ED-12C Compliant Autonomous Decision Making for UAVs

Category: abstract of regular paper;
Authors: Nick Tudor and Colin O'Halloran of D-RisQ Limited
Keywords: ED-12C/DO-178C, Certification, UAVs, Cost savings, Formal Methods, BVLOS

Introduction

The project exploited core skills from the combined project team of D-RisQ and Callen-Lenz which ranged from software design, verification and validation as well as Unmanned Air System (UAS) control systems. Through our high integrity approach to design, proof and testing we addressed the system verification, validation and certification challenge, looking to adopt a similar approach to that used for civil manned aircraft certification using international standards such as ED-12C/DO-178C and specifically the formal methods supplement ED-216/DO-333. Both project partners have had discussions with the UK Civil Aviation Authority on UAS certification for beyond line of sight or over the horizon operations and have contributed to various internationally recognised publications related to software and UAV/drone certification.

The core of the project examined the challenge of providing a level of integrity (appropriate to the risk of planned operation) for a cognitive system for UAS autonomous behavior. The current state of the art does not allow regular Beyond Visual Line of Sight (BVLOS) operations of UAVs because there is a major problem of assuring collision avoidance decision making. While we assumed adequate sensing for this project, some of which was also trialled as part of this project, the sensing aspect is out of scope for this paper. An increase in both capability and safety would allow commercial UAS operators and users to undertake more complex tasks with increased capability. In addition, we examined the cost savings through the use of two tools. The first tool supported the articulation of requirements for embedded reactive systems, such as a control system. The tool was designed to make requirements reviewable by a developer and hence by a certification authority, but precise enough to be guaranteed to be verifiable. The second tool automatically verified a Simulink/Stateflow® design against the requirements using off-the-shelf verification tools, if a verification failed a script was generated to explain the non-compliance using simulation in Simulink®. We addressed the issue of hazard avoidance within a near vicinity and Callen-Lenz addressed the task prioritisation and planning. This paper will examine the overall approach while focussing on the verification technology as the key enabler for BVLOS operations at a price the market can afford.

Background and Foundation Work

The foundations of the approach taken were originally used in some automotive work under and ISO26262 ASIL D project (PICASSOS). The techniques included the use of a formalised expression of the requirements. This required specialist skills to write the specification even though the training for this was relatively light and engineers were able to use the technique within a few days of on-line and remote assistance training. The tools were also somewhat under-developed and required some human intervention to get results. Nevertheless some very positive results were obtained in terms of error detection and speed of use (more on this aspect later – see 'Costs'). We then deployed the techniques to provide decision making software for an autonomous unmanned surface vessel to ensure compliance to the rules-of-the-sea for collision avoidance (known as 'COLREGS'). The system requirements were a little under-developed, but again it was shown what was possible with successful sea trials. We wanted to then push the work into the highly challenging air domain.

Project Overview

The project was staged into 2 main phases with the initial capability aimed to produce a design for the systems and software architecture along with a first stage decision making capability. The purpose of the first Phase was to explore potential designs and validate requirements using simulation and was not expected to be

flown. This approach enabled a low cost path to quickly explore behaviour. The second Phase would then adjust the previous design decisions made and add extra functionality that would enable a more sophisticated behaviour. The tools from D-RisQ were also developed alongside the UAV software development. The main demonstration for the project was to fly 2 UAVs at each other, one with the decision making software flying on a pre-determined route and autonomously making decisions, the other without any of that software but simply flying on an opposite track. A third UAV was fitted with a camera which was fully in the control of a ground based operator. Finally, a series of increasingly complex scenarios were to be flown in simulations.

