Simple CASE Example

# Reference Documents

This example includes the following documents:

|  |  |
| --- | --- |
| **Document Title** | **Description** |
| CASE Example.doc (this document) | Description of a simple CASE example in the context of our envisioned tool workflow |
| CASE Example.ppt | Detailed description of example architecture and requirements |
| UAV.aadl | AADL model of example architecture |
| UAV.aadl\_diagram | Graphical AADL model of example architecture |
| WaypointManager.c | Source code for Waypoint Manager functionality |
| WaypointManager.h | Header file for Waypoint Manager |
| Waypoint.h | File defining Waypoint structure |
| Location3D.h | File defining a 3D location (used by Waypoint.h) |
| WaypointManagerFunSpec.thy | Isabelle/HOL theory file for Waypoint Manager |
| WaypointManager\_TestHarness.tar.gz | Waypoint manager source with test harness \* |

\* The Waypoint Manager code is a stripped down version of a UxAS component. We are also including a version that contains a test harness and the AFRL Common Mission Automation Services Interface (CMASI) library. This version is not intended to be part of the simple example, but is included for those that would like more insight into how such a system could be implemented.

# Introduction

In order to gain alignment around tool interfaces and interoperability, we introduce a simple example architecture of a UAV surveillance system. In our scenario, a UAV is used to conduct surveillance over a specified region. A ground station transmits a map and flight pattern to a UAV, from which a flight mission is generated and executed by the flight controller. The map may be annotated with no-fly zones and other features relevant to the mission. The flight pattern is a description of the intended UAV behavior, such as *zig-zag across surveillance region*, or *follow river*. The Flight Planner on board the UAV Mission Computer takes the map and flight pattern input and generates the flight mission, which is a list of waypoints the UAV must follow. The Waypoint Manager passes the current window of waypoints to the UAV Flight Controller.

A high-level view of the system is illustrated in Figure 1.

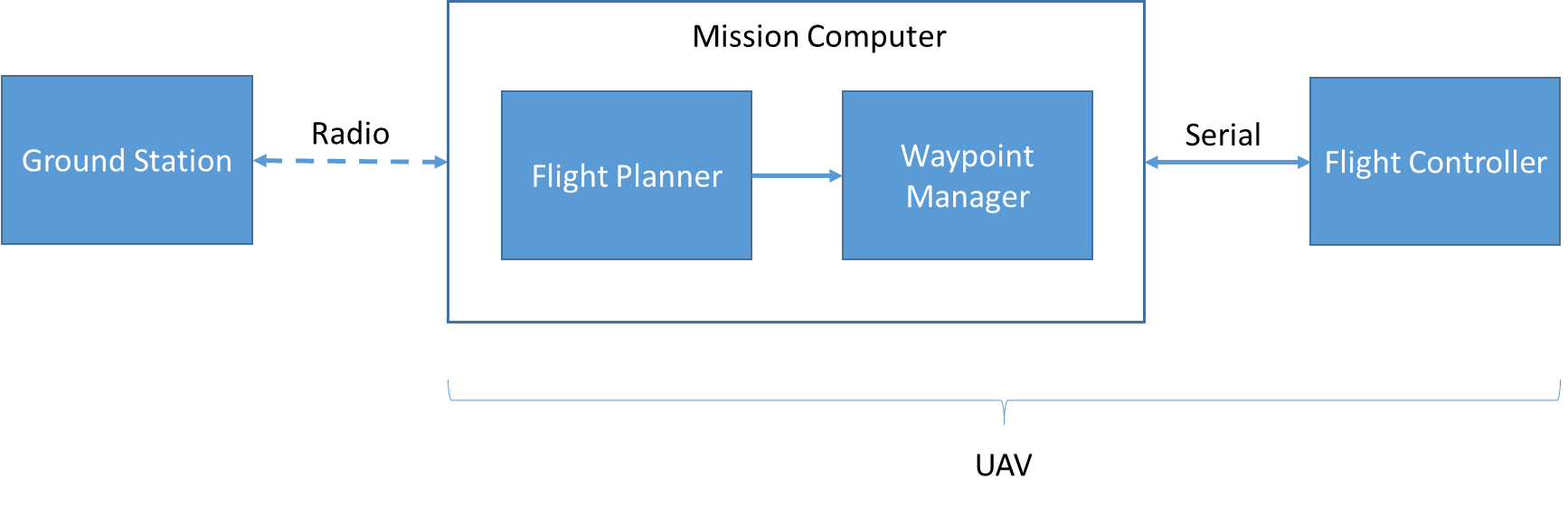


Figure 1. Waypoint Navigation System

Our example includes some high-level requirements, an AADL model of the system architecture, and a legacy code component for waypoint management. Our intention was to create an example with enough detail to be useful to all TA performers, but simple enough that we do not get lost in too many details at this early stage. We expect this example will not only clarify tool interfaces and expected data input/output between TA performers, but also expose interoperability issues early in the project, giving respective groups more opportunity to address them.

