Formal Verification of Space Systems Designed with TASTE

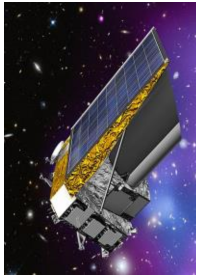
Iulia Dragomir

GMV

Model Based Space Systems and Software Engineering (MBSE) 2021

September 29th, 2021

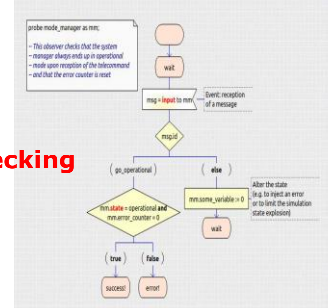
MoC4Space



(Courtesy of ESA)

$\models$ ✓

IF Toolset model-checking



INNOVATING SOLUTIONS

Institut de Recherche en Informatique de Toulouse
CNRS - INP - UT3 - UT1 - UT2J

Outline

Introduction

Model-checking TASTE Designs

Approach Validation

Status

Introduction

SherpaTT during the field tests of the ADE demonstrator developed with TASTE (Courtesy of DFKI)

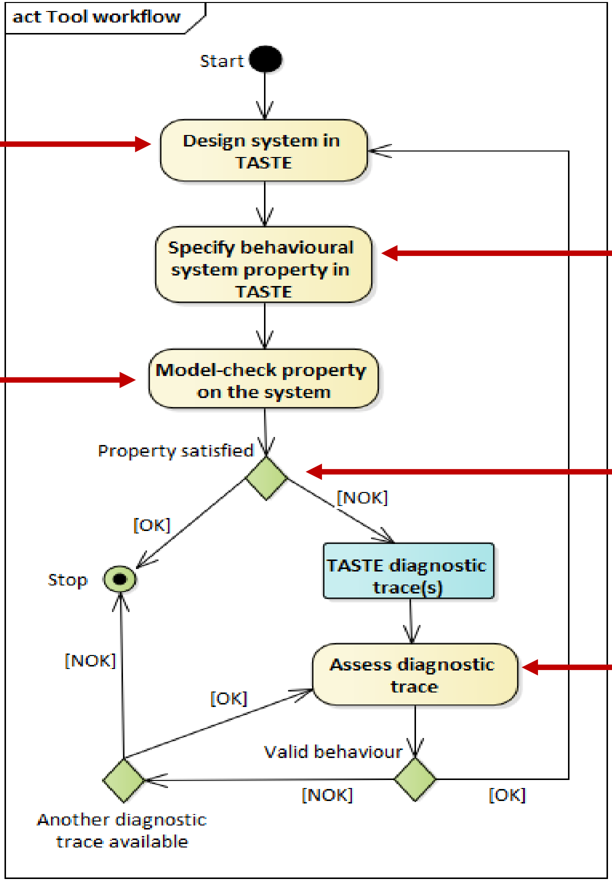
- **MBSE** is an established development approach that enables:
  - Designing large and complex systems with minimal effort and costs
    - ➢ System design includes, among others, software: data types, architecture, behaviour, deployment on processing units
  - Obtaining correct-by-construction implementations/deployments wrt system requirements with the help of (formal) V&V
    - ➢ V&V includes an assortment of techniques such as design review, testing, simulation and model-checking
- **TASTE** is an MBSE toolset that allows:
  - Designing a real-time software system by means of consistent multi-view modelling
  - Generating automatically the application’s executable(s)
  - Checking the system correctness by static type analysis, real-time scheduling, simulation and testing
  - Open topic: formal V&V of TASTE designs
    - ➢ ESA **MoC4Space project** (2021-2022) addresses this shortcoming by integrating a formal V&V approach based on model-checking in TASTE

Formal Verification Approach

Step 1: Design the desired system with TASTE: data view, interface view, SDL state machines / C code

Step 3: Invoke the automated verification technique (i.e., model-checker)

- Select property to check
- Select the design subset on which the property should hold
- Set model-checker parameters (e.g., subtyping, time limit for verification, number of scenarios to obtain)



