Application of modeling techniques for on-board satellite applications

Óscar Rodríguez Polo and Pablo Parra

ZR25

LangDev Conference 2024

Seville

17-19 October 2024

Summary

- Introduction
- Component-based software design modeling
    - Automatic EC++ code generation
- Model-Driven Software Validation and Verification process
    - Automatic generation of validation tests
    - Configuration control and deployment
    - Verification of temporal constraints
- Domain-specific programming languages

Introduction

SRG

Space Research Group (SRG-UAH)

On-Board software development

2004

Nanosat 01

National Institute of *Aerospace Technology (Spain)*

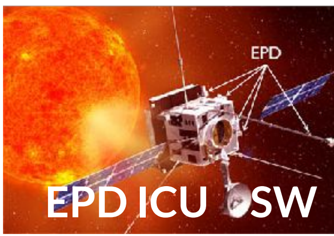
INTA

2009

Nanosat 1B

2020

Solar Orbiter



esa — NASA

Introduction

Space Research Group (SRG-UAH)

On-Board software development

2004

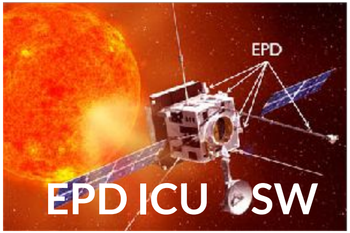
Nanosat 01

National Institute of Aerospace Technology (Spain)

2009

2020

Solar Orbiter



Modeling Techniques: MBSE-> MDE

Introduction

Space Research Group (SRG-UAH)

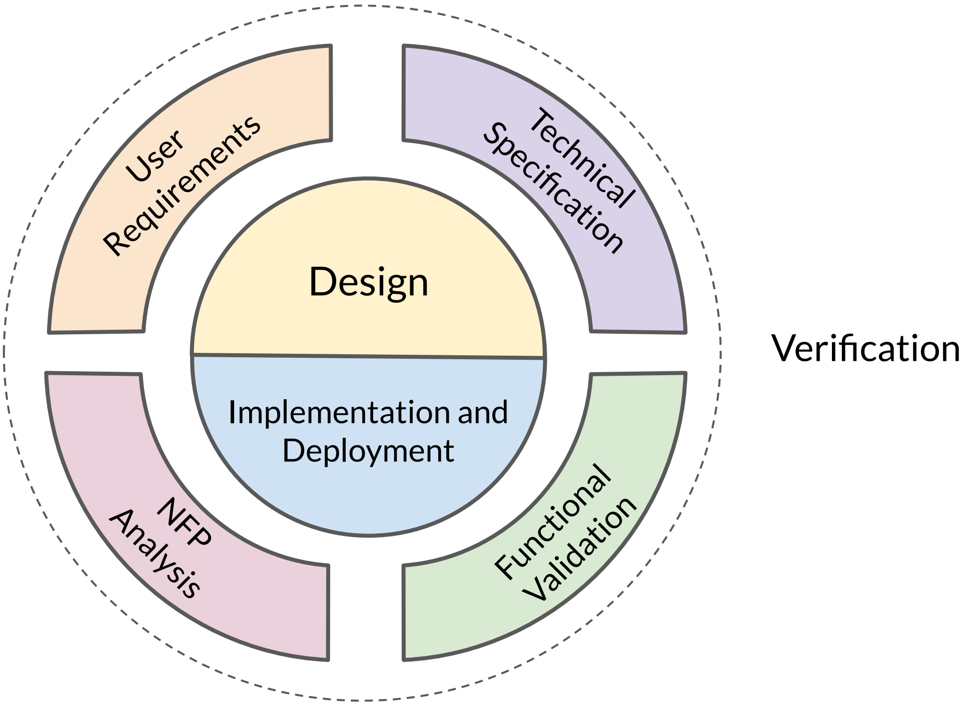
On-Board software

Embedded Software
Cross Compiled (Platform Config Control),
Low Memory Footprint (C or Embedded C++)

Real-Time Software
Processor load, deadlines (schedulability analysis)

Deterministic Software
No dynamic task creation or memory allocation

Introduction

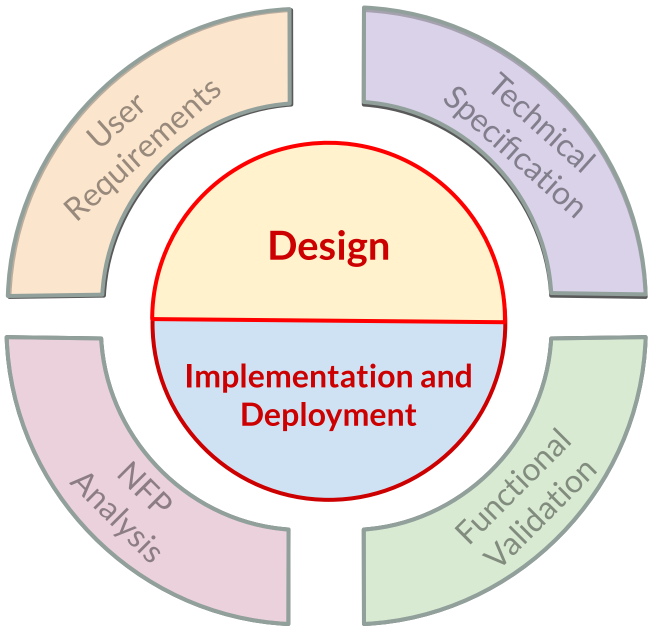


Component-based software design modeling

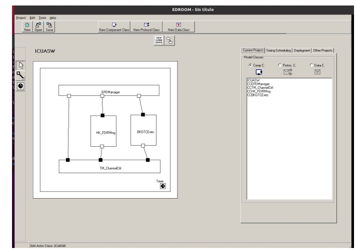
2001-**2004**

Nanosat 01

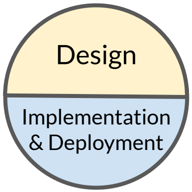
Component Based Design Modeling & Automatic Code Generation



