WATERS 2018 Challenge
RESSAC use case

2018/01/15

RESSAC is a European industrial project on the definition of new certification approaches. To support the study, a use case has been defined on which the certification methodology is applied. The use case, named "µXAV: A Collaborative Development Assurance Lab.", is the baseline of the WATERS 2018 challenge.
All documents relative to RESSAC are available at https://github.com/AdaCore/RESSAC_Use_Case

Table of content

1. INTRODUCTION

The challenge is based on a drone-like muti-system case study, developed by some actors of a collaborative project about aeronautic certification. The WATERS 2018 challenge is focused on end-to-end timing analysis, with emphasis on compositionality.

This document is intended to help the participants to prepare the challenge and to explore efficiently the RESSAC Web Site where the use case is specified and under development until June 2018.

This introductory note is organised as follows: first we give an overview of the web site and present the overall goals of the challenge. Then in Section 2 we give further technical details with emphasis on timing specifications and expected solutions.

1.1 Web Site

The repository of the collaborative project is https://github.com/AdaCore/RESSAC_Use_Case.

The website is structured in two parts corresponding to two processes:

1. *The development* part named *'use case data'* where are stored the development artefacts (specification documents, system architecture model, software models, source codes). A folder is dedicated to each layer of the hierarchical decomposition:
    a. **Layer 0 the air-vehicle level**, i.e the multi-system level: four systems in interaction that control the various physics involved in the drone's behaviour. The SysML architecture model is located at this level,
    b. **Layer 1 the system level**. Layer 1 entails four folders, one per system:
        i. Mission Management System (MMS)
        ii. Electric Propulsion System (EPS)
        iii. Hydraulic Braking System (HBS)
        iv. Maintenance System (MS)
        The maintenance system is not developed yet and should be ignored by WATERS 2018 participants,
    c. **Layer 2: the item level**. In aeronautic jargon an item is either a piece of software or a piece of electronic hardware, to be integrated in some equipment. The layer 2 folder is subdivided into as many folders as software or hardware items under development in the use case. Presently:
        i. MMS software developed with SCADE
        ii. MMS software developed with Spark Ada
        iii. EPS' ECU (Electronic Control Unit) developed as a SoC (System on Chip),
        iv. A case study on hardware COTS

2. *The development assurance process* named 'lifecycle processes'. It contains development plans that describe the parts of the drone's development processes where the new certification method is being tested. There should be no need for the participants to browse these plans, except may be that of the hardware use cases to get some flavour of the context and make technology choices accordingly (not requested).

The most relevant artefacts are:

- Layer 0: “µXAV Architectural Specification”, which gives the detailed logical architecture (i.e without any technological choice on computers, buses, engines etc. and without any 3D mapping of the system resources into the mechanical structure of the drone).
- Layer 0: “µXAV Operational Specification”. It helps understanding how the drone is planned to be operated (scenarios), and what simulation cases look like,
- Layer 0: “SysML architectural model”. It provides the functional and the organic architectures, with the mapping of the former on the latter. It has been developed by ANSYS with SCADE Architect. Free licences (academic program of ANSYS) are available for the challenge participants (contact: jean-louis.camus@ansys.com). An AADL profile may be made available to the interested participants.
- Layer 1: “MMS system requirements allocated to software (increment 1)”. The functions specified in this document provide the tasks to be scheduled and timely executed by the middleware of the mission management system. Several functions can be grouped into the same task, but splitting a function into several tasks should be avoided. At this stage the functions allocated to EPS and HBS that are to be implemented into software are known but not yet specified in detail. From schedulability and timing analysis standpoint, they are to be considered as timing budgets. Increment 1 is the first development increment, focused on control and mission management aspects. WATERS 2018 constitutes increment 4, focused on communication scheduling and real-time dimensioning. The challenge illustrates multi-system industrial developments where some parts are designed in detail whereas some other ones are only sketeched.
- Layer 2: “MMS software developed with SCADE”. Here can be found the SCADE model (free licences available as well contact: jean-louis.camus@ansys.com) and the generated C source code for the participants interested in performing WCET estimation.
- Layer 2: “MMS software developed with Spark Ada”. Similar to SCADE, but for Ada (contact: comar@adacore.com)

1.2 Challenge

The challenge is composed of three independent questions and a solution can consist of answering one or several of them.

