Application of modeling techniques for on-board satellite applications

Configuration control and deployment and verification of temporal constraints

Pablo Parra

I-MDE-A: WORKSHOP ON MODEL-DRIVEN ENGINEERING AND ITS APPLICATIONS

IMDEA Software Institute

May 16, 2023

Universidad de Alcalá

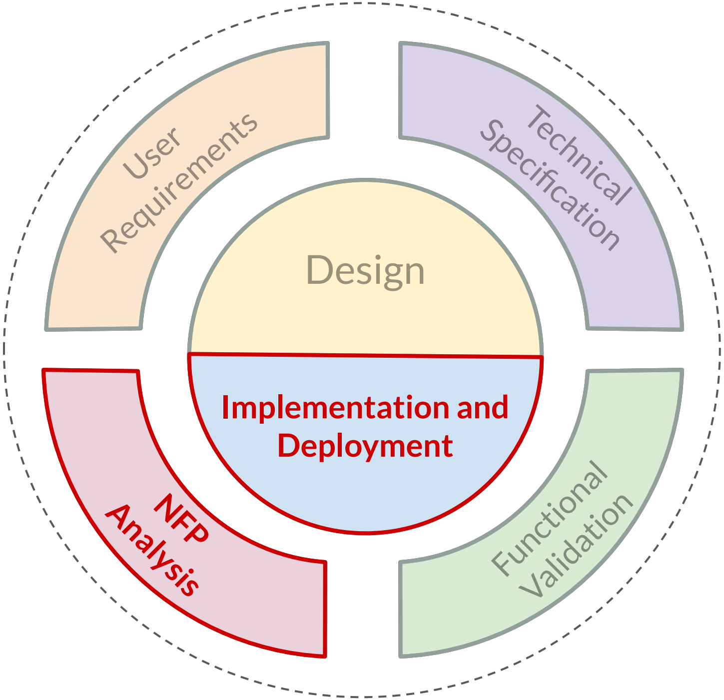
SRG

Summary

- Introduction
  - Platform models
- Configuration control and deployment
  - Software packages and interfaces
  - Embedded software projects
- Verification of temporal constraints
  - Component model and analysis tool integration
  - System-level analysis of the application software of EPD's ICU
- Conclusions

Introduction

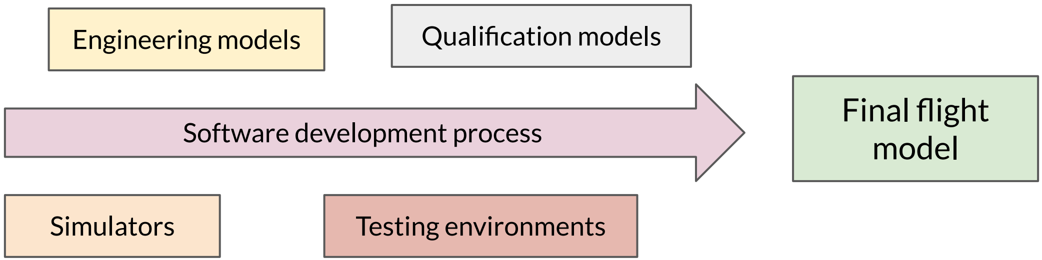
ECSS Standards
On-board software
ESA missions



Verification procedures

Introduction

- 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

Introduction

- Different strategies and design methodologies can be used to reduce software development times
- Component-Based Software Engineering (CBSE) is a powerful methodology for developing embedded software systems
  - There are different component models and technologies
- In embedded software systems there are both functional and non-functional requirements
  - Non-functional requirements are associated to design metrics such as memory size, power consumption or response times
- In many cases, the analysis of the non-functional properties of the system can be performed by composition from the properties of individual components

Introduction

- Several component models provide support for compositional analysis
  - They allow annotations to be made on the components
  - These annotations represent the property values for a given element
  - There are different tools that can be used to perform the analysis
- The annotation mechanisms are component technology-specific
  - There are multiple component technologies that lack of any support for property annotation
- There might be a relatively large distance between the annotated data and the analysis tools

Introduction

- The Space Research Group of the University of Alcala has participated in numerous on-board satellite software development projects:
  - SOlar and Heliospheric Observatory (SOHO)
    - CDPU of the CEPAC instrument
  - INTA's Nanosat and Microsat Programmes
    - NS-01 & NS-1B On-board software development
    - MS Flight software and OBDH electronics
  - Euclid's Near Infrared Spectrometer and Photometer (NISP)
    - Boot software
    - CDPU
  - **Solar Orbiter's Energetic Particle Detector (EPD)**
    - **Instrument Control Unit**
    - **Flight Software**

Introduction

- 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

Platform models

- The MICOBS framework defines the platform as the tuple

$$P = (osapi, os, architecture, compiler, microprocessor, board)$$

where:

