INTRODUCTION

This paper presents the work performed in the context of the MUROCO-II on-going project. It has as objective the implementation of a generic formal mission specification and verification tool, named FORMID, and its integration into the DREAMS ESA robotic ground control station ([1,2]), in order to support the A&R activities specification, formal verification and execution for the AURORA Mars exploration missions. MUROCO-II is the continuation of the MUROCO-I ([3]) study that provided a formal approach to the instantiation of the ESA Functional Reference Model robot control architecture. It established the framework and the benefits of the use of modern techniques for formal specification and formal verification of robotic tasks and actions that coordinate the robot behaviour to achieve the objectives of a mission. In contrast to the current approaches of robot programming, where there is a lack of high level primitives to specify parallelism, concurrency and synchronisation and where the validation of the tasks is performed by simulation covering a limited number of possible scenarios, in this case complex behaviours can be easily specified and the validation is done using rigorous mathematical analysis of all the possible states of execution. The use of formally defined languages for the coordination of the behaviours is expected to significantly increase the possibility of specifying complex behaviours providing more autonomy capabilities and to drastically reduce the testing and debugging interval while it will highly improve our confidence on the uploaded code.

The integration of FORMID into DREAMS and its customisation for Mars exploration missions offers a solid basis for the future ground control station of robotic exploration missions. This paper is structured as follows: first we present the MUROCO framework, then we provide a technical insight into FORMID and finally we present its integration into the DREAMS ground control station.

MUROCO CONTROL ARCHITECTURE: ASSUMPTIONS AND CONCEPTS

The FORMID module assumes that the control architecture of the target robotic system follows the design concepts of the Functional Reference Model (see Figure-1a) and uses for its implementation at least the concepts, if not the languages and tools, proposed in the MUROCO-I activity.

In particular, in this design the elemental robotic Action is formally defined ([4,5,3]) as the parameterised specification of an elementary control law, i.e. the activation of a control scheme structurally invariant along the task duration, and a
logical behaviour associated with a set of signals (events) which may occur just before, during and just after the task execution.

From a control point of view on the one hand, the specification of an Action requires the definition of the functions, the models and the parameters that appear in the (continuous time) analytical expression of the control outputs to be applied to the actuators in order to perform the physical action. On the other hand, the specification of the logical behaviour is obtained by setting the events to be considered and their exception handling.

These events and the associated processing are typed as follows:

- The pre-conditions: their occurrence is required for starting the execution of the control law. We distinguish between synchronisation and measurement pre-conditions. The synchronisation pre-conditions are related to the activity of other Actions and Tasks while the measurement pre-conditions are related to the environment evolution.

- The exceptions: they might be emitted during the execution of the control law to indicate a failure detection. Their processing is as follows:
  - Type 1: the reaction to the received exception is limited to the modification of a parameter value within the control scheme, e.g. setting a regularisation parameter to smoothly cross a kinematic singularity.
  - Type 2: the exception requires the activation of a new Action. The reaction consists of killing the current one and reporting the causes of the malfunction to the adequate level. The recovery process to activate is known and specified in the Task in which this Action participates.
  - Type 3: the exception is considered as fatal. Type 3 exceptions lead to a Task abortion through a safety behaviour which may be context dependent and the request of re-planning in case an o/b planner is available.

- The post-conditions: The measurement post-conditions are often related to the environment and they are handled as conditions for a normal termination of the Action. The synchronisation post-conditions are communicated to the Tasks (e.g. indication of the nominal termination of the Action).

A very important feature of critical systems is the exception handling (see Figure 1b). The events are strongly typed with precise semantics. This typing allows for a systematic and hierarchical exception handling and is a key feature for automatic code generation ensuring that exception handlers are correctly nested according to their importance.

The characterisation of the interface of an Action with its environment in a clear way, using typed input/output events, allows composing them easily in order to construct more complex robotic activities, the Tasks. The aim in designing this entity is to be able to define a representation of robotic activity that could fit any abstraction level needed by the mission specification system. In its simplest expression a Task coincides with an Action, while in the most complex, one might represent the whole mission. Briefly speaking, a Task specifies in a structured way a logical and temporal arrangement of Actions in order to achieve an objective in a context-depandant and reliable way, providing predefined corrective actions in the case of unsuccessful execution of Actions.

