SPARC – Description

Project description

Modern aircraft will rely more and more on artificial intelligence at the interplay between machines and humans to improve overall system performance. Consider an aircraft that learns from its pilot and adapts its behavior to maximize flight and pilot performance. The aircraft becomes aware of the needs and individual behavior of each specific pilot, can anticipate the pilot's reactions to different flight and threat situations, and prepares and adapts information visualization and actuation for the pilot. In such situations, a pilot cannot simply jump into an aircraft and fly it. The profile of the pilot must be loaded and integrated on each aircraft for each specific mission.

You are required to design and develop a software system that (1) satisfies a set of user needs related to this context and (2) is maintainable in unknown future scenarios. (Note that several aspects have been simplified and abstracted to make the student competition feasible.) For simplicity, three use cases are defined (ref. Figure 1 and Table 1).

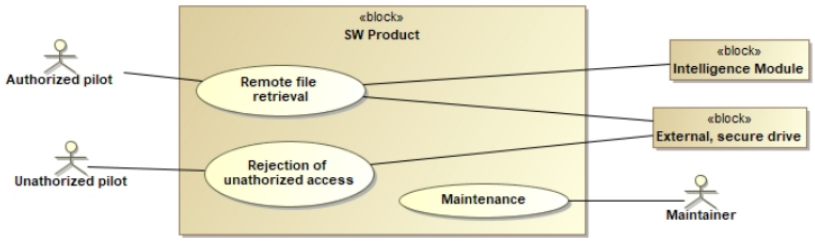

Figure 1. Use Case Diagram

Table 1. Description of Use Cases

Use case	Pre-conditions
Remote profile retrieval	A pilot requests access to fly an aircraft for a specific mission. The pilot's profile, which is stored in an external, secure drive, is authenticated by the aircraft and is loaded into its intelligence module. The pilot receives a confirmation of the pilot and mission profiles that have been loaded to the aircraft. *Pre-/Post-conditions*: Hardware is available and correctly working. External drive is connected to the aircraft computer and recognized by its operative system. The aircraft system is pre-populated with a database of authorized pilots and mission profiles and their authentication data.
Rejection of unauthorized access	A pilot attempts to gain access to fly an aircraft and/or in a mission he/she is not authorized to fly. The aircraft detects the pilot-aircraft-mission is unauthorized and notifies the pilot and a security officer about the attempted access. *Pre-/Post-conditions*: Hardware is available and correctly working. External drive is connected to the computer and recognized by the operative system. The system is pre-populated with a database of authorized users and their authentication data.
Maintenance	It is described later in this document. *Pre-/Post-conditions*: The SW product has been delivered.

Because the students will not have access to aircraft and pilots, the following surrogates will be used:

- The aircraft will be simulated by a regular computer.

- The external, secure drive will be simulated by an external drive connected to the computer through a USB-C port.
- The Intelligence module will be simulated by an external drive connected to the computer via a USB-C port.
- The user interface of the pilot will be simulated by a phone with SMS capability.

This project does not impose any restrictions on the development environment, tools, and programming languages that you use. Similarly, you can define your own development and verification, validation, and test plan.

If you have questions about the information contained in this document or the competition in general, you may address them to SPARCChallenge@gofirepoint.org.

User requirements

UR-10 The pilot needs to load his/her profile into the Intelligence Module from an external, secure drive.
Note 1: The profile is in an .xlsx file stored in an external drive. The external drive is configured in NFTS and connected to a computer through USB 3.0.

UR-20 The pilot needs to load his profile remotely.
Note: Remotely means that the pilot will not interact directly with the aircraft (computer hosting the software), the external drive containing the profile, or the Intelligence Module. Profile loading will be requested via external command and the confirmation delivered to another external device.

UR-30 The pilot needs to initiate profile loading by sending a request via SMS.
Note: The information and format of the command are not prescribed, but it must include the identification of the pilot and of the mission he/she is intended to fly in. The SW developers must define such interface.

UR-40 The pilot needs to receive confirmation of profile loading in their phone through SMS.
Note 1: Profile loading begins when the pilot sends a command to load his/her profile and ends when the pilot receives the confirmation. The format of the confirmation is not prescribed. The SW developers must define such interface.
Note 2: Assume a phone that only has SMS capability.

UR-45 The pilot needs to receive confirmation of profile loading only after his/her profile has been loaded to the Intelligence Module.

UR-50 The pilot needs to complete profile loading in less than 25 s.
Note: Profile loading begins when the pilot sends a command to load his/her profile and ends when the pilot receives the confirmation.

UR-55 The system shall only upload profiles of authenticated pilots.
Note: No authentication protocol is prescribed. The SW developers must define such protocol.

UR-60 The system shall deliver profile loading confirmation only to authenticated pilots.
Note 1: Timing restriction in UR-40 includes any time necessary for authentication purposes.
Note 2: No authentication protocol is prescribed. The SW developers must define such protocol.