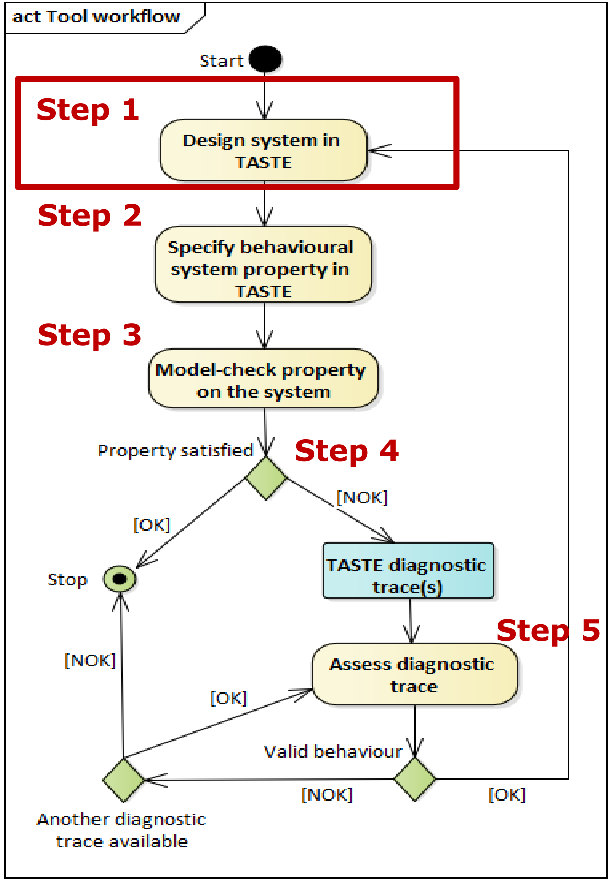
Step 2: Model the properties in TASTE as a Boolean stop condition, MSC or SDL observer

Step 4: Analyse the result obtained

- If the property is satisfied, the workflow stops
- If the property is violated, assess the diagnostic traces

Step 5: Analyse the diagnostic traces one-by-one in the TASTE environment and correct the system design/modelled property if applicable

Step 1: System Design with TASTE



TASTE

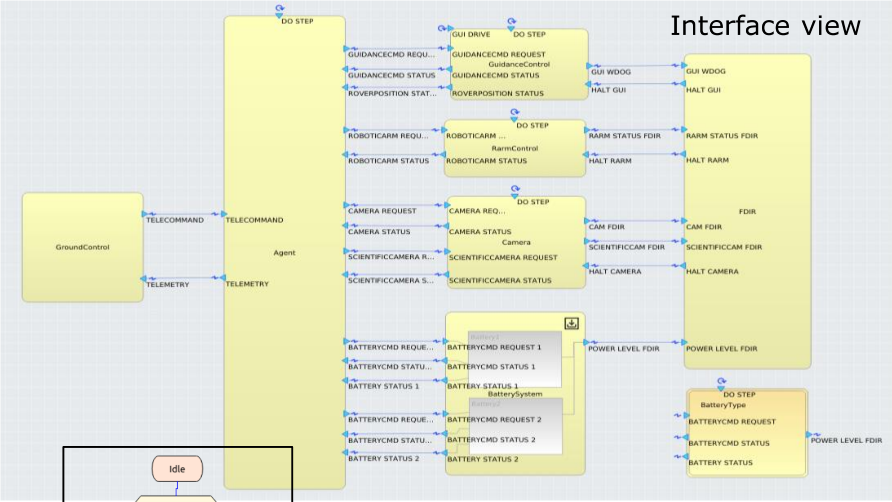
Excerpt from ERGO case study TASTE design

Data view

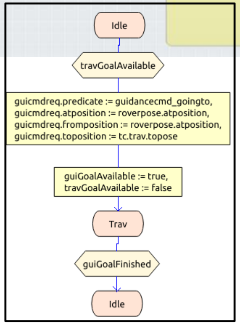
```
Position ::= Vector3d

Pose2D ::= SEQUENCE {
    pos      Position2D,
    orient   T-Double
}

-- Definition of Agent-Functional types

CameraPred ::= ENUMERATED { camera-idle, camera-takingpicture, camera-fault, camera-cancel }
ScientificcameraPred ::= ENUMERATED { scientificcamera-idle, scientificcamera-scanning, scie
BatteryPred ::= ENUMERATED { battery-set, battery-cancel }
```