  - ***osapi***: Application Programming Interface (API) provided to the OBSW by the operating system (e.g. POSIX)
  - ***os***: Operating System that supports the OBSW (e.g. RTEMS)
  - ***architecture***: Instruction Set Architecture (ISA) implemented by the microprocessor that will run the OBSW (e.g. SPARC)
  - ***compiler***: Compiler that is used to build the final executable image of the OBSW for this particular platform (e.g. GCC compiling toolchain)
  - ***microprocessor***: Microprocessor that implements the architecture on top of which the OBSW is to be executed (e.g. LEON2)
  - ***board***: Processor Module board (or simulation environment) used during the OBSW development process

Platform models

- Platforms and their constituent elements may have attached a set of configuration parameters
    - Specify possible variations associated with a given platform, including the construction process of the final executable image
- Platform models are defined using a textual syntax
    - They are version-tagged and configured separately in a repository using a version control system

```
platform CDPU_EM_1_0 {

  version := 1.0;

  osapi := RTEMSAPI(4.8);
  os := RTEMS(4.8);
  architecture := SPARC(8.0);
  compiler := GCC(4.2);
  microprocessor := LEON(2.0);
  board := CDPU_EM(1.0);

  configuration parameters {
    string RTEMS_BIN_PATH :=
        "/opt/rtems-4.8/bin";
    string EXTRA_LDFLAGS := "";
  };
};
```

Configuration control and deployment

Software packages and interfaces

Main goal

Definition of a code organization that facilitates the configuration and reuse of the on-board software for the different configured deployment platforms used during the development process

- The code is organized into ***software packages*** and ***interfaces***
  - A **software package** is a piece of software that can be configured, compiled and linked together with other software packages to form an on-board application
  - **Software interfaces** represent the services required and provided, at the implementation level, between the different software packages
  - The set of interfaces provided by a package is fixed, but the set of required interfaces may depend on the different supported platforms

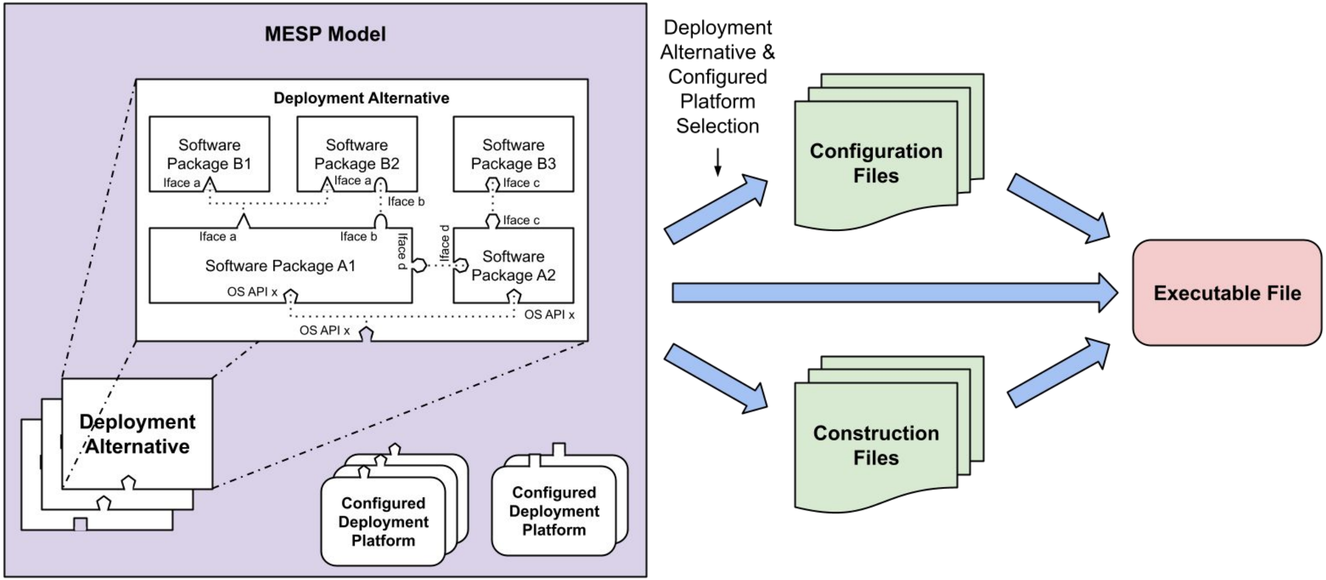
Configuration control and deployment

Software packages and interfaces

- Two special types of packages:
  - Operating system software packages
  - Platform software packages
- Software packages may define configuration parameters
  - Configuration parameters can be declared as platform-dependent or platform-independent
- Software package models also allow specifying **quantifiable resources**
  - Resources that are provided and demanded by the packages
- For each project a Multiplatform Embedded Software Project (MESP) model is defined
  - Declares all the software packages that are part of the application
  - It is organized in different **deployment alternatives**