1.2.1 Question 1: Refining the Cyber part of the Architecture

The technological choices on communication buses and computing units are left open. The first part of the challenge consists in **choosing of set of concrete hardware resources** to host the three systems and to perform a fully instantiated end-to-end timing analysis multi-system wide. The resources have to be chosen to minimize: mass, energy consumption, cost, and obsolescence risk.

1.2.2 Question 2: End-to-end Timing Analysis

The idea is to propose:

- A decomposition in tasks, their data dependencies, their deadlines and to assume some upper bounds WCET. The WCET of every task is assumed to be given, except when the task's source code is available.
- A specification of the message groups, their ordering dependencies, their activation logics, and their transfer deadlines.

The computation and communication resources on the one hand, the distributed scheduling policies on the other hand, have to be chosen to meet the end-to-end timeliness requirements.

Some failure events may trigger a reconfiguration of the sets of tasks to be scheduled in the three systems, with potential induced transient delays. Detailed analysis of robustness in case of failure is an important aspect of the challenge (safety-oriented – worst case approach to performance analysis).

A compositional approach, when feasible, would be appreciated.

Formal arguments on timeliness are welcome.

1.2.3 Question 3: Simulation or other test / analysis means

System level simulation with software in the loop is considered by the RESSAC project. System modeling is supported by two tools:

- *SCADE Architect* for the system architectures and for the software architectures (static modelling),
- *Simulink* at system and multi-system level (dynamic modelling),

Simulation of the time-critical scenarios after the architectural refinement step (question 1) and the timing analysis step (question 2) is the third objective of the challenge. These time-critical scenarios are related to fault detection (hardware failures or activation of errors), isolation and recovery (fail-safe design). Some of them are presented in the sequel.

Timing analysis conditions control stability. The flight control loop spans over the three systems.

Methods and tools to test the results of questions 1 and 2 are welcome.

2. GENERAL OVERVIEW OF THE USE CASE

2.1 Drone description

μXAV is an air vehicle dedicated to 7/7-H24 autonomous or remotely controlled cargo transport. The payloads are of small size (< 10kg). Flight must be ensured for a large domain of weather conditions and as consequence *availability, reliability, robustness* and *energy efficiency* are the key design drivers. A mission should last less than 15 min.

Drone logic:

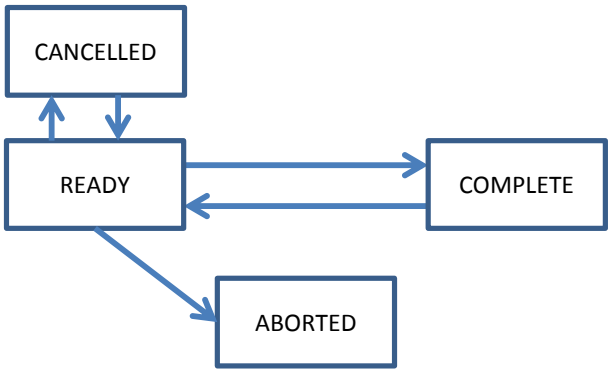


Figure 1: Different states of the drone

At the beginning of the mission, the drone is *ready.* If take-off appears to be unfeasible because of a failure or an energy problem, the drone reaches the state *cancelled*. If the mission starts, either the drone will *complete* the mission in normal situation (landing at the right place in satisfactory conditions), or it will *abort* meaning that the vehicle will land before the waypoint where the payload has to be delivered. When aborting the mission, there are two types of emergency landings: *soft landing* occurs when enough energy and resources are available and control succeeds (anticipated and mastered situation); and *hard landing* occurs when there is shortage of energy or too many failures.

Safety objectives:

- The drone's dynamic must be kept in its safety envelope, anyway. No timeliness incident (periodic or aperiodic deadlines) should lead to control loss, or even hazardous disturbances,
- When in the course of the mission the weather conditions appear to be worse than the standard ones, a mission abortion logic followed by emergency landing prevents drone loss (and ensuing damage or casualties).
- No single failure should lead to the loss of the drone.