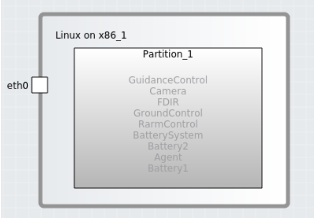
- Model-based development of heterogeneous, reactive, discrete embedded systems
- Uses several modelling formalisms (ASN.1, AADL , SDL, etc.) or programming languages (e.g., C)
- A **TASTE design** consists of:
  - **Data view** (in ASN.1)
  - **Hierarchical interface views** (software architecture and behaviour)
    - Communication is based on the notion of interfaces:
      - Cyclic: execute a behaviour at a certain frequency
      - Sporadic: whenever a request is received handle it
      - Protected: handle the request and provide an answer
    - Behaviour is either modelled as SDL state machines or implemented in C
  - **Deployment view**
  - **Concurrency view** computed from the above

Interface view



SDL behavior



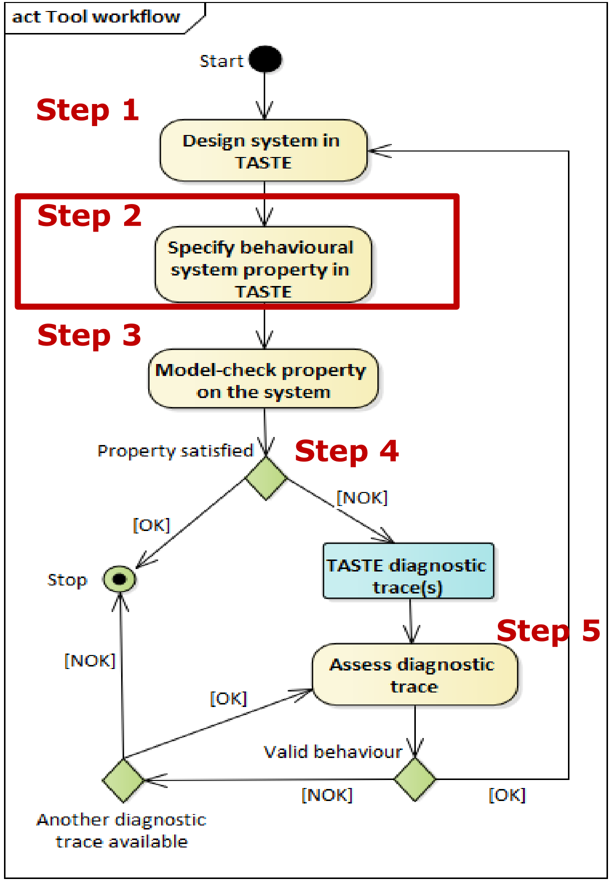


Deployment view

Step 2: Property Modelling

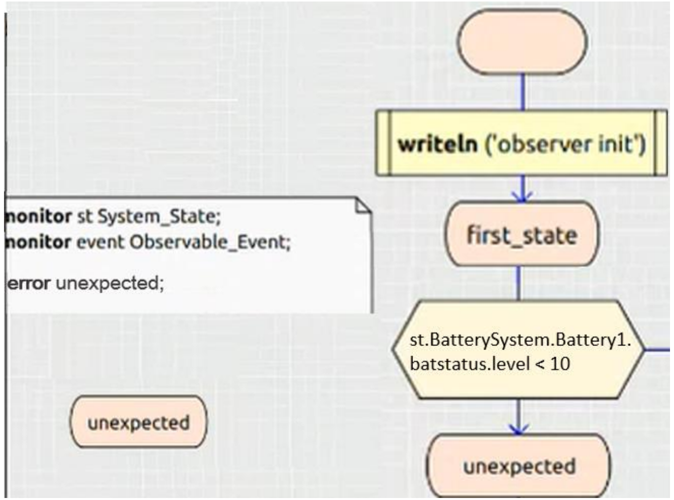
3 types of properties:

- Boolean stop condition (BSC)
- Message Sequence Chart (MSC)
- Observer (in SDL)