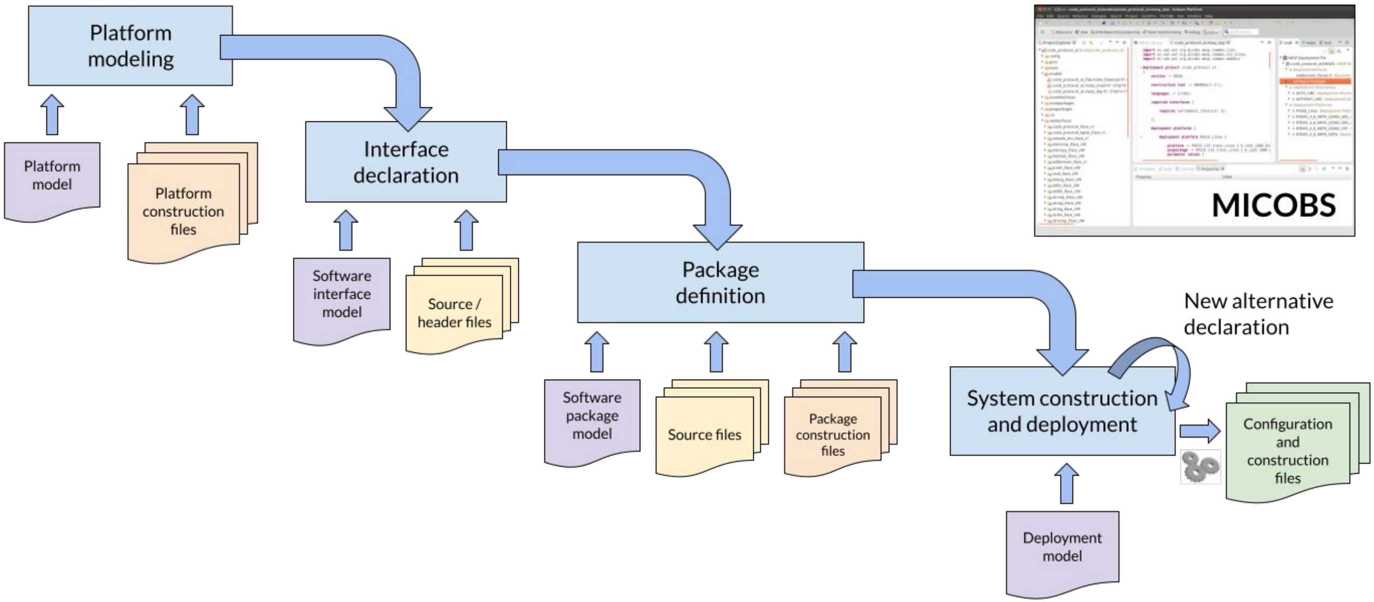
Configuration control and deployment

Embedded software projects



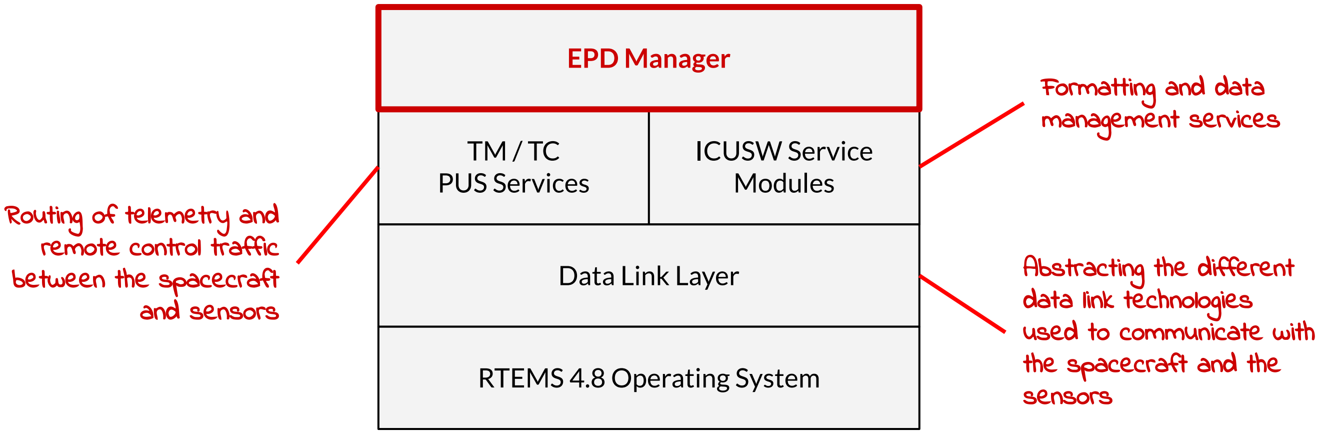
Configuration control and deployment

Embedded software projects



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**

The EDROOM component model

- EDROOM is a component-based graphical modelling and automatic C++ code generation tool inspired in ROOM
  - It had been successfully used by our team to develop the on-board software of Nanosat-01 and Nanosat-1B satellites
  - It was selected for the development of the on-board application software of EPD's instrument control unit
  - It uses a variant of the statecharts of Harel, called ROOMCharts, for defining the hierarchical behaviour of the components
    - States and transitions between them
    - Messages that trigger the transitions
    - The actions associated to the transitions