EDROOM

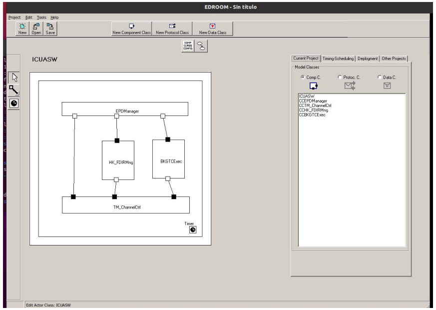


Component-based software design modeling

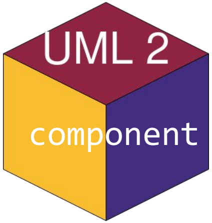


Component-Based Design Modeling & Automatic Code Generation

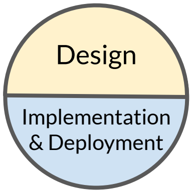
EDROOM



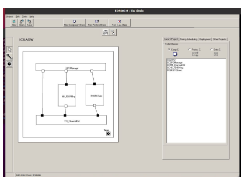
Based on ROOM
(Real-Time Object-Oriented modeling)
Bran Selic



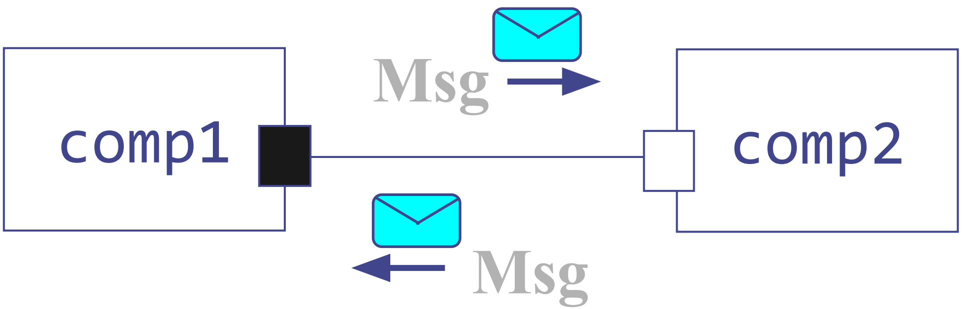
Component-based software design modeling



EDROOM

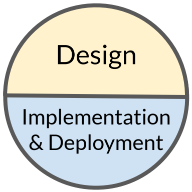


Communication based on message passing through ports



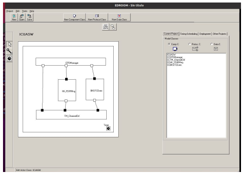
ROOM Model

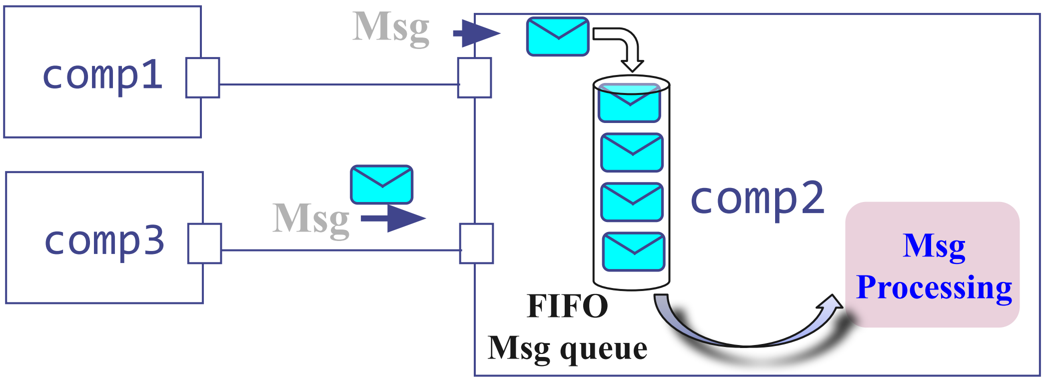
Component-based software design modeling



Component = Message Processor

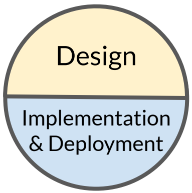
EDROOM



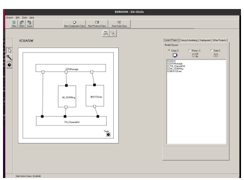


ROOM Model

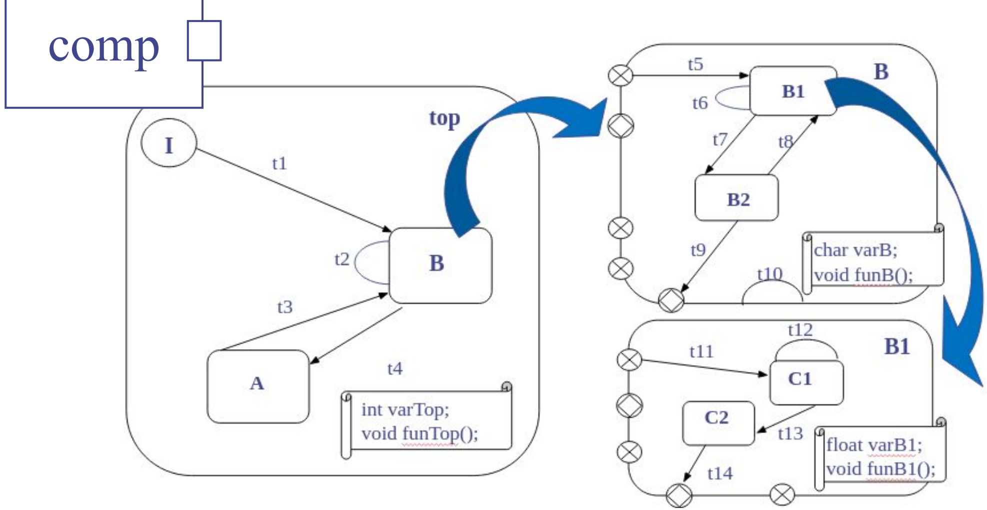
Component-based software design modeling