Boolean Stop Condition Properties

- Describe undesired behaviour of the system: ***stop if (condition)***
  - The condition is expressed over TASTE system states and variables
  - The evaluation to true of the condition implies that the property is not satisfied, and hence the design/property need to be corrected
- Directly accessible in TASTE space-creator GUI
- Modelled by the user as observers in OpenGEGODE
  - The property skeleton is automatically generated
  - The user specifies the condition in TASTE SDL observer language (already available)



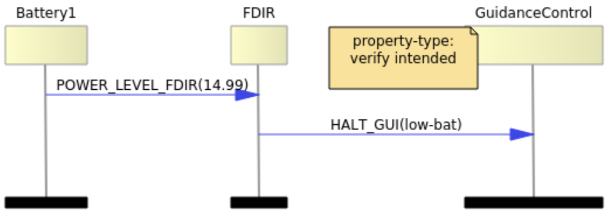
Property: the level of battery 1 shall not drop to the critical value of 10 units

Message Sequence Chart Properties

- Describe desired/undesired sequences of I/O events between some of the functions defined in the Interface View
- Properties available to the user

  search [[**from-start** | **nonstrict**]]
  ((**intended** | **unintended**))
  verify [[**from-start**]] **intended**

- Directly accessible in TASTE space-creator GUI
- Modelled with the available TASTE MSC editor, property type specified via a comment



Property: if the level of battery 1 drops below 15 units, the FDIR stops the system

search: find a system execution complying to the MSC

verify: all system executions comply to the MSC

from-start: the execution is matched from the beginning of interactions between the represented functions

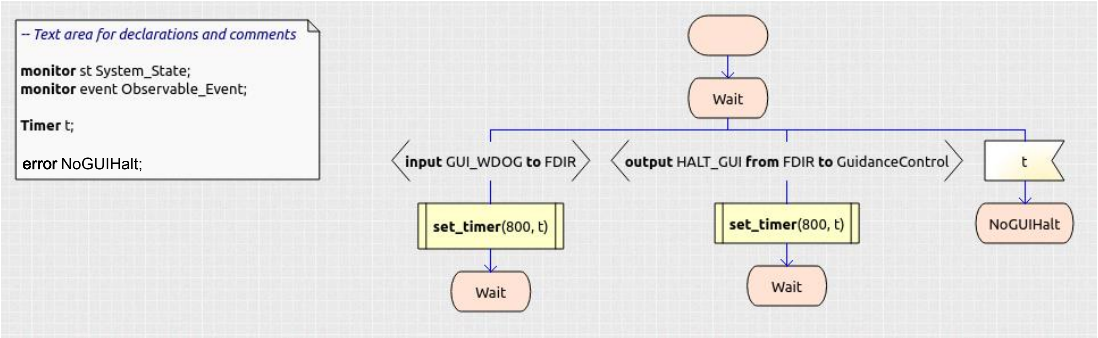
nonstrict: other interactions can happen between the represented functions

