dead code
- **Resource starvation**: no dynamic memory allocation
- **Absence of deadlocks**: no states leading to blocked access of shared resource
- **Real-time properties**: verification of worst case execution time
- **Sequencing issues**: what is done in which order

**Validation** process not only w.r.t. functionality but also in terms of **RAMS**
Resource limitations: SW

Limited available SW

- **OS**: RTEMS (and some proprietary in case of big companies)

- **Libraries & device drivers**: few libraries available and drivers need to be coded for the OS in use. We cannot inherit libraries that cannot be validated.

- **Programming languages**: Most of the code in C or ADA, some C++ but following MISRA coding rules.
Traditional software development

- Traditional software development is based on:
  - Lists of textual requirements
  - Manual coding
  - Testing

- In this approach the **V&V of SW depends on massive amount of work** performed by skilled and experienced individuals.

- But with the development of larger and more complex multi-domain systems that contain safety requirements and heterogeneous components, the development approach has to be better. We do not have the time and the people to cope with the growing complexity.
How do we solve this?

How to put everything together?

- SDL
- Python, Tcl
- Simulink, Scade
- VDM
- VHDL
- Ada
- C
- SMP2

- state machines
- test scripts
- control laws
- algorithms
- execution trace
- FPGA code
- drivers, user code
- applicative code
- system simulation
- system simulation
The solution

A SW development framework needs to be thought and designed with **V&V and code generation in mind** and **provide the tools** to validate the product.

**Note**: the framework is not space qualified flight SW but gives the tools and processes for building space applications.

ESA has chosen to employ a **model based** approach to base the SW development. **TASTE** is a set of tools and processes, implementing model based SW engineering, to target SW complexity and to ensure **consistency** during the whole development lifecycle.

ESA built TASTE based on **free open-source** software, as an exploration platform putting together the state-of-the-art in various software technologies.
The approach to model driven SW

Modelling languages have to be used with a goal in mind:

- Capture the system architecture to analyse the system feasibility
- Capture data types (ranges, units) to ensure consistency
- Capture the software expected behaviour (state machines, algorithms) and let tools explore this behaviour to verify properties
- Automate the production of code, documentation and testing

Unfortunately modelling is often misused and results in more troubles than visible benefits

Benefits come at a price:

- a solid process (methodology/workflow) is as important as good tools
- Training counts
How do the tools look like?

Graphical approach to unambiguously capture the system architecture and its real-time properties.
How do the tools look like? (cont.)

Tools to describe state machines – they are complex, and capture the core of the system behaviour
Simulation and interaction with the user
TASTE Software factory - philosophy

Use existing technologies – glue them together when semantics are compatible

Software modelling is not new: learn, use and build on top of languages that are mature and widely used in other industries (AADL, ASN.1, SDL, Simulink)

Let application designers choose the technology that is the most appropriate for each purpose – don’t try to code drivers in UML!

Automate everything that can be

Code generators that translate formal requirements into reliable code

Develop tools that make the life of developers easier – keep the right balance between abstraction and concrete implementation. Both count!

The target is to implement SW and Systems, the models are just a mean to it!
What is SARGON

**SPACE AUTOMATION & ROBOTICS GENERAL CONTROLLER**

Activity to design a **Robot Control Operating System** (RCOS) with special care on **RAMS requirements** and **re-usability** at different implementation phases.

Generic schematic representation of a RCOS.
SARGON – Why

To Reduce the Gaps Between Different Robotics Developments

- In **Space**, ExoMars or European Robotic Arm (ERA) require significant software engineering effort compared to other satellite missions, due to complexity and low heritage. SW tools used are highly customized -> very little percentage of this software development, validation and verification effort becomes re-usable.

- In **Industry**, commercial closed source solutions customized for their HW/products. Reliable, but far from getting to standardization, being portable or achieving inter-operation.

- In **Academia**, the trend is to use robotics SW frameworks (such as ROS), which allow fast prototyping and testing for R&D activities but far from RAMS-compliant -> Transitions to reliable applications require extensive recoding.
SARGON – How

TASTE – A MODEL-DRIVEN APPROACH

• SARGON builds on the TASTE system, a model-driven approach for the development of reusable and RAMS-compliant on-board software.

• SARGON focuses on the analysis and definition of requirements for an RCOS and on identifying the building blocks to complement TASTE for covering all needed RCOS functionalities.

• The aim of this RCOS is to be the base of future European space robotics applications, and in this context SARGON is a first step in that direction.
Programmatic

SARGON is an activity funded by the ESA Basic **Technology Research Programme** (TRP). The activity was started in January 2016 and is planned to conclude in December 2017. -> http://www.sargon-project.eu/

While the activity is ongoing, it has already been successful in securing a continuation. The **EC H2020** programme, within the Strategic Research Cluster in Space Robotics, has awarded an operational grant to the **ESROCOS** team for the further development of **SARGON**. -> http://www.h2020-esrocos.eu/
Questions?

Acknowledgements to:
ESA and industrial partners

END
Relevant RCOS functionality additions of SARGON to TASTE

Robotics Data types
Component life-cycle
URDF modelling of Kinematics
Logging and visualizing of complex data types
Runtime system configuration and introspection
Software development processes

- ECSS: the European Cooperation for Space Standardization (www.ecss.nl)
- Engineering and Quality standards for all aspects of space systems
- Software standards:
  - ECSS-E-ST-40C: Space Engineering - Software
  - ECSS-Q-ST-80C: Space software Quality Assurance
- Milestone-based with detailed data packages to be produced by industry
  - SRR : System Requirement Review
  - PDR : Preliminary Design Review
  - CDR : Critical Design Review
  - QAR/FAR : Qualification and Flight Acceptance Review