EDROOM



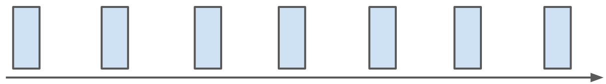
Hierarchical State Machine Behaviour

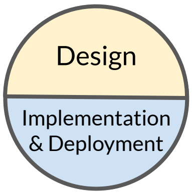


ROOM Model

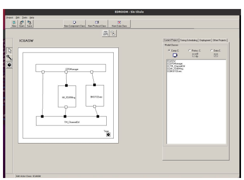
Component-based software design modeling

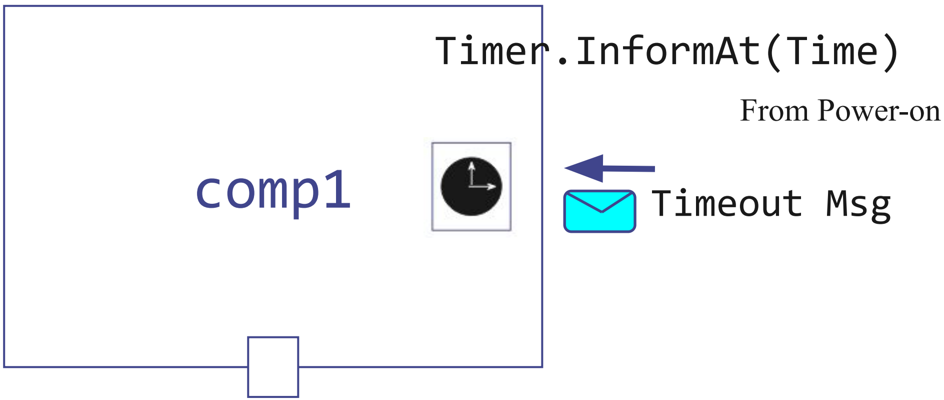
Task Periodic execution





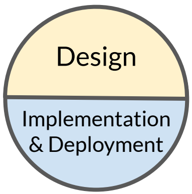
EDROOM



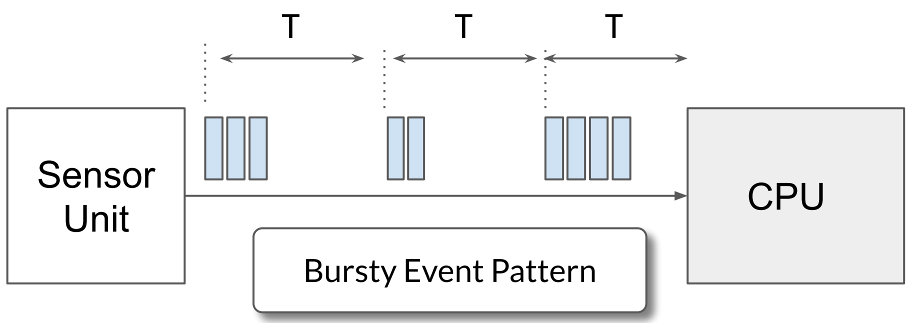


ROOM Model

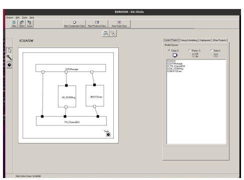
Component-based software design modeling



Events bounded but not periodic

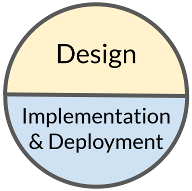


EDROOM

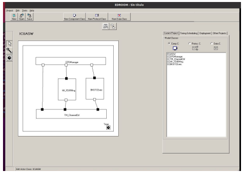


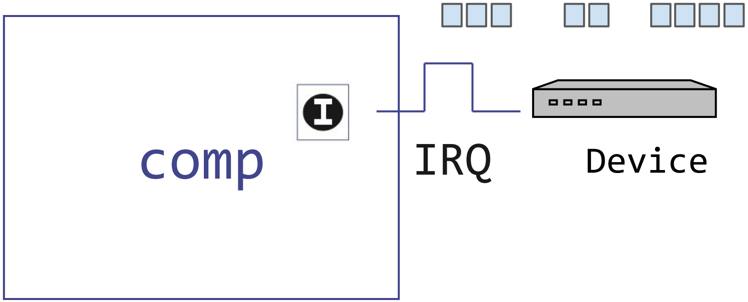
ROOM Model Extension

Component-based software design modeling



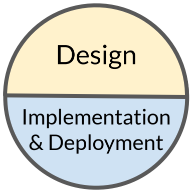
EDROOM



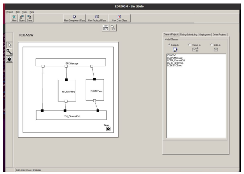


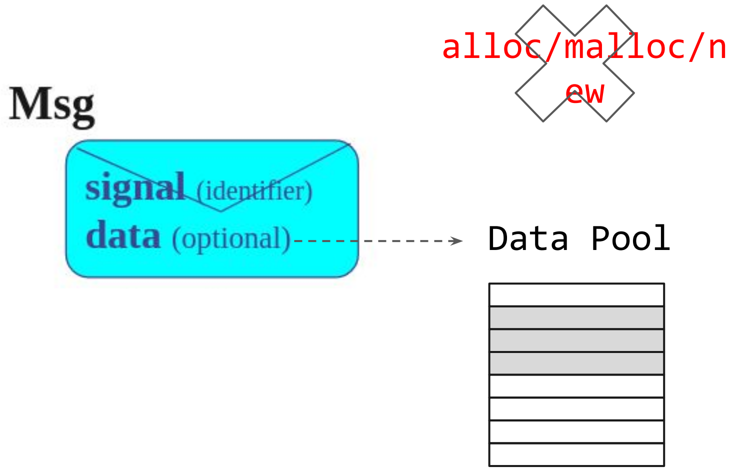
ROOM Model Extension

Component-based software design modeling



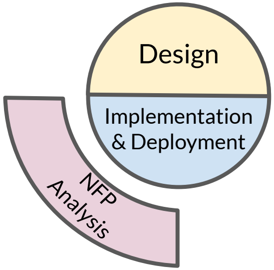
EDROOM





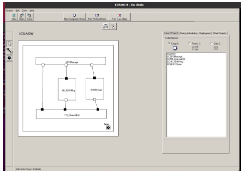
ROOM Model Extension

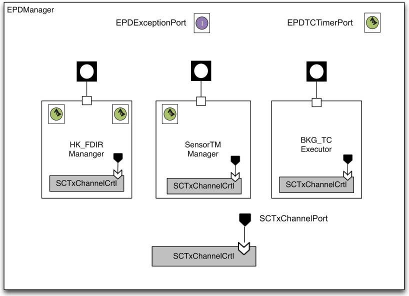
Component-based software design modeling