intended: desired behavior

unintended: undesired behavior

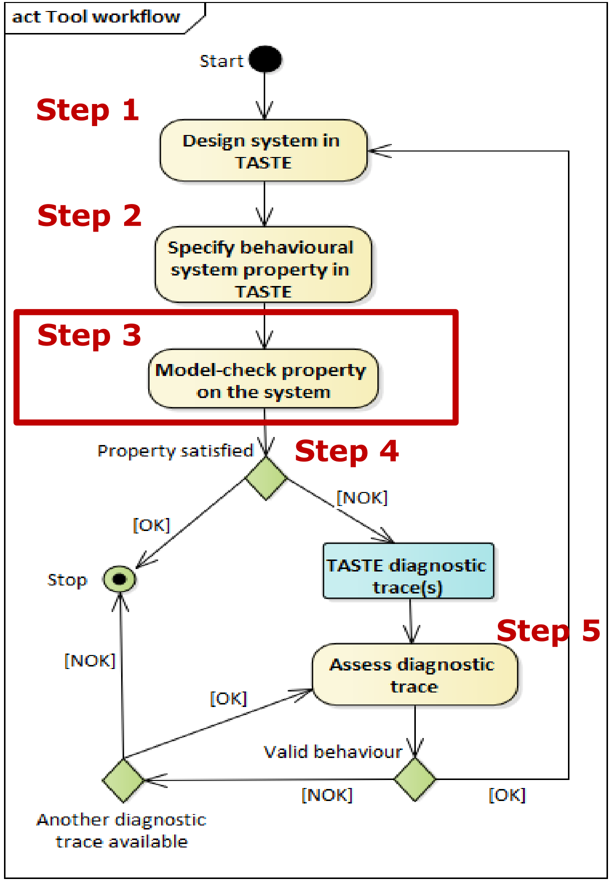
SDL Observer Properties

- Describe desired/undesired behaviour of the system in the form of state machine in OpenGEODE
  - Monitor the system state, variables and I/O events
  - Alter the system execution to guide the verification
  - Desired behaviour is modelled by reaching a state catalogued **success**
  - Undesired behaviour is modelled by reaching a state catalogued **error**
- Directly accessible in TASTE space-creator GUI
- Modelled with the available OpenGEODE editor, already extended for observers



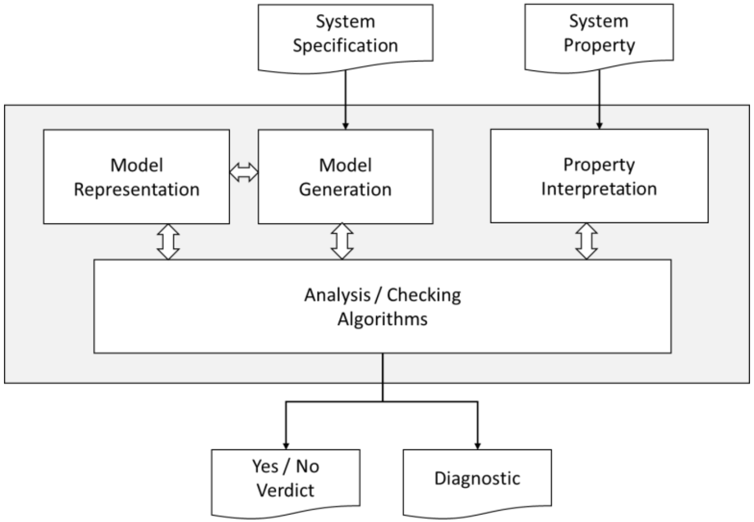
Property: the FDIR stops GuidanceControl
if no status is received before 800ms