EDROOM does not provide any annotation capabilities

Component model and analysis tool integration

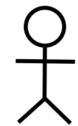
- MICOBS allows defining **differentiated analysis models** to annotate the extra-functional properties
  - Following a model-driven engineering approach, these analysis models are transformed into other models
  - Eventually, one of these models can be used as input for a tool with which to perform an analysis of the entire system
- Each property is associated with **one or more analysis models**
  - Models can be defined to be semantically closer to the ones that are finally used for the analysis
- Models can be reused for different component technologies
  - Mappings shall be defined between the computational model of the component technology and the elements of the analysis model
- Models are ***platform-aware***
  - The mapping of the extra-functional properties shall be done in accordance with the platforms on which the components can be deployed

Component model and analysis tool integration

- Definition of component domains
  - Component model
  - Semantical restrictions
  - Transformations
  - Domain-specific models

MICOBS Component Meta-model

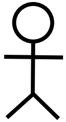
Framework Architect



- The integration enables the use of several MICOBS features
  - Component Common Representation (CCR)
  - Multi-platform Component Architecture and Deployment (MCAD)

Component model and analysis tool integration

Framework Architect



- Definition of System Analysis Models (SAMs)
  - Analysis Oriented Models (AOMs)
    - Element models
      - Component-level models
      - System-level models
    - Platform models
- SAMs are component-technology independent
  - Auxiliary models can be used to map the SAMs to the component technologies
    - Component auxiliary models
    - Architecture auxiliary models

Component model and analysis tool integration

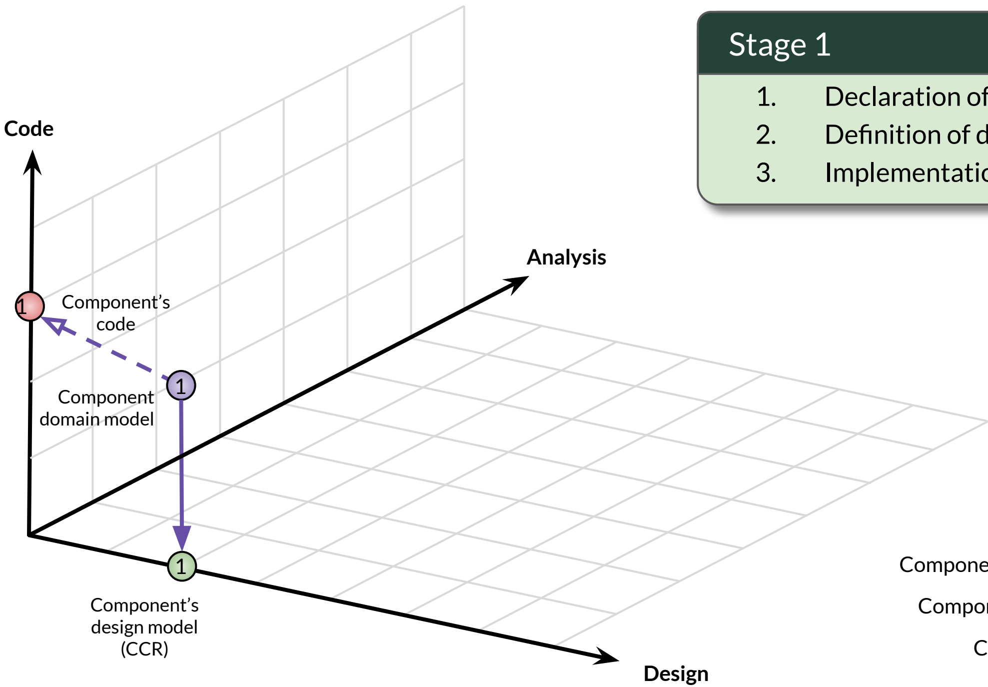
Main goal

To enable the generation, by means of transformations, of the input files of the analysis tools

- AOMs can include information that depends on deployment platforms and component configuration attributes
  - AOMs have to be annotated with the final values for each platform and configuration
    - Static property annotation
    - Dynamic property annotation

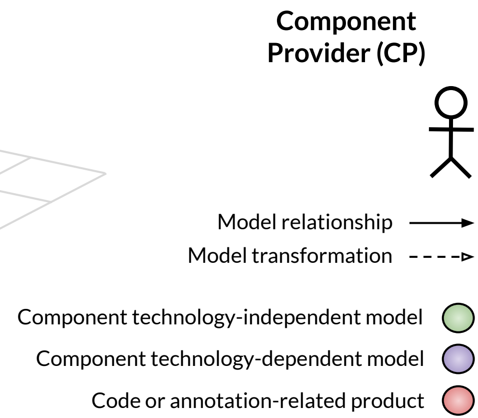
Component model and analysis tool integration