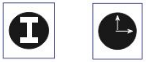
EDROOM Model Key Aspect

EDROOM



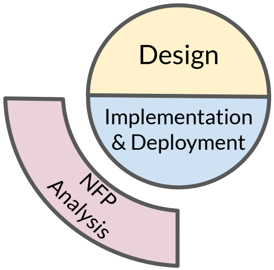


Source of Events

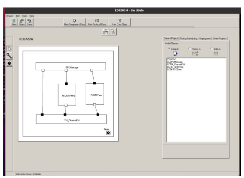


Bounded Response to Events

Component-based software design modeling

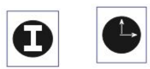


EDROOM



EDROOM Model Key Aspect

Source of Events



EDROOM Model

MAST Model

Bounded Response to Events

MAST: **Modeling and Analysis Suite for Real Time Applications**

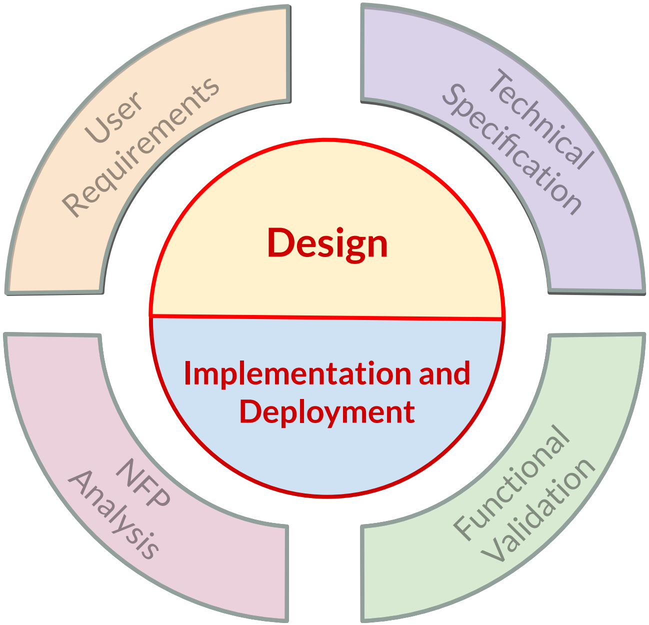
http://mast.unican.es/

Schedulability Analysis!

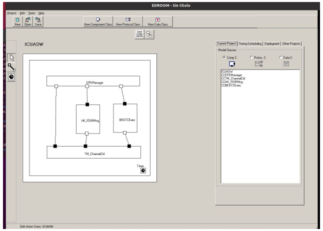
Component-based software design modeling

2006-**2009**

Component-Based Design Modeling & Automatic Code Generation



EDROOM

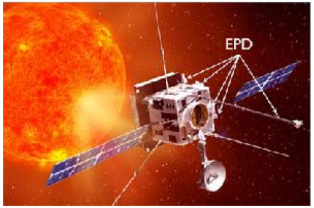


Oscar R. Polo, Pablo Parra, et. al. 2012. Component based engineering and multi-platform deployment for nanosatellite on-board software. In *DASIA 2012 - DAta Systems In Aerospace*

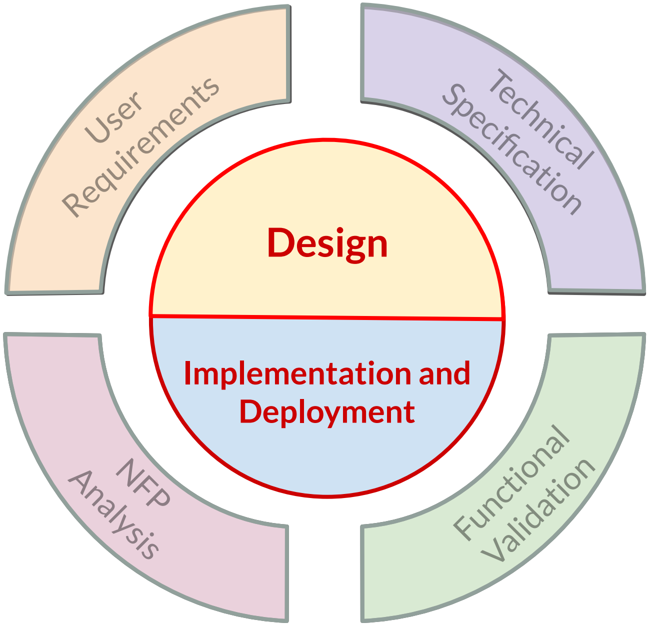
Component-based software design modeling

2006-**2009**

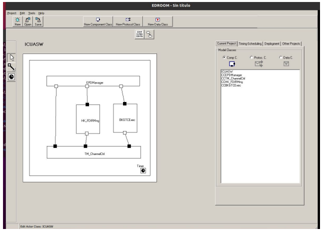
Solar Orbiter



Component-Based Design Modeling & Automatic Code Generation



EDROOM

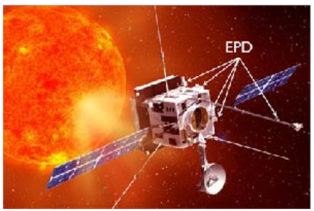


Sebastián Sánchez, Manuel Prieto, Óscar R. Polo, Pablo Parra, et. al. 2013. HW/SW Co-design of the Instrument Control Unit for the Energetic Particle Detector on-board Solar Orbiter. *Adv. Space Res.* 52, 6 (September 2013)