Step 3: Formal Verification



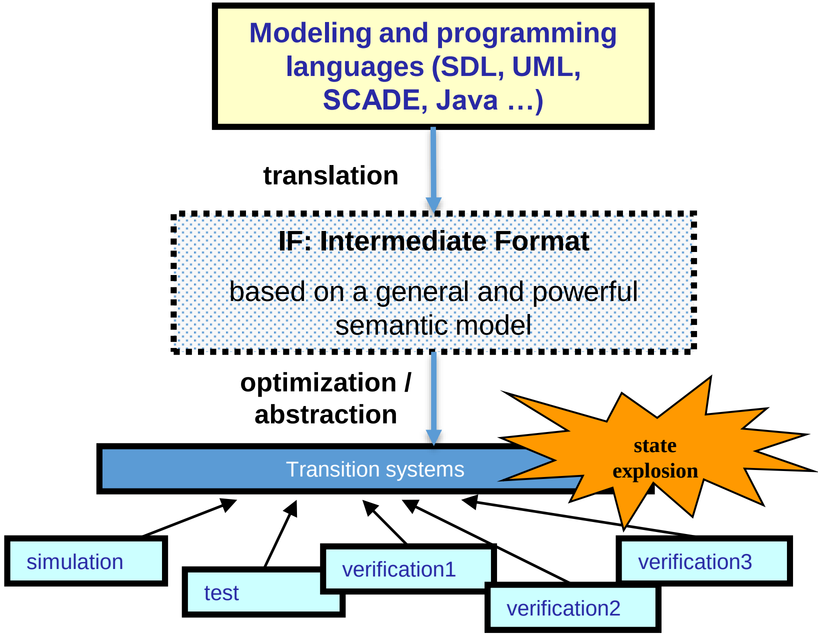
Step 3: Model-Checking

- **Formal verification technique for system correctness with respect to a defined set of properties**
- **Pros**: exhaustive exploration of the model (potentially guided by properties), fully automated, easy production of counterexamples
- **Cons**: state space explosion problem
- **Main principles**:
  - *Model generation*: building the system state space
  - *Model representation*: data structures and methods to store and explore efficiently the model
  - *Property interpretation*: language for property specification and the data structures and algorithms for verifying them
  - *Analysis/Checking algorithms*: algorithms and tools to explore the model for verifying properties
- **Tools**: IF, UPPAAL, NuXMV, Spin, LTSmin



The IF Toolset

- **Model-based development of real-time systems**
- **Features**:
  - Use of *high level modelling* and *programming* languages: expressivity for faithful and natural modelling, cover functional and extra-functional aspects, openness
  - *Expressiveness*: direct mapping of concepts and primitives of high modelling and programming languages (asynchronous, synchronous, timed execution, buffered interaction, shared memory, method call, etc.)
  - *Semantic tuning*: when translating languages to express semantic variation points, such as time semantics, execution and interaction modes
  - *Model-based validation:* combines static analysis and model-based validation; integrates verification, testing, simulation and debugging
- **Applications**: protocols, embedded systems, asynchronous circuits, planning and scheduling



Resources: https://www-verimag.imag.fr/~async/IF

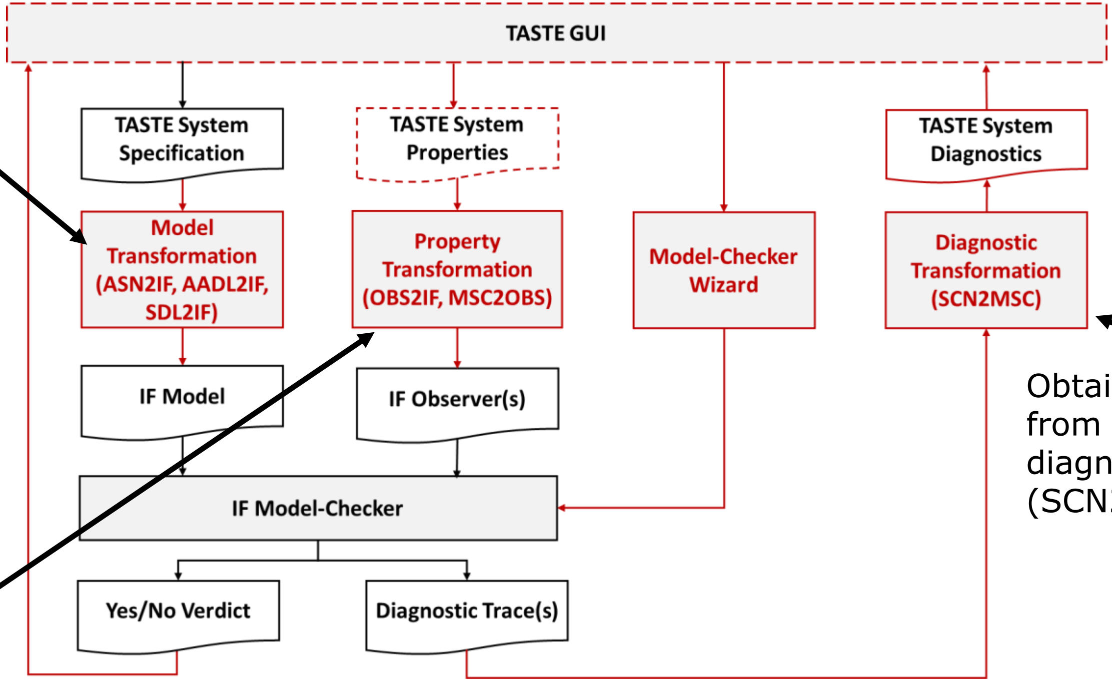
Integration in TASTE: Transformations

Obtains an IF model from a TASTE design:

- ASN2IF for data view
- AADL2IF for interface view, GUI functions and C functions behavior
- SDL2IF for SDL state machines