Definition, validation and annotation of new components



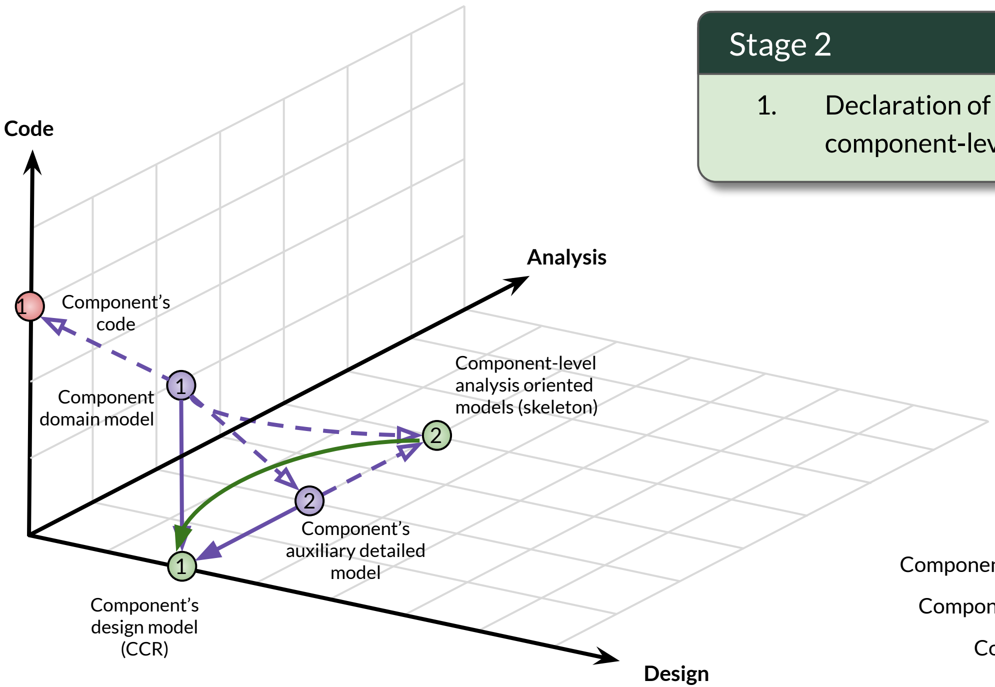
Stage 1

1. Declaration of the component using CCR
2. Definition of domain-specific models
3. Implementation of the component's code



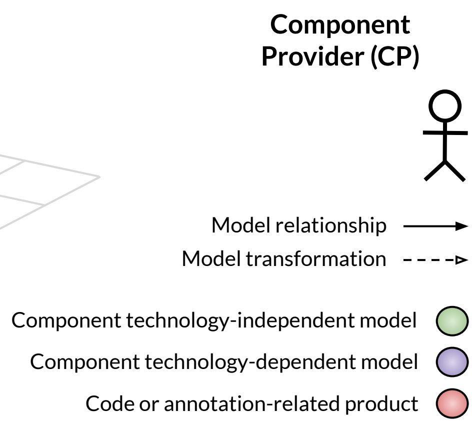
Component model and analysis tool integration

Definition, validation and annotation of new components



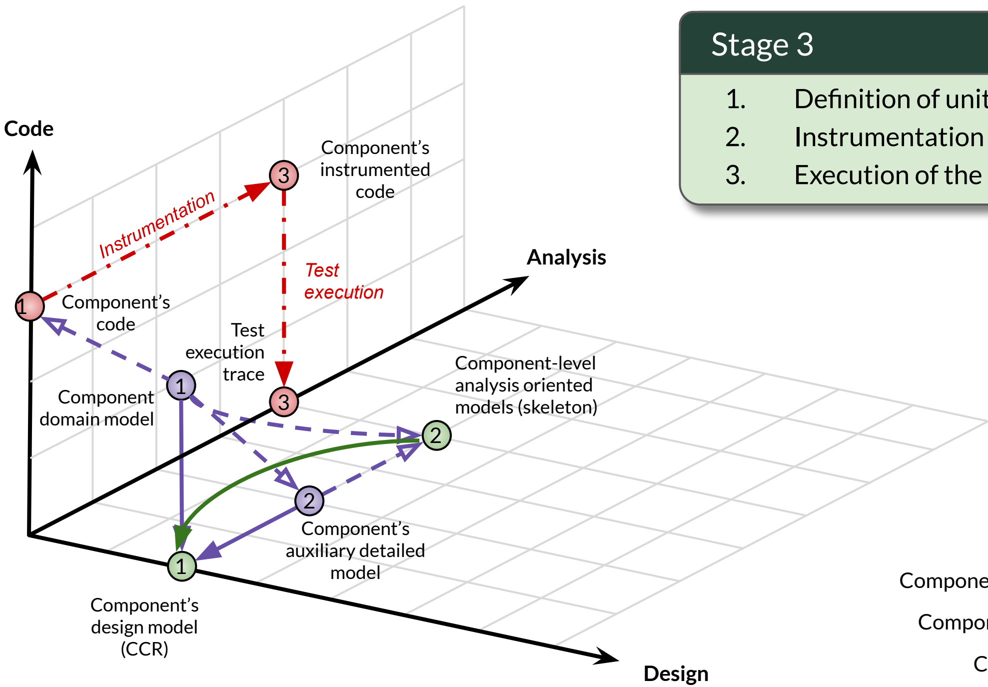
Stage 2

1. Declaration of the skeletons of the component-level analysis oriented models



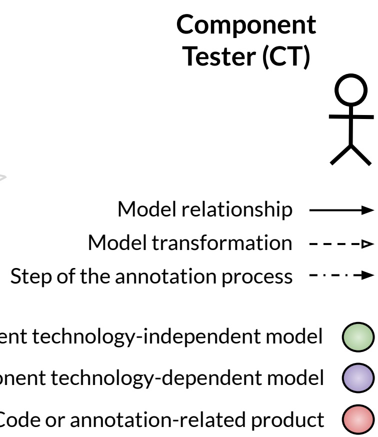
Component model and analysis tool integration