Communication with the ground:

- drone to ground:
  - Before the mission: download of the mission parameters set by the field operator,
  - During the mission: the state vector, i.e speed, altitude, position, and the 0/1 status of all failure modes handled by the Health Monitoring functions.
  - After mission completion: a message with the remaining electrical capacities of the two sources.

– Ground to drone:
  - The mode (RP, A) and navigation options (energy efficiency, speed control, altitude control),
  - The navigation parameters: distance, cruise altitude and speed,
  - Possible updates of the mode, parameters or option during the course of the mission.

2.2 Architecture description

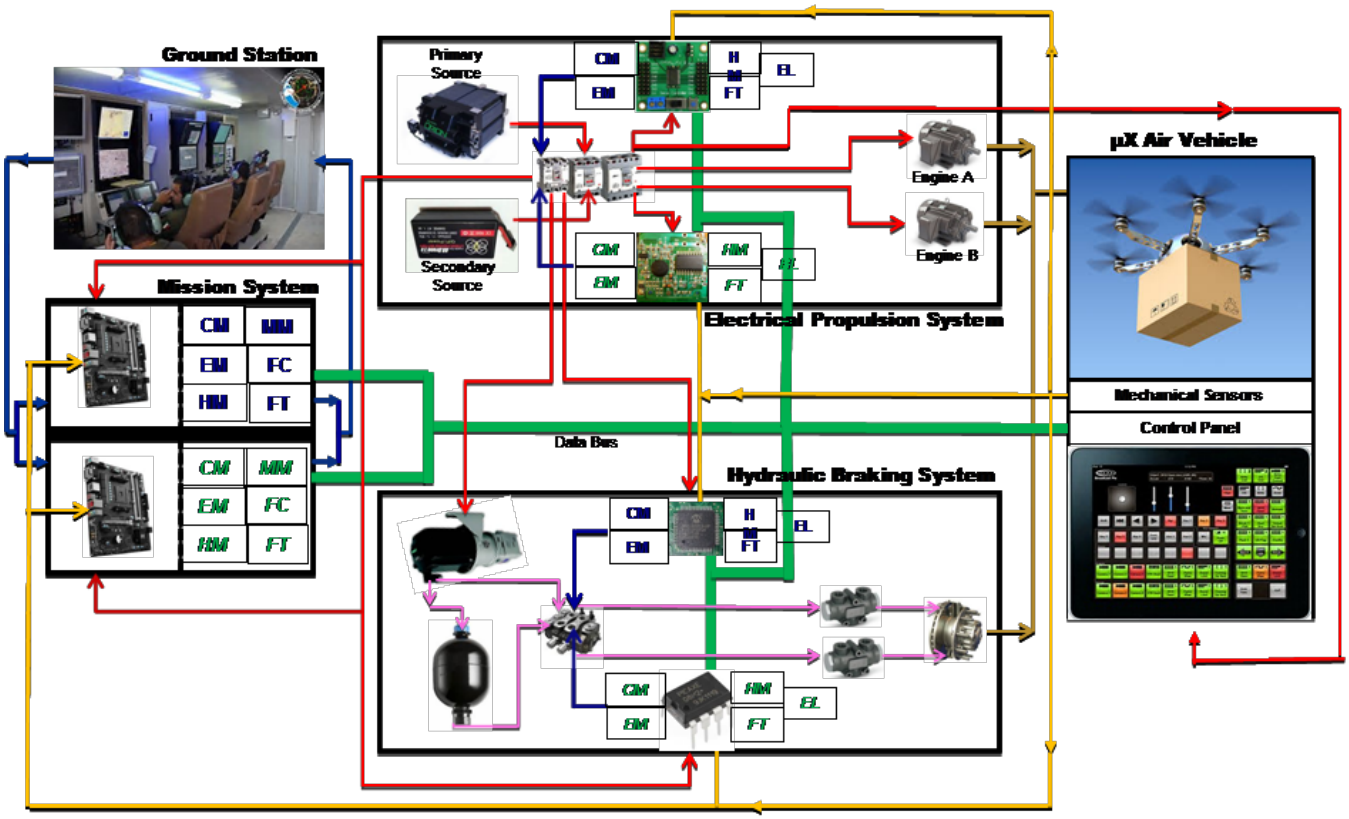


Figure 2: On-board architecture

The three systems are given a 2-redudant architecture at any level (sensing, computation, actuation) to meet the regulatory “no catastrophic single cause failure” objective.