UR-70 The pilot shall complete profile loading without requiring any device other than their phone.
Note 1: Timing restriction in UR-40 includes any time necessary for authentication purposes.
Note 2: As per UR-40, assume that a phone that only has SMS capability.
Note 3: No authentication protocol is prescribed. The SW developers must define such protocol.

UR-80 The pilot shall complete profile loading without requiring any interface other than as defined in UR-30 and UR-40.

UR-100 The system shall notify the pilot that profile loading is not available if the profile loading request cannot be initiated.
Note: For example, when the request surpasses the maximum capacity of the system. This will be referred to as unavailability notification *and/or* notification of unavailable service*.*

UR-110 The system shall complete unavailability notification in less than 5 s.
Note: Unavailability notification begins when the pilot sends a command to load his/her profile and ends when the pilot receives the unavailability notification.

UR-120 The system shall notify the pilot of unavailable service without requiring any device other than their phone.
Note 1: As per UR-40, assume that a phone that only has SMS capability.
Note 2: The unavailability notification protocol is not prescribed. The SW developers must define the unavailability notification protocol.

UR-130 The system shall notify the pilot of unavailable service without requiring any interface other than as defined in UR-30 and UR-40.

UR-140 The information in the external drive shall remain unaltered after completion of a profile loading.

UR-150 The system shall reject every profile loading request after at least 3 consecutive requests have failed to authenticate.

UR-160 The system shall operate in the latest version of Windows 11 running on an Intel i5 2GHZ machine or higher, with 4 GB of RAM.
Note: While the computer is operating at 50% load.

UR-170 The system shall use less than 2 GB of hard drive space.

Maintainability

Requirements or a specific measure for maintainability are not provided. You should be aware that actual maintenance scenarios are unknown at the time of designing and developing the system. You must architect and develop your system aiming at facilitating unknown, future maintenance scenarios.

Maintenance scenarios are defined as those that could be constructed using the structure presented in Table 2.

Table 2. Template to define maintainability scenarios

Element	Description	Options
Stimulus	Source that triggers the need for maintenance.	*Corrective change*: Request to correct an error in the code. *Perfective change*: Request to modify the quality of a function (e.g., more efficient, faster calculation, fewer resources used…). *Adaptive change*: Request to operate in a changed environment. *Preventive change*: Request to replace a software element with a new version (e.g., apply a security path)
Artifact	The part of the SW that is affected by the maintenance activity.	Single SW element, multiple SW elements, or the entire SW system.
Actor	The entity performing the maintenance activity.	The student team or someone else.
Measure	Consequences of the maintenance activity.	Size of the changes, invested effort, calendar time, extent to which other functions or capabilities are affected by the maintenance activity, and introduction of new defects.

Deliverables

Each student team shall deliver the following artifacts:

- D1. *Design Description and Justification Report*: An MS Word document that describes both
    - (1) the design of the SW product and
    - (2) the justifications for the different choices and trade-off's made by the team.

  The report shall include a dedicated section that explains in detail what architecture and design choices the team has done to make the solution maintainable. For each architectural choice of the SW, describe in separate paragraphs the what (that is, a description of the architectural aspect) and the why (that is, the intentionality or what you wanted to achieve by implementing such architectural aspect.
- D2. *Source code*: All code developed and necessary to run the SW product, in a ZIP file. Please, use good commenting practices to ease review of your code.
- D3. *Architecture models*: All architecture models developed by the team. These may be any combination of diagrams, UML models, etc. The architecture models must incorporate (within the models) descriptions of key architecting and coding decisions that have been made to make the solution maintainable, as well as the main decisions that were driven by performance drivers derived from the user requirements. If dedicated architecting tools have been used, please provide the source files as well as a comprehensive export in pdf format. For each architectural choice of the SW, describe in separate paragraphs the what (that is, a description of the architectural aspect) and the why (that is, the intentionality or what you wanted to achieve by implementing such architectural aspect.
- D4. *Verification and validation evidence*: An MS Excel table that lists for each user requirement and or maintainability the following information:

- *State of compliance*. This is, whether the requirement is satisfied or not.
- *Evidence*. This may be the identification of test data, simulation results, or any other document that contains evidence of the requirement being satisfied. All such information must be also delivered as a ZIP file.

Evaluation

The goodness of the SW products delivered by the student teams will be evaluated by a panel of architecture experts representing government, industry, and academia. Evaluation will be performed according to the rubric in Table 3.

Table 3. Rubric.

Scoring criteria	Points
Satisfaction of user requirements. Evidence is provided on the satisfaction of the user/system requirements. Total number of points will factor both the number of requirements that are satisfied, and the confidence provided by the verification and validation evidence that is presented.	33
Maintainability. The product responds adequately to maintenance scenarios that will be identified by the panel of judges at the moment of the evaluation. Reminder: These scenarios will not be known during the architecture, design, and development of the SW product by the student teams.	33
Documentation. The documentation provided by the teams provides detailed justification of how architectural and design decisions have addressed maintainability.	34