# Using CASE Tools to Develop a Waypoint Navigation System

To illustrate how we envision incorporating output from TA 1-4 performers into a common engineering framework, we present a hypothetical scenario about how our workflow and other CASE technologies could be used by a TA 6 performer. Figure 2 illustrates the TA interactions. A TA 6 systems engineer, using the CASE integrated tool suite (TA 5), plans to design a UAV waypoint navigation system to be cyber-resilient (or perhaps alternatively, re-engineer an existing implementation to improve its cyber resiliency).

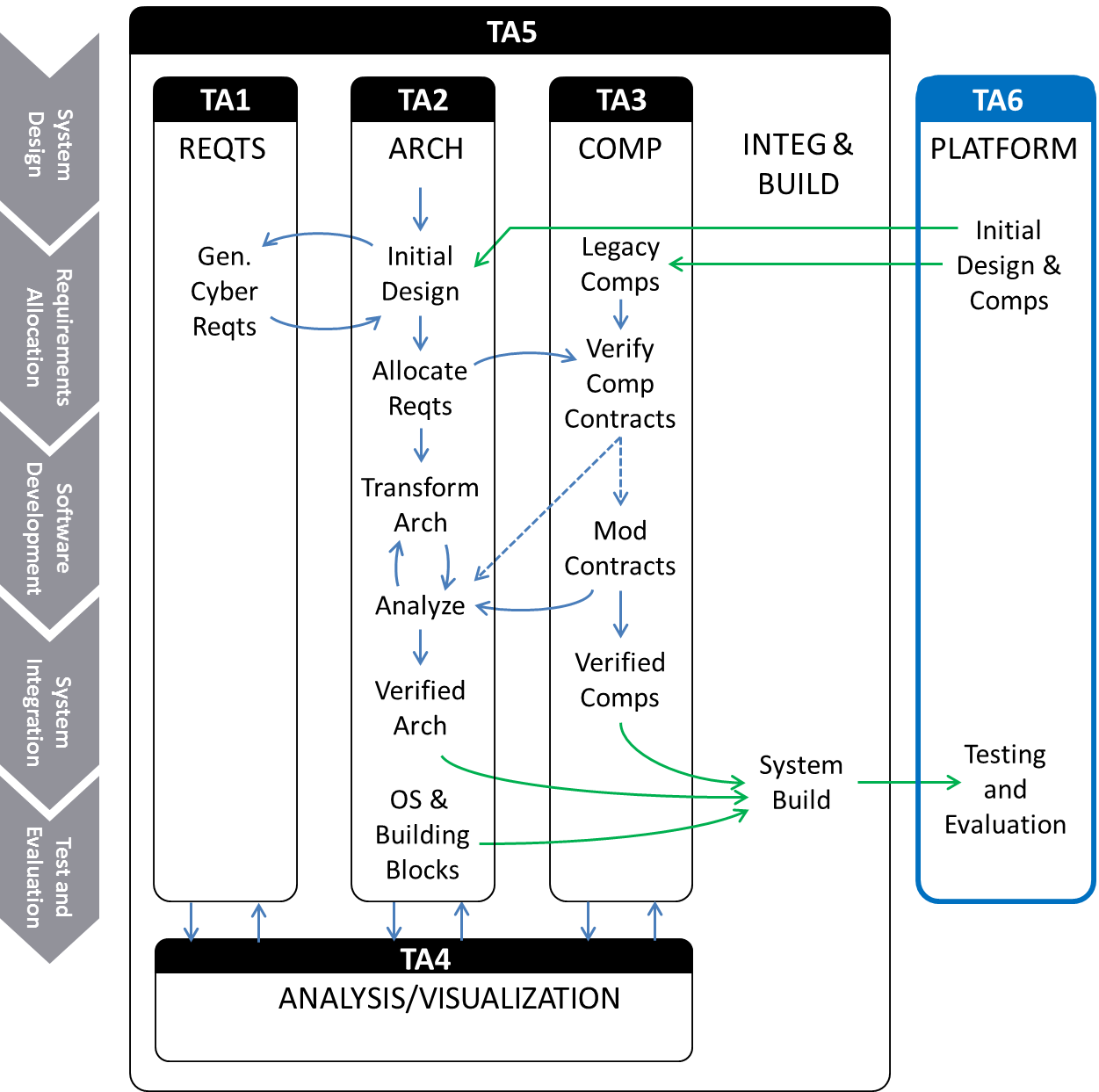


Figure . Interactions between TA tools

The TA 6 engineer starts with an architecture of the system. The architecture model is created using the TA 2 architecture tool. Architecture development is an iterative process. Initially, the system architecture will be driven from a set of high-level functional requirements (some of these requirements are included with our example), or possibly generated from an existing behavioral model (i.e., Simulink). For this example, we have created an AADL model of a simple UAV waypoint navigation system architecture, as pictured in Figure 3.