Definition, validation and annotation of new components



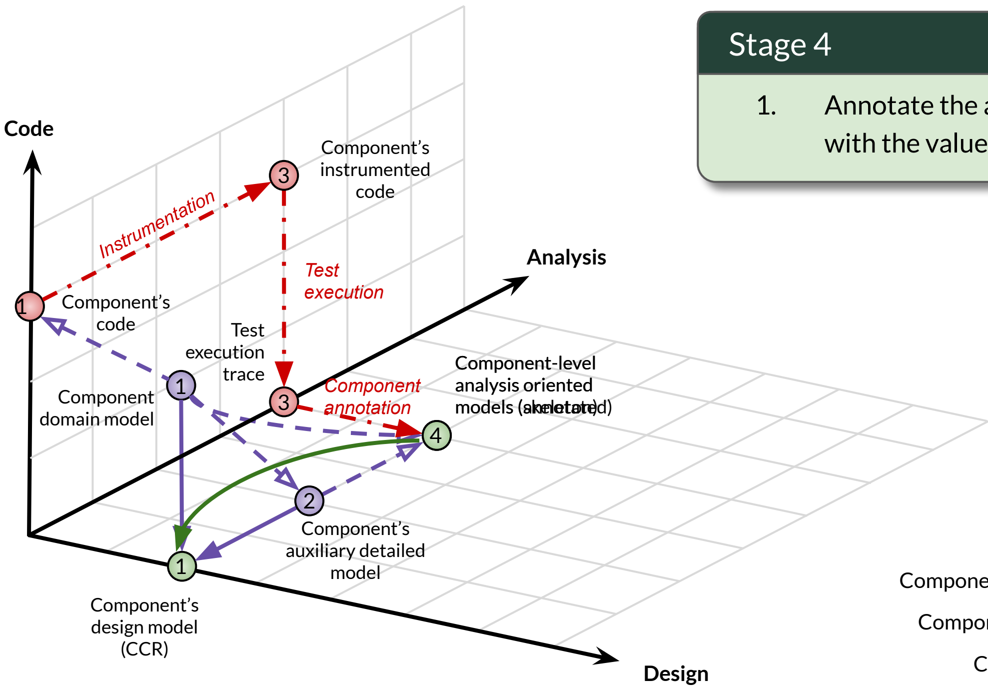
Stage 3

1. Definition of unitary tests for the component
2. Instrumentation of the component's code
3. Execution of the tests



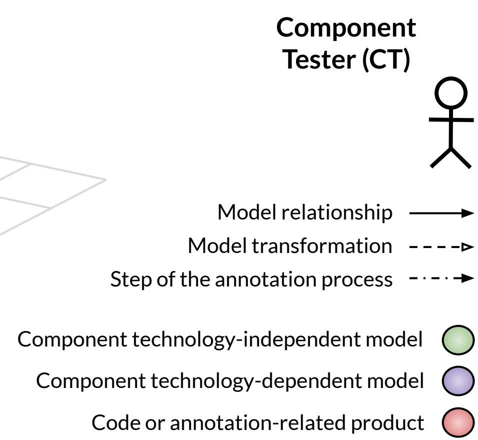
Component model and analysis tool integration