Obtains an IF observer from any TASTE modelled property:

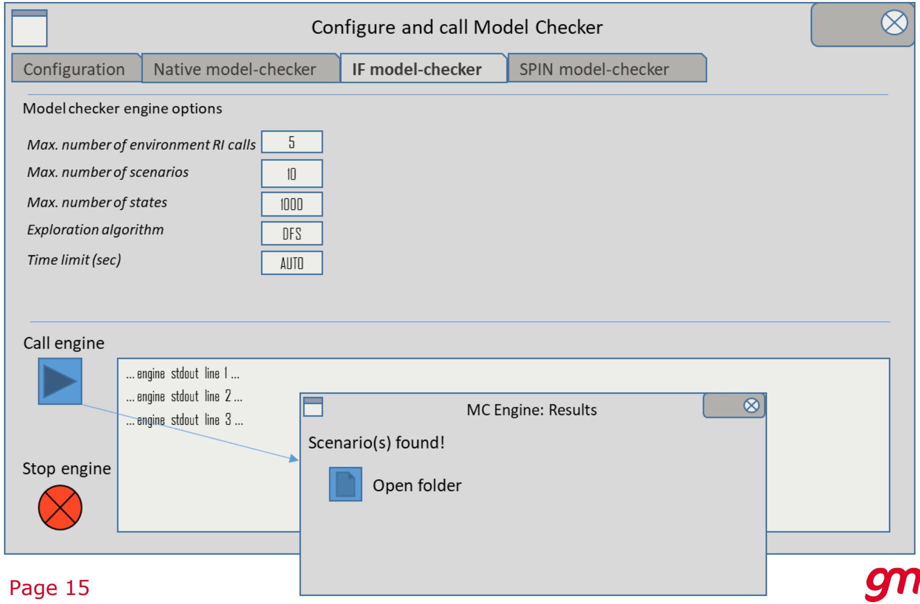
- OBS2IF for BSC and SDL observers
- MSC2OBS for MSC to SDL observers and then using OBS2IF



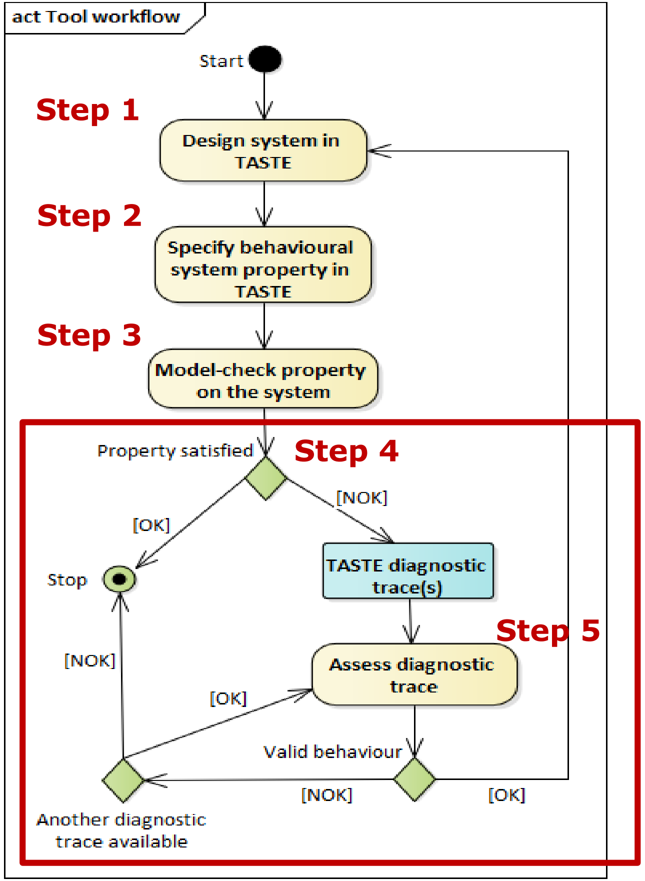
Obtains a TASTE MSC from the IF generated diagnostic traces (SCN2MSC)

Integration in TASTE: Model-Checker Wizard

- **Configuration tab:**
  - System properties: listing, creation, editing, deleting
  - Sub-system on which to check the properties (if applicable)
  - Environment subtyping (restricting the values the model-checker can produce as input of the model)
- **One tab per available model-checker** with their inherent options, calling and stopping the model-checker
  - E.g., IF model-checker with
    - Maximum environment RI calls
    - Number of diagnostic traces to be produced
    - Number of states to explore
    - Algorithm for exploration (bfs, dfs)
    - Time limit for model-checking
    - Etc.