MMS receives the sensor values (pitch, pitch rate and roll) and ensures control of the drone by sending mechanical commands to EPS and HBS (torques). MMS monitors energy, resource availability, and mission progress. Depending on these states, it decides if the mission can proceed or has to be aborted, with soft or hard landing. MMS is also in charge of managing the digital internal and external communications.

EPS accurately and timely converts the mechanical commands from the flight controller (MMS) into voltage commands to the electrical motors. EPS is in slave mode with respect to MMS, it never operates autonomously unless MMS is disabled because of some double failure. In that degraded condition EPS becomes the master, in particular it is in charge of flight control but has no mission management

capabilities. It tries to complete the mission in-progress, with frozen navigation parameters. EPS is also MMS' safety monitor when MMS is in 1-channel degraded mode.

HBS is in charge of degrading the kinematic energy of the drone, i.e of lowering the altitude or the speed. It is also in charge of emergency landing. It has to drive the µXAV to zero altitude and zero speed state, preferably simultaneously. It can only perform braking commands, whatever the wind conditions are. Depending on those conditions, some residual speed at touchdown may be unavoidable.

All the systems are "dual channel" (fail-operational design rationale). The main component of each channel (in MMS, EPS, and HBS) is an Electronic Control Unit (ECU) that runs the software functions (represented as green or blue squared XY on figure 2). The overall list of software functions, i.e of schedulable tasks, is:

- CM: communication management
- EM: energy management
- HM: health monitoring
- EL: emergency landing
- FT: fault-tolerance
- FC: flight control
- MM: Mission Management
- PC: propulsion control
- BC: braking control
- PT: payload transport

See the "MMS software specification (increment 1)" document for a detailed account of EM, EL, FC, MM, PT. *CM is going to be specified in detail, in interaction with the participants to the challenge*.

There are two types of communication: analog (shown in red in Figure 2) and digital (green in Figure 2).

For each ECU one gives (see appendices):

- The list of tasks to be scheduled
- The data dependencies between the tasks
- The task activation logic (time-driven or event-driven)
- The code of the task or its assumed WCET
- The messages to be sent and received for each channel of the end-to-end specification
- The content of the messages

There are no constraints on the choice of ECU hardware. Micro-controllers, mono-core or multi-core processors, are all acceptable solutions. The participants may choose the products they are most familiar with. Real suitability for drone embedding is not an issue. This use case is primarily methodological. The point is not the development of an actual drone. The point is to assess research state of the art in multi-system timing analysis.

Freedom of choice is similar for the communication technologies:

- *Wired communications*: CAN bus, Field bus, Time-triggered solutions, Ethernet solutions, etc.
- *Wireless communications* between the ground station and the drone: any option is open (Zigbee, Wifi, Bluetooth etc.). The participants can choose the technologies they are most at hand with. What matters is their design rationale and the coverage of their analysis.

For the chosen technologies, a thorough analysis of the safety-relevant and timing-relevant aspects are expected (determinism, fault models tolerated and non-tolerated, contentions, timeliness etc.).

2.3 Scheduling constraints

2.3.1 Task scheduling

Task scheduling is performed by the two computations units embedded in each of the three systems.
A task may be assigned to a single function, or to a group of functions sharing the same timing constraints.
For compatibility with the safety aspects of the use case, the atomic functions (i.e non decomposed into sub-functions) must be single execution units (fail-stop failure mode).

FT can be activated on an event-driven basis as it manages reconfiguration transitions to mitigate failure modes (events). The part of CM devoted to communicating with the ground station and with the control panel can also be addressed in aperiodic event-driven mode (operators' interactions).