The composition of Actions and Tasks is obtained through operators which express sequence, parallelism, conditions, iterations, rendez-vous and various levels of pre-emption. The operational semantics of the Tasks are defined by the use of the Esterel synchronous language as specification language. Esterel is well suited for the Tasks specification since it is a high level, imperative language especially designed for writing reactive programs, i.e. communicating programs of which the behaviour depends on the ordering of input and output events. Esterel’s operators and its modular design allows for hierarchical programming and for straightforward specification of complex behaviours.

FORMID: FORMAL ROBOTIC MISSION INSPECTION AND DEBUGGING TOOL

FORMID is the environment that we are developing for DREAMS that enforces the MUROCO formal approach on the specification and verification of robotic Actions and Tasks. We use Esterel as the underlying high-level specification language for a concrete implementation as proposed in the MUROCO study. However, the approach is fairly independent from the chosen specification language.

FORMID consists of four components that reflect the different stages in the specification and verification of Actions and Tasks, and a main component that supports the coordination and management of the work involved:
- The Workbench, that provides centralised access to the other components and allows the operator to organise his work.
- The Action/Task Specification is FORMID’s formal specification component with a HMI that enforces the MUROCO approach.
- The Task Verification is the formal verification component that also facilitates the specification of the properties to be checked.
- The Action/Task Simulation is FORMID’s discrete event simulator with capabilities of visual debugger and scenario playback.
- The Execution Machine Generation automatically creates the code to be uploaded to the target controller.

**Figure-2:** The FORMID a) Workbench panel, b) Action Specification panel and c) Verification panel

**The Workbench**

The Workbench (see Figure-2a) is the central component of FORMID from which the other components are launched. The main widget of its HMI is the Librarian that supports the organisation of Actions and Tasks in libraries. For Actions, there is a fixed predefined set of libraries, roughly reflecting the robotic subsystems (both physical and control). On the other hand, there is no restriction on the number or themes of Task libraries. The Librarian also helps keeping track of the validation status of Actions and Tasks, both in function of verification and simulation results as well as specification modifications.

The Repository on the Workbench is a pool of validated Actions and Tasks that must be explicitly managed by the operator. The content of the Repository provides the component references for new Tasks. Using an explicitly maintained pool enforces a methodological approach to compositing Task hierarchies and simplifies the search for valid Task components during Task specification.

Finally, the Completed Tasks list on the Workbench is a summary of all validated Tasks that are eligible for validation on the A&R Emulator (ARE) or Ground Reference Model (GRM). The generated code to be uploaded (Execution Machine (EM) generation) can be requested for a Task on this list.

**Action/Task Specification**

This component enables the specification of the building blocks in the MUROCO approach, i.e. Actions and Tasks. The aim is to hide as much as possible the implementation details of these concepts in the underlying specification language. By adhering to a strict definition of the building blocks, the specific design of the specification HMI can enforce constraints on building block properties that can not be directly represented in simple constructs of the chosen specification language, thereby simplifying the operators’ work and improving correctness. The obtained specifications
are stored in an activity database from which most of the corresponding Esterel code can be automatically generated on demand.

The Action/Task Specification component is further subdivided in two independent subcomponents, one specialised in Action specification, the other in Task specification. They both allow the specification of only a single Action resp. Task at the same time.

**Action Specification**

In FORMID, we generalise the continuous time aspect of an Action to include tele-commands and computational tasks and call this generalisation an Action Implementation. Hence, in the context of FORMID, specifying an Action consists of encapsulating an Action Implementation in a logical behaviour. A full specification therefore consists of a list of typed parameters to the Action Implementation and sets of events that the Action must handle (see Figure-2b).

The corresponding Esterel code is completely automatically generated because the logical behaviour basically consists of handling the synchronisation (‘Start’/‘Abort’/‘GoodEnd’) (in lockstep with activating/deactivating the Action Implementation) and condition events, acknowledging the occurrence of Type 1 exception events, and forwarding Type 2 and 3 exception events to Task level.

**Task Specification**

A full specification of a Task consists of a list of component references (i.e. to other Actions and Tasks) copied from the Repository, a list of typed parameters, a set of Task-level events and the Task coordinator module, which in the current conception will be text entered in the Esterel language. The final set of events will be the union of the set of Task-level events with the set of events derived from the composition.

In FORMID, we restrict the hierarchical structure of the Task composition to a tree, i.e. we do not allow recursion between any level of the hierarchy. This is enforced during modification by not allowing the addition of sub-Tasks that have already the root Task as a member of their composition tree.