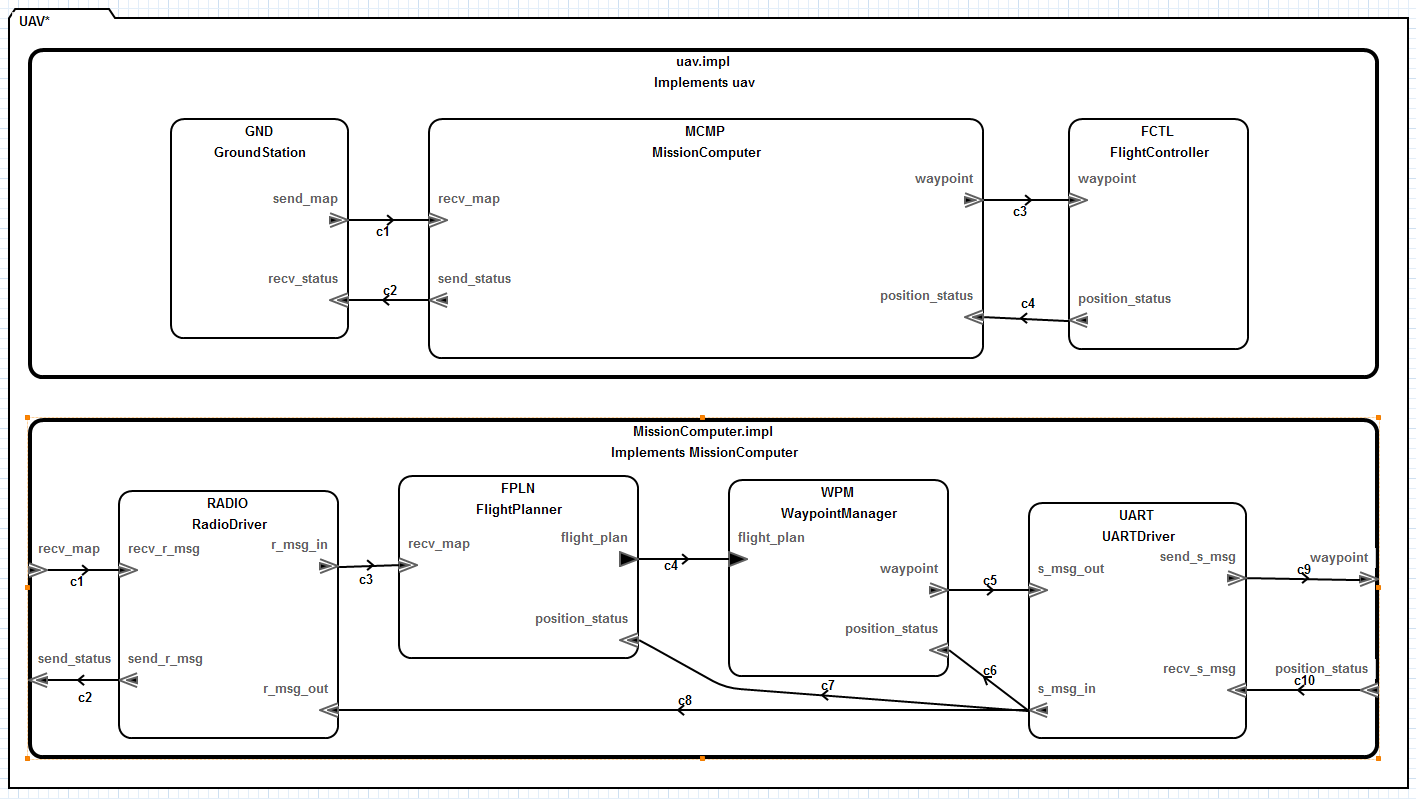


Figure . AADL model

The architecture (and possibly the functional requirements that drive it) are then passed to TA 1 tools. The TA 1 tools perform analyses that lead to the generation of cyber requirements. The cyber requirements are consumed by TA 2 tools, and the system architecture is modified in response. This interaction between TA 1 and TA 2 may consist of multiple iterations until the refined architecture no longer results in generation of new cyber requirements from TA 1.

In the case where there are legacy software components that will be used in the implementation of this system, the TA 2 design tool will determine which TA 1 requirements are allocated to these legacy components and TA 3 tools will perform adaptations to the code and analyses that verify the code satisfies the TA 1 cyber requirements. In our example, the Waypoint Manager (source code provided) is one such legacy component.

The underlying analysis engines that TAs 1-3 utilize will be provided by TA 4. These engines will provide feedback in the same semantic context that is being used at the design level.

TA 1-4 tools will be integrated into a holistic development framework (TA 5) that will facilitate tool interoperability and present the TA 6 engineer with a useful cyber resilient systems development interface. The TA 5 tool will enable the TA 6 engineer to perform analyses on the system level, generate design assurance cases, and provide the appropriate mechanisms for building the system.