Definition, validation and annotation of new components



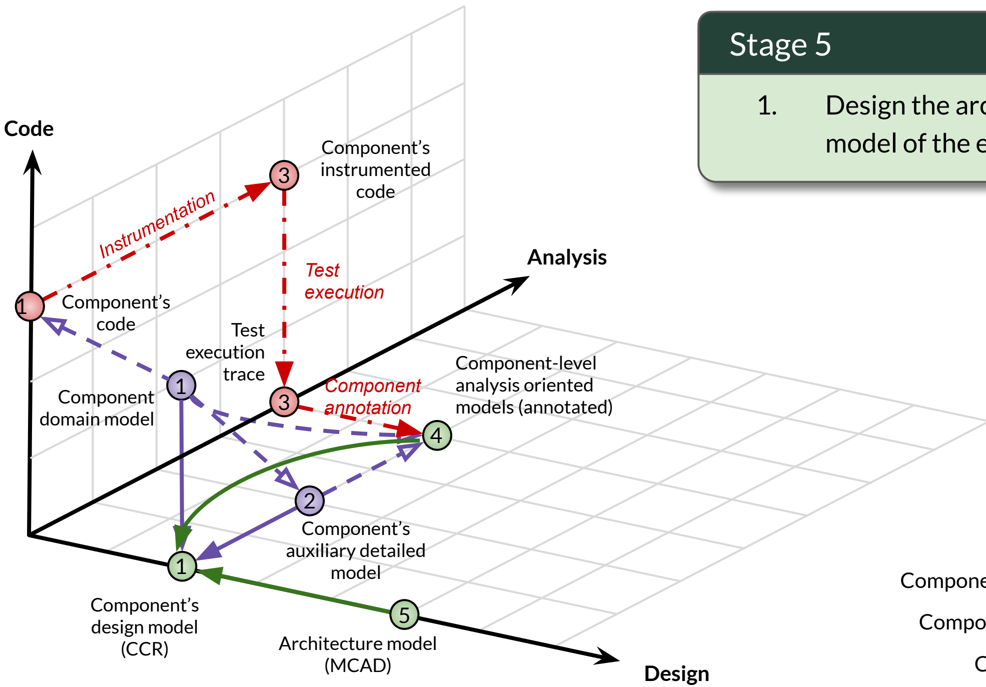
Stage 4

1. Annotate the analysis oriented models with the values of the properties



Component model and analysis tool integration

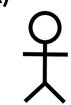
Application assembly and system-level analysis



Stage 5

1. Design the architecture and deployment model of the embedded application

Application Architect (AA)



Model relationship

Model transformation

Step of the annotation process

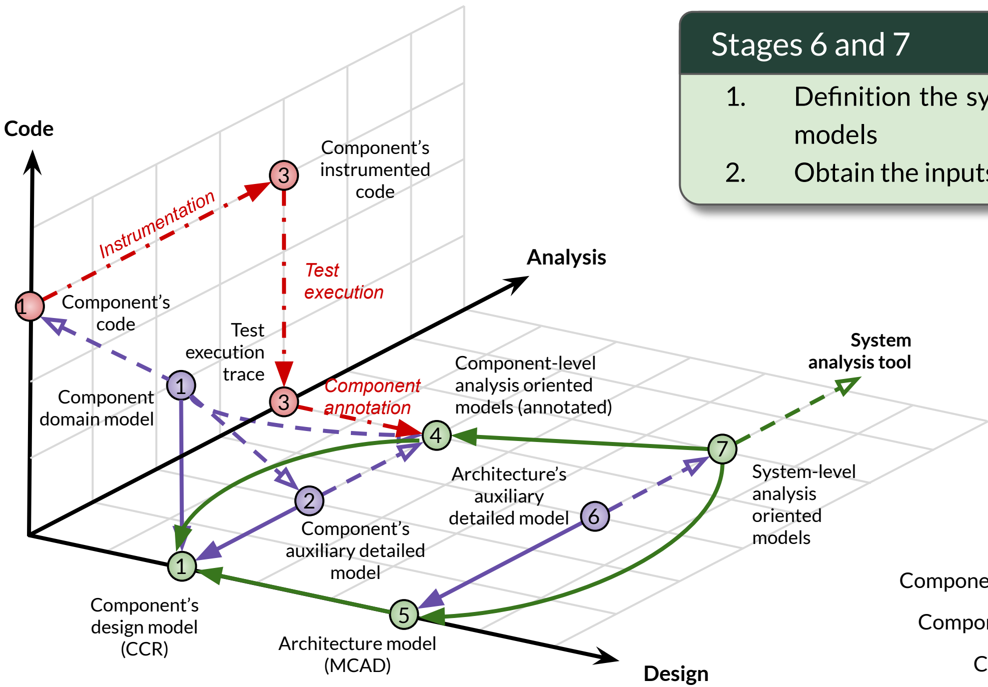
Component technology-independent model

Component technology-dependent model

Code or annotation-related product

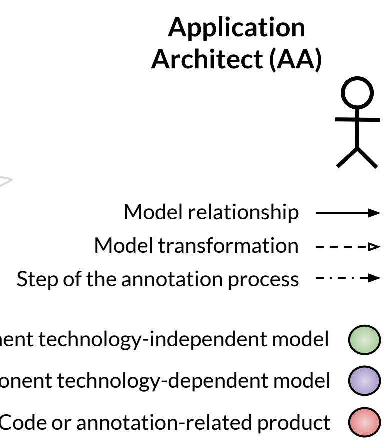
Component model and analysis tool integration

Application assembly and system-level analysis



Stages 6 and 7

1. Definition the system-level analysis oriented models
2. Obtain the inputs for the system analysis tool



Component model and analysis tool integration

Application deployment and validation

- If new components are needed, the Application Architect will request them to the Component Provider
- Once the model of the application is defined, it has to be deployed into the selected platform(s) by the Application Developer (AD)
  - To do so, it shall use the artefacts and procedures defined within the component domain
- The Application Tester will design and implement the integration and acceptance tests
  - If a failure is detected, it shall join the CP, the AA and the AD in a Test Review Board to analyse the alternatives to solving and closing it