Formal Specification Selection

Within ED-216/DO-333, there are 4 additional Objectives that have to be met to use a formal technique. The main one of concern is focussed on justifying that the formal specification language and analysis techniques can deliver results that would support the certification Objectives. We have chosen CSPm for requirements and design because of the natural compositionality of the language. It enables representation of the environment in which a system (or software is intended to interact and can in principle be used to exhaustively check that desired properties hold and even that undesired properties are not present. There are well-known practical limitations with the use of model checking which can be avoided if the formal description is carefully designed-in to enable scalability. This is therefore how we ensure that the process can be automated. CSP was first described in a 1978 paper by C.A.R Hoare1. Professor Sir Tony Hoare (one of the world's foremost computer scientists) developed CSP, a formal mathematical language for describing the patterns of interaction between systems operating together, such as parallel processors, computers talking to each other, and the components of an individual microchip. CSP now has decades of academic development and industrial use behind it. There have a been a number of publications2,3 and the book Communicating Sequential Processes (CSP) is available for free download at: http://www.usingcsp.com/. The other 3 Objectives are mainly a part of tool qualification and are not discussed in detail in this paper and a summary of applicable the references from ED-216/DO-333 are in Table 1 below.

Objective		Reference Section	
		Objective	Activity
FM.A-3.8, FM.A-4.14, FM.A-5.10	Formal analysis cases and procedures are correct	FM.6.3.6.a FM.6.3.6.b	FM.6.3.6
FM.A-3.9, FM.A-4.15, FM.A-5.11	Formal analysis results are correct	FM.6.3.6.c	FM.6.3.6
FM.A-3.10, FM.A-4.16, FM.A-5.12	Requirements formalisation is correct	FM.6.3.i	FM.6.3.i
FM.A-3.11, FM.A-4.17, FM.A-5.13 FM.A-7-10*	Formal Method is correctly defined, justified and appropriate	FM.6.2.1	FM.6.2.1.a FM.6.2.1.b FM.6.2.1.c

Table 1 - DO-333 Additional Objectives

However, in summary, it is required to show that the formalisation process is correct, that the analysis cases are correct and that the results are correct. None of these are necessarily easy to accomplish and require a considerable amount of evidence to show that, in particular, the formalisation process is correct.

[1] Communications of the ACM Volume 21 Issue 8, Aug. 1978 Pages 666-677

[2] A.W. Roscoe. "The Theory and Practice of Concurrency", Prentice-Hall (1997)

[3] A.W. Roscoe. "Understanding Concurrent Systems". Springer (2010)

Requirements and Design

This was an experiment to explore the art of the possible but with the intent of this being a final confirmatory project that the tools could work, the decision making software could be built and verified using the tools, all with the necessary support material for certification. A lot of detailed contextual investigation was undertaken to ensure that we had correctly identified the environment in which the UAV would operate. We specifically needed to be able to control the inputs to the decision making as we had to be sure that the aircraft was reacting correctly to the right inputs. As such we used only a single input from ADS-B as the source for location information of possible conflicting objects (airborne or otherwise). It is acknowledged that other sensors would be required in various parts of the spectrum to enable beyond visual line of sight operations in practice.

A Systems Requirements Document (SRD) was developed and agreed for the decision making software for Phase 1. This was then transformed into a Software Requirements Specification (SRS) to describe the software High Level Requirements (HLR). This automatically gives a formal semantics to the natural language expressions in machine readable Communicating Sequential Processes (CSPm) and enables compliance to a number of ED-216/DO-333 Objectives and, in particular, enables a verification of consistency as well as compliance to Requirements Standards. It also enables the next step of verification to be automated.

The SRS was then used to develop a design (Low Level Requirements - LLR) in Simulink. A tool was used that automatically provides a formal semantics to a subset of Simulink /Stateflow in CSPm and it then uses the formal semantics of the HLR to automatically check that the design satisfies the requirements, or show where there is a mistake. Note that the tool user does not get to see the formal semantics, just the natural language requirements, the design in Simulink and the results of the formal analysis in a form such as simulation scripts to show where non-compliances exist. There are some requirements that cannot be formally verified, for example, 'The software shall be developed to ED-12C/DO-178C Level A" is not formally verifiable and hence there remain review aspects to requirements. Once we were satisfied that the design satisfied the requirements, we auto-coded to a subset of C and this was loaded onto the simulator and subsequently on to a UAV. It was agreed that with some simulation with hardware in the loop that we could forego formal software test for the purposes of the experiments.