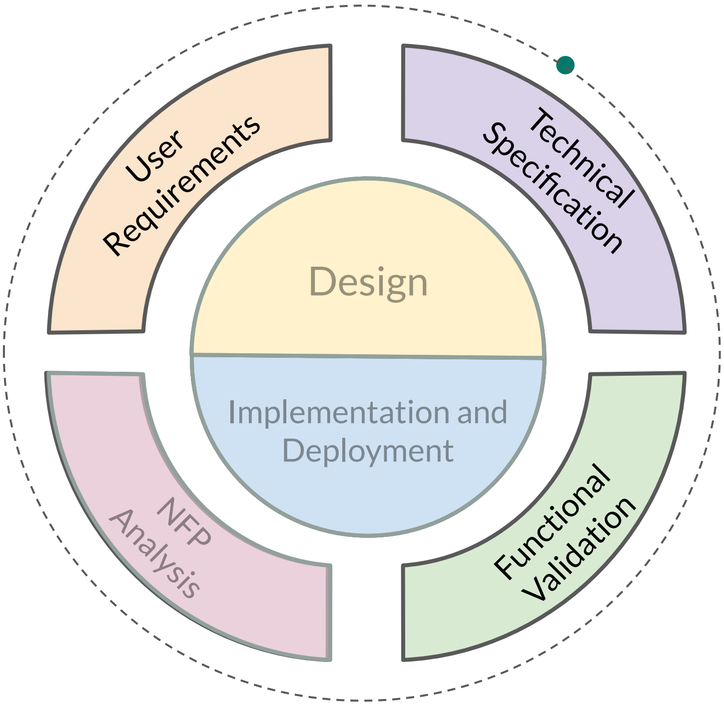
Validation and Verification process

2009-**2020**

Solar Orbiter



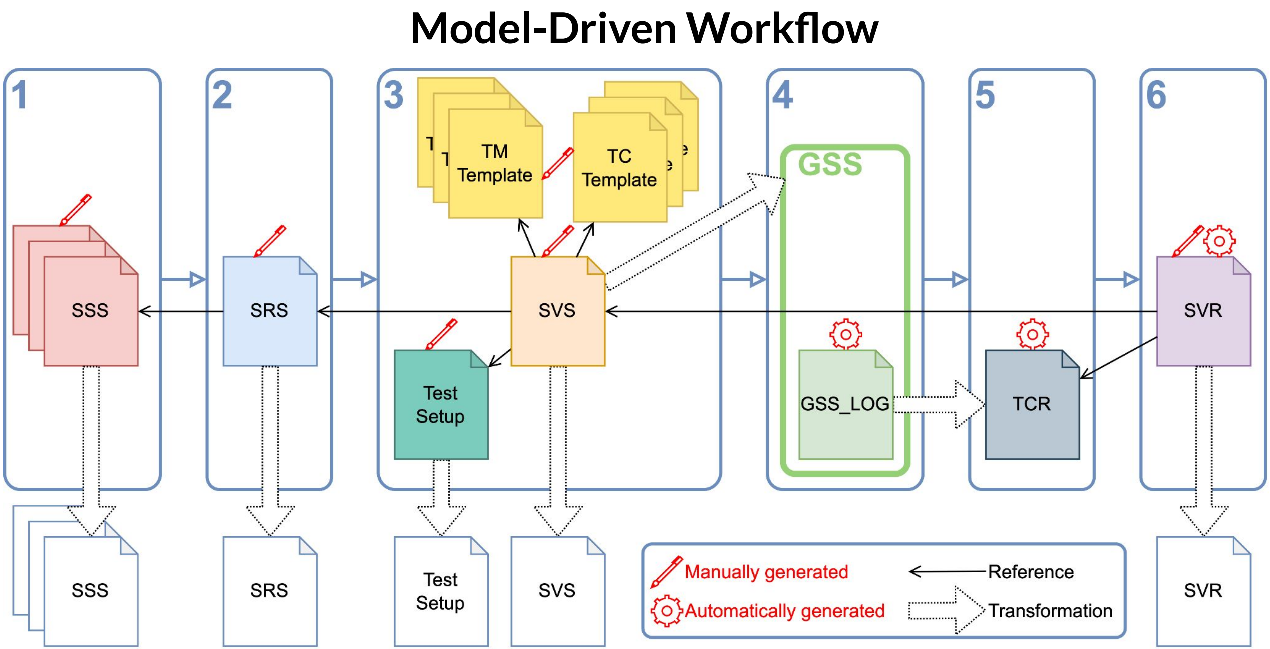
Model-Driven Software Validation and Verification (software system-level)