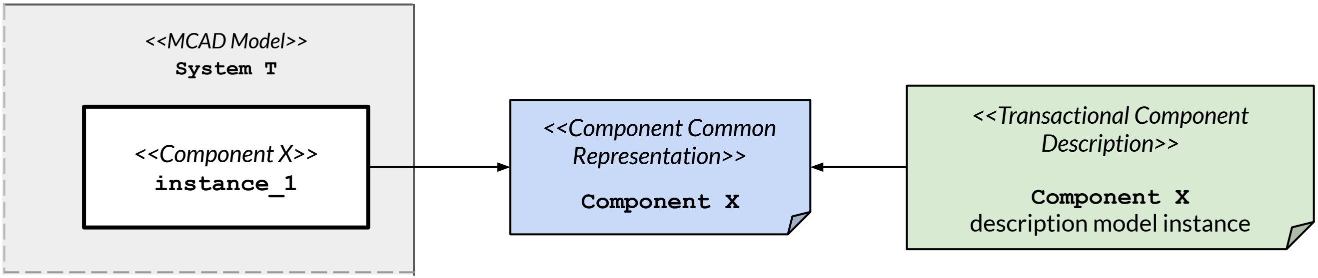
System-level analysis of the application software of EPD's ICU

General overview

- MAST is a model-based schedulability analysis tool developed by the University of Cantabria
- The integration of the MAST tool has been performed by defining the ***transactional system analysis model***
    - Transactional component description model
    - Code blocks description model
    - Real-time requirements model
    - Real-time platform model
- The MAST system model for schedulability analysis is generated through a model-to-model transformation
    - The transformation requires that a deployment platform is chosen between the ones available for the given application

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

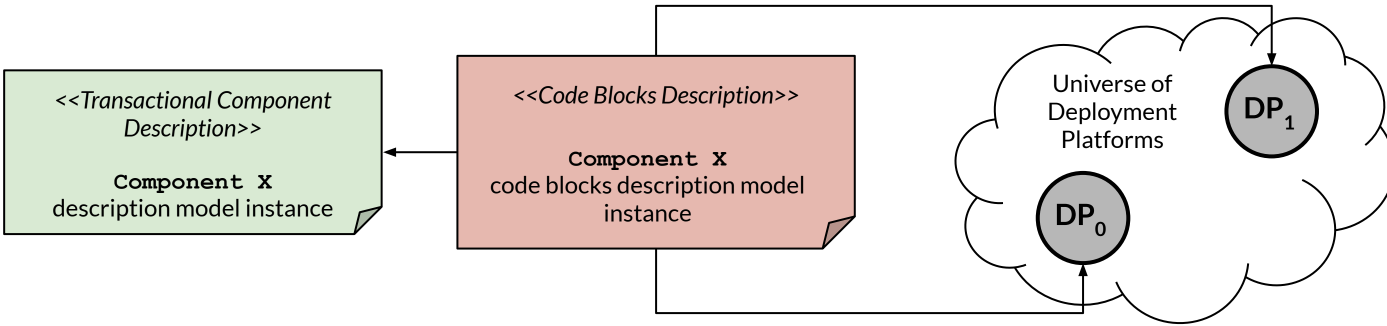
Component-level descriptions



- Transactional component description model
  - It describes the runtime behaviour of the components in terms of the messages they can receive from, and send through, their ports
  - It establishes a list of message handlers, which define the sequence of actions that are taken and messages that are sent in response to the arrival of a message
  - It allows for each component to provide different handlers for the same message

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

Component-level descriptions



- Code blocks description model
  - It describes the body of every action as a set of code paths
  - A code path represents a possible flow or path of execution
  - Each path has attached the information regarding the worst case, average and minimum execution times of the code for a given platform
  - This information can automatically be obtained from specific tools (e.g.: RapiTime)

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

System-level and platform descriptions

- Real-time requirements model
  - It is a system-level analysis oriented model
  - It declares the interrupt and timing events that occur at system level, as well as their trigger patterns
  - It defines the real-time constraints the system shall meet
  - These constraints are expressed as deadlines, associated to a given event, for the execution of the message handlers of the components
- Real-time platform model
  - It is a platform analysis oriented model
  - It includes the timing properties of the system, e.g.: the worst case context switch time, interrupt latency, etc.

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

Integration with the EDROOM component model

- EDROOM was integrated into MICOBS to enable the use of the MAST analysis tool
  - A new auxiliary model was defined called transactional EDROOM component
    - Describes the behaviour of the components in term of message handlers
  - This model was automatically generated using a model-to-model transformation from the behavioural description of the components
  - It allowed the generation of the transactional and code block descriptions of the transactional system analysis model

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

- The different analysis models of the on-board software were generated using the associated transformations
- The code block description models were annotated automatically using RapiTime and the RTBx data logger
    - The instrumented code is executed on the target
    - It generates traces that are sampled by the RTBx data logger
    - RapiTime generates a database from the sampled data
    - An plug-in was defined to extract the information from the database and store it in the analysis models

Conclusions

- The Space Research Group developed the MICOBS framework to support the development of on-board software applications
    - Configuration and deployment of on-board software applications
        - Organization based on software packages and interfaces
        - Software package parameterization and configuration
        - Generation of configuration and construction files
    - Component-based software development
        - Integration of component technologies and analysis tools
        - Generation of system-level analysis-oriented models
        - Successful integration of MAST and EDROOM
- It has been successfully applied in the development of the application software of the instrument control unit of EPD

Thank you very much for your attention

Any questions?