A large fraction of the corresponding Esterel code is automatically generated, including a skeleton for the coordinator module. The body of the coordinator that implements the logical behaviour of the Task will have to be hand-coded but FORMID provides editor tools that generate certain code patterns that will have to be completed manually. Other editor tools help in simple code validation by searching for markings in the generated patterns that indicate sections that have not yet been completed. The markings are chosen such that compilation is guaranteed to fail.

**Task Verification**

FORMID Task Verification (see Figure-2c) applies the structuring proposed in the MUROCO study. It allows two methods of verification: automatic verification of formally verifiable properties and visual verification. The first method involves the specification of generic properties or properties as instantiations of property templates by selecting a template and a set of entities. These entities are the events and all the Actions and Tasks in the node set of the composition tree.

The generic properties are ‘Safety’, ‘Liveness’ and ‘Conflict-free’ (i.e. the absence of contention of Actions on a subsystem). The property templates express temporal relations between the occurrences or executions of entities, and the execution of entities in an unordered sequence. The supported temporal relations are:

- “… triggers …”
- “… is always preceded by …”
- “… always takes place during …”
- “… always implies …”

of which only the first relation can involve events, in particular, as triggers.
Then a formal verifier is invoked, on request, that checks the properties of the model without test vectors. It uses model-checking techniques to compute all possible reachable states and then checks whether a property holds under all allowed input conditions. Formal verification is performed by generating ‘Observer’ modules. The role of an Observer is to spy, without interfering with the system model whenever a property violation is detected. In that case, the verifier provides the user with a scenario that induces the violation. If desired, this scenario can then be played back in the interactive simulator to find the cause of the verification failure.

The second method of verification involves the specification of a property as a general set of entities (as defined above). Then an abstract view of the Task automaton with respect to the selected entity set is created and the verification is performed by visually inspecting the abstract view (see Figure-3a). Verification is a necessary step in FORMID for Tasks to reach discrete event validation.

**Figure-3:** a) Abstract view of a Task automaton. b) The simulation tool.

**Action/Task Simulation**

Simulation is practical and complete enough to validate Actions against discrete events. In the case of Tasks, simulation is an additional, necessary step in FORMID to reach validation after a specific set of properties have been formally verified. Task simulation can be used to test explicit scenarios in order to catch problematic Task behaviour that can not be caught through verification alone.

The discrete event simulator is a visual debugging tool provided by the Esterel environment (see Figure-3b). It is used to animate the model, set breakpoints, position input events, examine the value of variables and output events, etc. It is also possible to record and playback scenarios. As mentioned above, one way to automatically produce these scenarios is to use the verifier. Discrete event simulation is a necessary step in FORMID for both Actions and Tasks to reach discrete event validation.

The simulator is also used as a monitoring and tracing tool during execution, or playback of previous executions, of a Task on the ARE/GRM by driving it with telemetry.

**Execution Machine Generation**

This component automatically generates, from a discrete event validated high level specification of a Task, the embedded Execution Machine (EM) that runs the Task. The EM consists of a Synchronous Process (SP) that is driven by an InputModule (that converts physical input signals to logical signals) and that in turn drives an OutputModule (that converts logical signals produced by reactions in the SP to physical output signals).

EM generation involves the generation of the event jump table that connects the Task automaton to the generic event loop of the SP, the generation of the interfaces connecting the Action Esterel modules with their corresponding Action...
Implementations and the compilation of the Esterel modules for the elements in the node set of the Task composition tree. This component does not require operator intervention.

**F-DREAMS GROUND CONTROL STATION FOR THE EXOMARS09 MISSION**

The operational environment of the FORMID module is the F-DREAMS Ground Control Station as proposed for the ExoMars09 mission ([6]). It is assumed that the ExoMars09 ground segment consists of the ESOC control stations, the control stations located at the Facility Responsible Centre (FRC) and a set of User Home Base (UHB) control stations distributed over the Internet. F-DREAMS implements the FRC and the UHB’s control stations (see Figure-4). The responsibilities attributed to each of these control stations are as follows ([7]):

- ESOC responsibility is the ‘Rover and Payload commissioning’ phase. During this phase the ESOC control stations are used to check out the rover systems (but without any locomotion), to check out the payload and to commission it at the landing site. During the ‘Routine Operations’ phase the ESOC ensures the communications with the flight segment.

- FRC responsibility is the ‘Routine Operations’ phase. Initially, the rover locomotion and navigation system is checked-out and commissioned. The rover software parameters may have to be adapted to the landing site. At the end of this first phase, under nominal conditions the rover will operate on a pre-planned basis: a stack of Tasks prepared and validated on ground are uploaded onto the on-board controller. These Tasks are executed with the help of on-board intelligence and on-board feedback loops (the Actions). Contingency situations are handled under the responsibility of the FRC. The data download/upload and the TM/TC exchange with the flight segment are done via the ESOC control stations.