ECSS Standards
On-board software
ESA missions

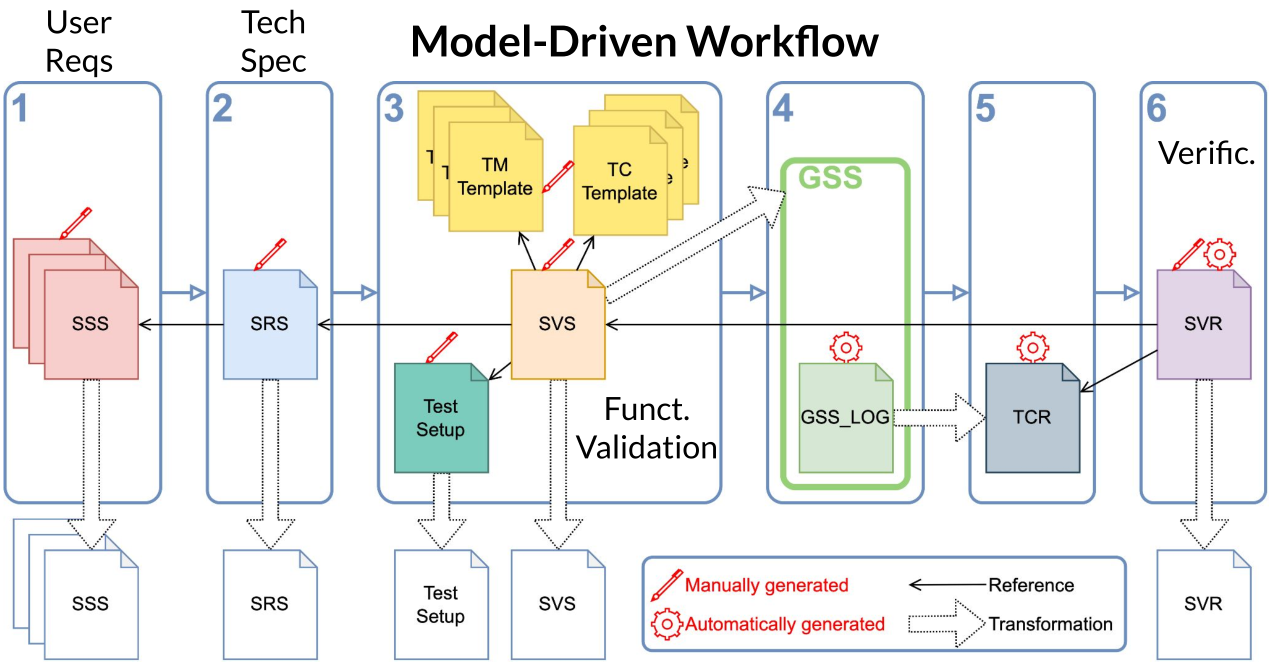
Verification

Model-Driven Validation and Verification process



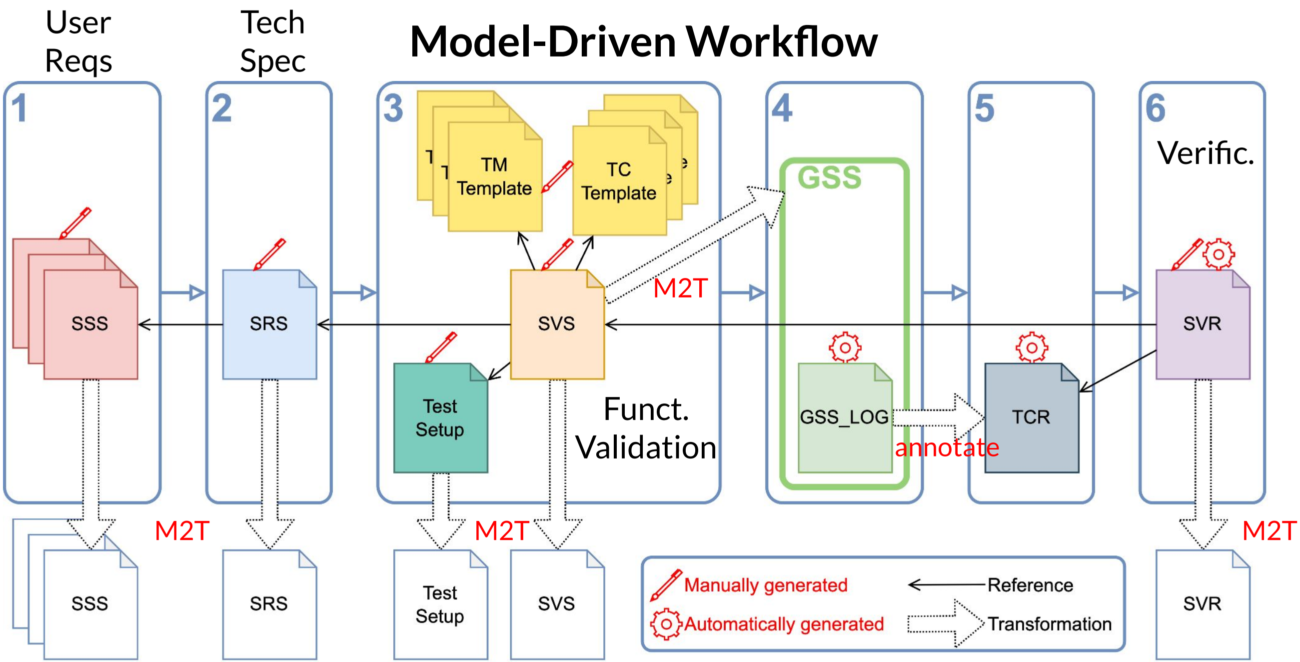
Aarón Montalvo, Pablo Parra, Óscar Rodríguez Polo, et. al. 2021. Model-Driven system-level validation and verification on the space software domain. *Software and Systems Modeling* (November 2021)

Model-Driven Validation and Verification process



Aarón Montalvo, Pablo Parra, Óscar Rodríguez Polo, et. al. 2021. Model-Driven system-level validation and verification on the space software domain. *Software and Systems Modeling* (November 2021)

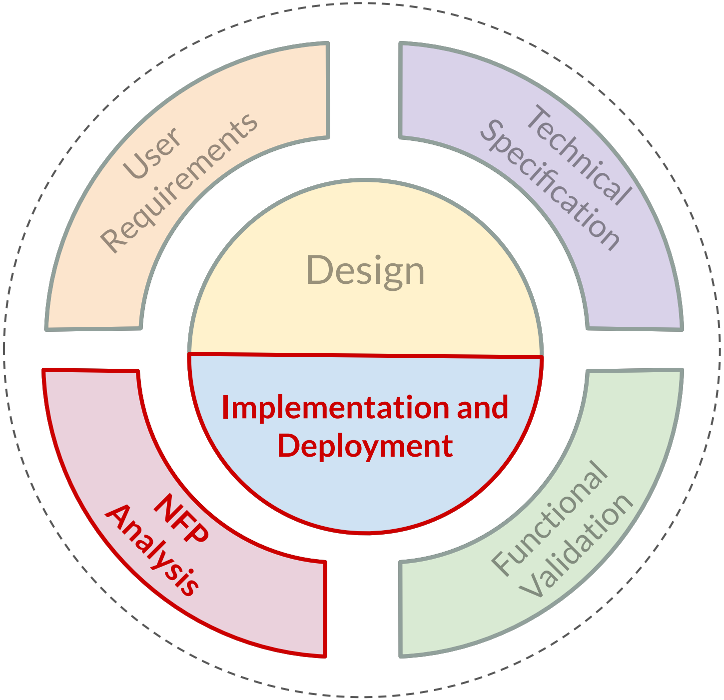
Model-Driven Validation and Verification process