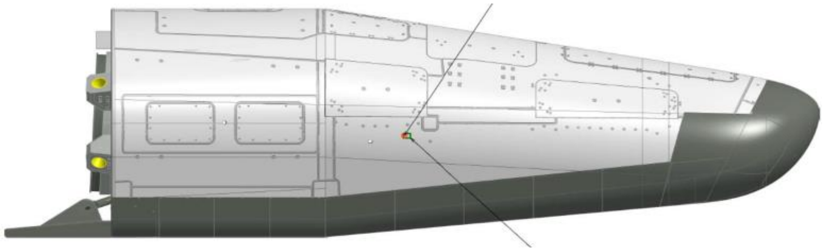
Steps 4 & 5: Model-Checking Results Assessment

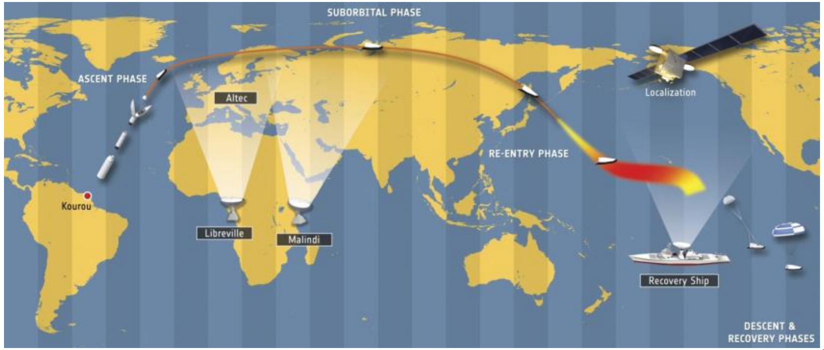


User activities

Approach Validation: IXV case study

- Space vehicle to experiment on atmospheric re-entry with fully-automated sub-orbital flight
- Successful flight in Feb 2015
- Originally modelled in UML (Enterprise Architect) and having 77KLOC hand-written C code
- Two main functional behaviours fully modelled with TASTE:
  - Flaps positioning sequence upon LV separation
  - Flaps FDIR sequence and deactivation
- Properties defined (together with the expected verification result) and partially modelled in TASTE: 2 BSC, 6 MSC, and 2 OBS
  - Examples: separation from launch vehicle and start of the flaps positioning sequence; flaps FDIR sequence





Approach Validation: ERGO case study

- Scenario inspired by the Mars Sample Return of an autonomous planetary exploration rover able to pick samples with a robotic arm, as well as taking images of scientific interest
- Originally developed with TASTE, C++ and BIP, around 2 MLOC (generated code included), and demonstrated in Morocco's desert in Nov-Dec 2018
- Case study consisting of the simplified functionalities of
  - Telecommanding (E1) and goal commanding (E4)
  - Simulation of traverses to specified poses, sample picking/dropping at different location, image taking of the environment (snapshots or periodically), battery operations and FDIR
- Properties defined (together with the expected verification result) and partially modelled in TASTE : 3BSC, 7 MSC, 5 OBS
  - Examples: there is no drop operation before a pick; FDIR works nominally

DFKI's SherpaTT in the Moroccan desert for the ERGO demonstration

Current Status

- Tool partially implemented:
  - Completed: ASN2IF, SDL2IF, MSC2OBS
  - Ongoing: AADL2IF, TASTE project template and compilation
  - Pending: OBS2IF, model-checking wizard (work in collaboration with N7 Space)
- Case studies fully developed in TASTE (properties included)
- Validation of the available components on the case studies
- Toolset and case studies available at https://gitrepos.estec.esa.int/taste/if-model-checking

Thank you!

idragomir@gmv.com