Figure-4: F-DREAMS configuration for ExoMars09 mission
- A set of UHB control stations, geographically distributed and connected over Internet, can make connections with the FRC to submit science objectives.

Based on this distribution of responsibilities the F-DREAMS GCS subsystems (s/s) are allocated as follows on the different control centres (see Figure-4).

- The RM&C s/s shows, in real time, the evolution of the robotics data on the 2D and 3D monitors. Activity level TCs are prepared, checked and transmitted to the rover facility.

Two control stations are located at the FRC: the Monitoring & Control (M&C) Station and the Preparation & Verification (P&V) Station (see Figure-5).

- The M&C Station has the responsibility to supervise, i.e. monitor and control the rover facility by exchanging in real-time TM/TC with the Flight Model (FM), to upload Activity Plans produced by the P&V Station and to download, process and distribute data from the FM. To this end the M&C Station integrates the following s/ss:
  
  - an On-Ground Data Handling Unit (OGDHU) component that decodes the data and the TM received by the FM, transmits them for real-time monitoring to the RM&C and the FMC s/s, generates in parallel the derived products and stores them in the databases. In addition, the TC issued by the Robot M&C (RM&C) and the Facility M&C (FMC) s/s are encoded and transmitted to the FM.
  
  - the FMC s/s shows, in real-time, the evolution of the system level FM TM values, generates alarms in case out-of-line (OOL) situations are detected and manages the FM o/b software. System level TCs are prepared, checked and transmitted to the rover facility.

- The P&V Station has the responsibility to prepare, using the ARE and the GRM, the Activity Plan that will be transmitted to the M&C Station for uploading and execution by the rover during the next time period. In addition, it is the station that prepares and validates existing and new Activities and controller data to be uploaded to the FM. To this end, the P&V Station integrates the following s/ss:
  
  - the OGDHU component decodes the data and the TM received by the ARE/GRM, transmits them for real-time monitoring to the P&V and the FMC s/s, that generates in parallel the derived products and stores them in the databases. In addition, it encodes and transmits the TC received by the P&V and the FMC s/s to the ARE/GRM.
  
  - the P&V s/s allows the specification, verification and validation using the ARE and the GRM of existing and new Activities, prepares Activity Plans mainly for engineering objectives, receives Activity Plans specified by the PMC Stations and merges them to a final Activity Plan to be transmitted to the M&C Station. The P&V s/s can be instantiated and replay ARE/GRM and FM downloaded Activities related data, such as execution traces.
  
  - the FMC s/s shows, in real-time, the evolution of the system level ARE/GRM TM values, generates alarms in case OOL situations are detected and manages the ARE/GRM o/b software. System level TCs are prepared, checked and transmitted to the ARE/GRM. The FMC s/s shows in retrieval mode ARE/GRM and FM downloaded system level data for further analysis.
  
  - the CALIB s/s manages the GRM and FM mast and instrument arm calibration and produces new models to be uploaded to the GRM and transmitted to the M&C Station for uploading to the FM.
  
  - the Perception s/s (PERC) manages the Imaging Head providing the means for single camera and stereo camera calibration and DEM generation.
- The **Payload M&C (PMC) Stations** (see Figure-6) at the UHBs retrieve from the M&C Station of the FRC the science and the needed engineering data to instantiate the PMC s/s. The operator, using the 2D Monitoring functionality visualises the scientific data in adequate format and analyses them to judge their importance. They could possibly indicate that the investigated area has to be further explored or that a new area has to be visited. Using 3D Monitoring and Activity Planning functionality the operator specifies and validates by simulation (at state level) an Activity Plan that later is submitted to the P&V Station at the FRC to be scheduled in the next operations window.

![Figure-6: DREAMS Payload Monitoring & Control s/ss](image)

The functionalities of the FORMID module are involved as follows in the proposed GCS structure:

- The FORMID module is used by the P&V Station for:
  - the specification, simulation and verification of new Actions and Tasks.
  - the on-line monitoring of the Tasks (and components) that are executed by the ARE and/or the GRM.
  - the off-line tracing of the Tasks (and components) that have been executed by the FM, the ARE and/or the GRM.
  - the packaging of Tasks.

- The FORMID module is used by the RM&C Station for the on-line monitoring of the Tasks (and components) that are executed by the FM.

**REFERENCES**