Aarón Montalvo, Pablo Parra, Óscar Rodríguez Polo, et. al. 2021. Model-Driven system-level validation and verification on the space software domain. *Software and Systems Modeling* (November 2021)

Model-Driven Validation and Verification process

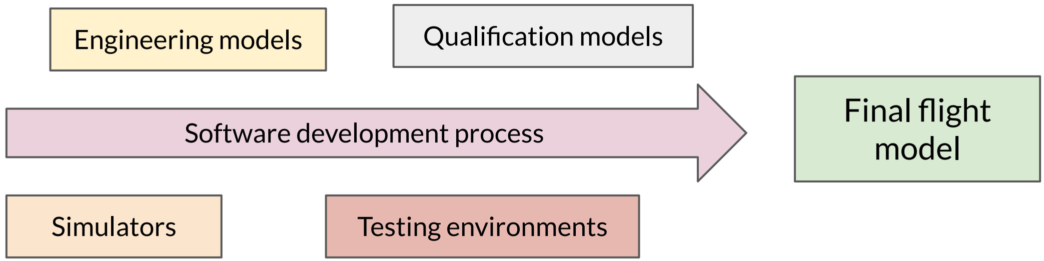
ECSS Standards
On-board software
ESA missions



Verification

Model-Driven Validation and Verification process

- Software development projects for on-board satellite systems face numerous challenges
  - Hardware and software are usually developed in parallel
  - The engineering process generally involves the use of different configured deployment platforms, including testing environments and simulators



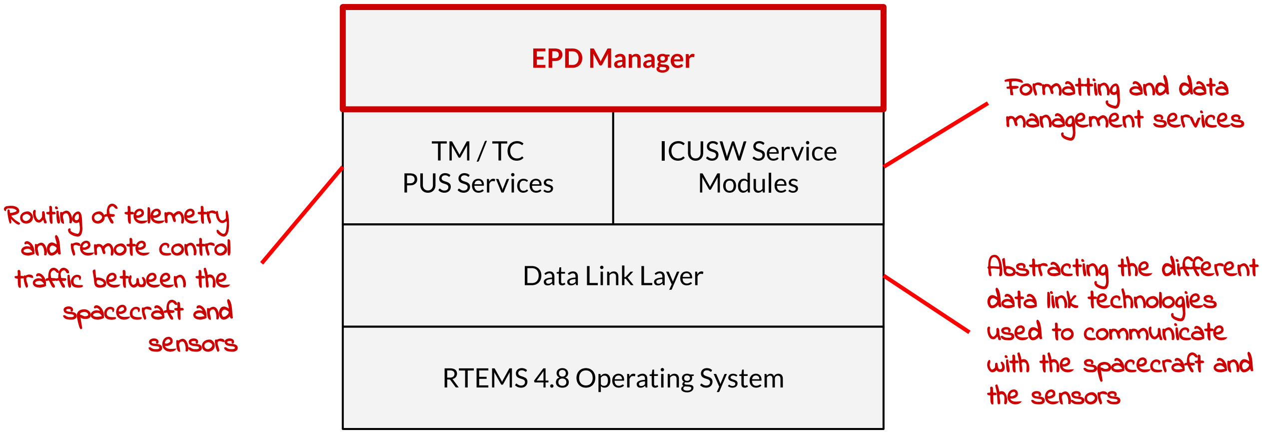
We need to manage software variability

Model-Driven Validation and Verification process

- The Space Research Group developed the MICOBS framework
  - Model-based software development framework
    - Implemented using the Eclipse Modeling Framework (EMF)
  - Includes the platform as a design variable
- MICOBS provides support at two levels:
  - Configuration and deployment of on-board software applications
    - Software variability management facilitating its parameterization
  - Component-based software development
    - Integration of component technologies and analysis tools
- It has been successfully applied in the development of the application software of the Instrument Control Unit (ICU) of the Energetic Particle Detector (EPD) on-board Solar Orbiter

Use case: application software of the ICU of EPD

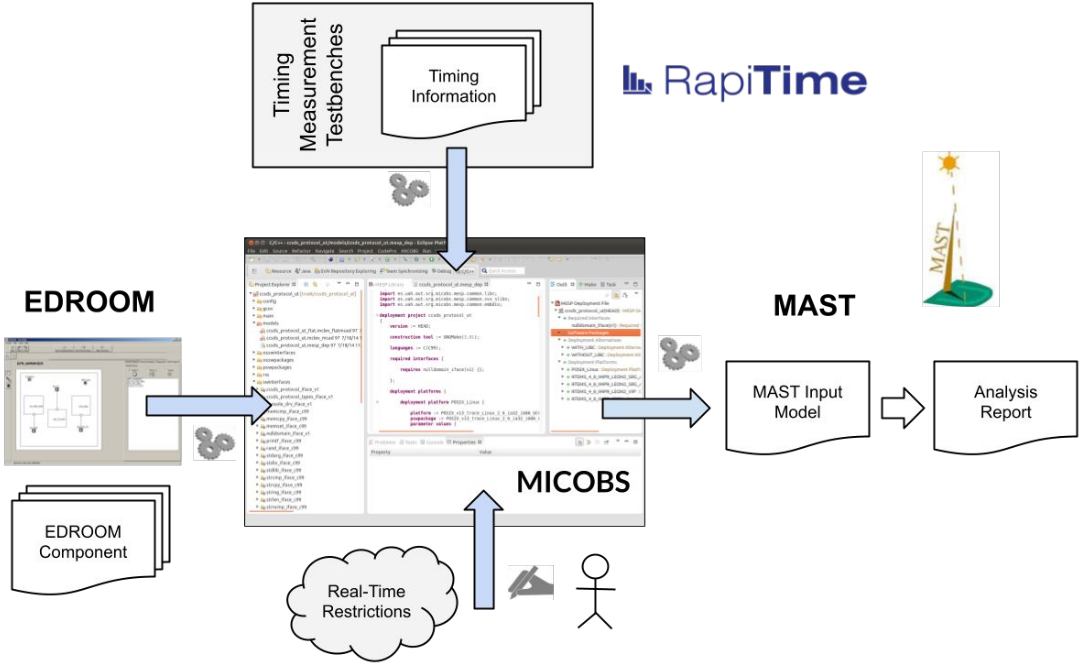
- The application software of the Instrument Control Unit (ICU) of the Energetic Particle Detector (EPD) is structured in different layers