All the other functions have to be activated on a periodic basis:

System or Device	Task	Period	Jitter max
Electrical Propulsion System	EM, PC, FC, EL, CM, HM	20Hz	30ms
Electrical Propulsion System	FT	aperiodic	
Hydraulic Braking Systems	EM, BC, EL, CM, HM	20Hz	30ms
Electrical Propulsion System	FT	aperiodic	
Mission Management System	PT, EM, MM, FC, HM	20Hz	30ms
Mission Management System	FT, CM.GS	aperiodic	50ms / 1s
Control Panel	CM	10Hz	5ms
Sensors / actuators	CM	50Hz	No jitter
Communication with ground	CM	1Hz	100ms

2.3.2 Communication scheduling

On each of the six computation units, function CM has to convert the analogous (2-redundant) sensor signals into digital values, and the command digital values into electrical signals. Stability of control (FC and EL) is highly dependent on timeliness of these tasks.

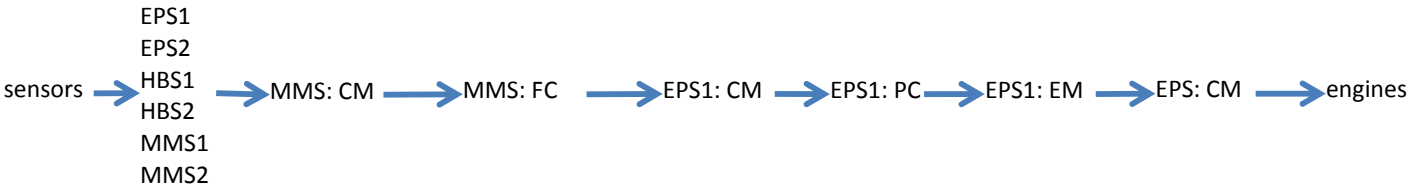
CM is also in charge of bus management. MMS is bus master as long as at least one of its two channels is active. The protocol to be used is technology dependent and left to the participant.

The ordering dependences of messages and their logical format (groups of data to be transmitted at once) will be provided in the appendix. The "MMS system requirements allocated to software" document provides a detailed account of the messages to be transferred from/to the ground station and the control panel.

2.4 End-to-end temporal constraints

These constraints are given on different channels

Channel 1: nominal command
The first channel should last at most 20ms.



Channel 2: nominal situation and altitude change request from the ground

The ground may request to change some fight parameters such as the altitude or the speed.

The message from the ground station is received by CM and MM checks if the remaining energy level is sufficient to apply the request. If no, the request is simply ignored, otherwise it is sent to FC and the associated new regulation is performed. Then, the maximal latency between the reception of the message and the order application by EPS should be less than 100ms.

Channel 3: nominal situation and energy management

During the flight, the MMS must be aware of the energy level to decide mission abort and emergency landing if needed. The active lane of EPS (ECU1 or ECU2), abbreviated EPS1, EPS2, transfers the measured energy levels to MMS.

The channel should be realized every 1s.

Channel 4: degraded mode where MMS not in charge
This occurs when the MMS does not manage the mission any more. This situation may happen when both lanes of MMS'computer are down or the digital communication is not working. In that case, EPS becomes the new supervisor and bus manager.

Channel 5: EPS1 auto-tested as down inhibits itself and hands over to EPS2

Thanks to the health-monitoring, each system is able in some situations to detect itself as failed. EPS1 for instance has 100ms to detect failure, inhibit itself and broadcast the failure event to all the systems that will reconfigure accordingly.

Channel 6: reconfiguration in case of back-up failure

After MMS double failure EPS1 (back-up of MMS) is the master and EPS2 monitors its behavior. When EPS1 fails, EPS2 becomes the new master and HBS1 becomes the new monitor. EPS1 failure is either self-detected (auto-tests) or detected by the safety monitor (EPS2). The component that detects the failure sends the signals and messages that trigger the distributed reconfiguration transition.

Such a reconfiguration must take at most 100ms. If EPS1 cannot detect itself as failed, EPS2 must inhibit EPS1, become the new master and rely on HBS1 as safety monitor. The reconfiguration channel is given below.

In case EPS2 also fails, HBS1 becomes the master and HBS2 its monitor.

An emergency landing is launched. This reconfiguration should also last at most 100ms.

Channel 7: soft landing

The soft landing is managed by MMS or EPS. The longest event path, given below, corresponds to the case where MMS is still the supervisor:

This distributed transition should last at most 100ms.

3. APPENDIX

More details on the task and communication scheduling constraints.

AOB needed by the participants to the challenge.

To be supplied later, on request.