- Each of these layers has been modelled as a **collection of software packages**

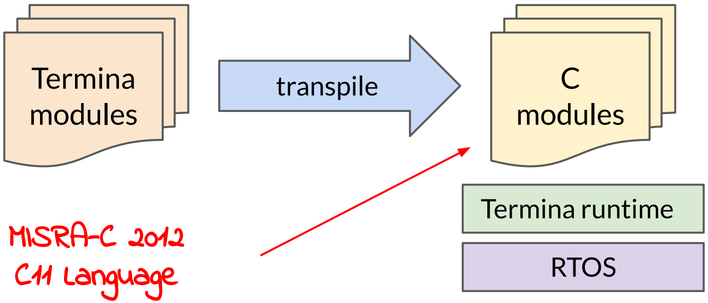
System-level analysis of the application software of EPD's ICU

Integration with the EDROOM component model



The `Termina` programming language

- It is a **domain-specific strongly-typed imperative programming language** that is specifically targeted for embedded real-time critical systems
- A *transpiler* is being implemented to translate `termina` code into MISRA-C



MISRA-C 2012
C11 Language version

The generated code is executed on top of a portable runtime environment

An initial implementation of the Termina runtime is being developed for RTEMS

Design principles

Ease of learning and use

Follows the imperative programming paradigm with C-like syntax to facilitate adoption by developers familiar with C, C++, and Ada.

Reactive programming model

Definition of reactive entities that respond to global events, enhancing application analysis and verification

Run-to-Completion Semantics

All actions complete without blocking, simplifying verification procedures by **enforcing terminating functions**

Safety and robustness

Interaction mechanisms with enforced separation of concerns and controlled hardware access

Deterministic Dynamic Memory Management

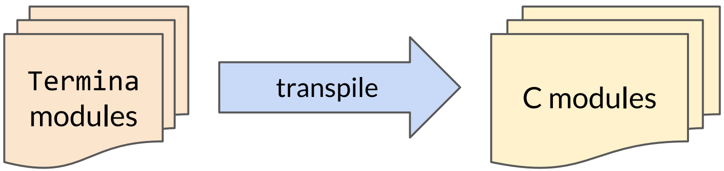
Utilizes memory pools to ensure memory management results are independent of system state

Memory-Safety Properties

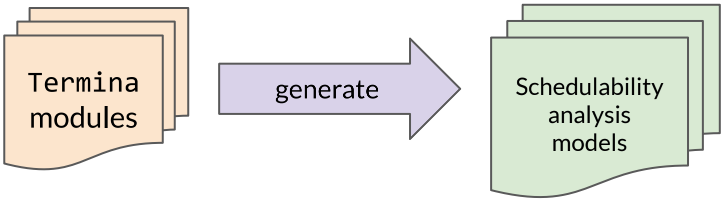
Guarantees absence of race conditions or deadlocks when accessing shared resources, plus null-pointer dereferences and memory leaks

Development status

Software development for real-time systems

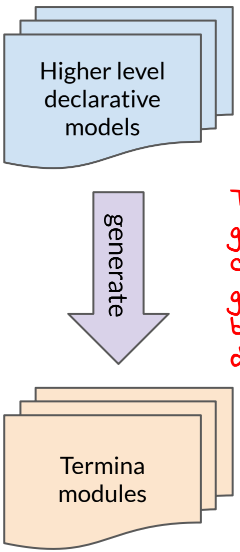


Generation of analysis models



Generation of other products?

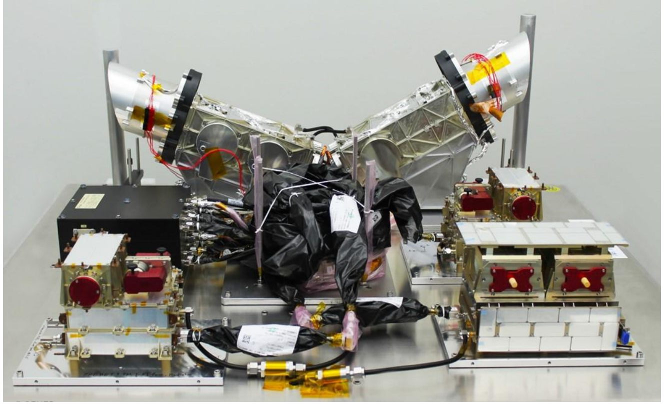
Use `Termina` as a pivot language



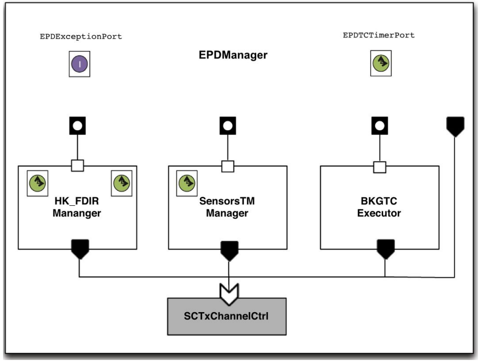
The generated code provides guarantees by language design

Main use case

We are currently re-implementing a (symplified) version of the application software of the instrument control unit of EPD



Qualification Model (QM) of the Energetic Particle Detector Instrument



Component architecture of the application software

Future lines of work

- Update the language to support new features:
  - Generic programming
  - Support for Symmetric Multiprocessing Systems
- Use of static analysis techniques to increase safety
  - Verification of the absence of run-time errors, such as out-of-bounds indexing in the generated code
- Look for collaboration opportunities!

Thank you very much for your attention

Any questions?

ZR25