Project Technical Outcomes

Phase 1 – Basic behaviour and Trials

The project went so well that the Phase 1 software was flown in hardware in the loop simulations and then in flight trials 6 months early, firstly with a static obstacle. Figure 1 shows the route taken by the UAV – this is not a simulation but actual flight trial data overlaid onto map data. This flight trial showed that the behaviour operated as designed and as expected because the manoeuvre taken can be explained. Having explicable behaviour is crucial to making the case for safe flight of autonomous unmanned aircraft. Figure 1 shows that the UAV took-off and headed to waypoint 1 and then headed automatically for waypoint 2. As the UAV moved it encountered the static ADS-B transmitter (in red) which encroached upon the limits for the high integrity decision making software. The software autonomously decided that the best behaviour would be a 90^0 right turn in accordance with the Standardised European Rules of the Air (SERA). Once collision was no longer a threat, the decision making software handed back to the low integrity path planning software to get back on course. However, the ADS-B transmitter was still encroaching and a subsequent manoeuvre was commanded. Again, once the manoeuvre had been completed, the path planner was given control and the UAV completed the route. This is very basic unrefined behaviour and not exactly what would be required for a commercial system. The behavioural characteristics can be refined so that, for instance, a second manoeuvre would not have been required as in this initial case. Subsequently a dynamic trial with 2 UAVs on a collision course was undertaken. A video of the behaviour was produced with one UAV autonomously avoiding the other in accordance with SERA. We believe that this is the only trial of this nature.

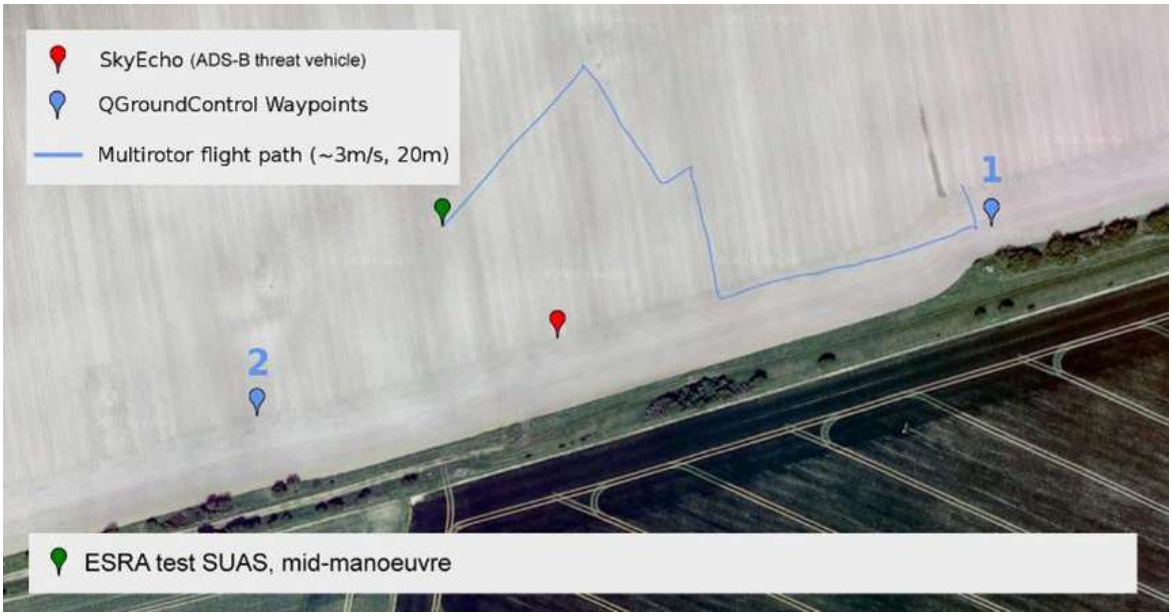

Figure 1 - Initial Flight Trial with Static Obstacle

Phase 2- Advanced Behaviour

The project progressed to Phase 2 behaviour starting with a new SRD. This not only refined the simple Phase 1 behaviour but crucially added a significant enhancement. This new functionality enabled the UAV to anticipate options should it need to manoeuvre; this was called the "Airmanship Package". Furthermore, we allowed the UAV to undertake manoeuvres that a pilot might make but would not necessarily be in accordance with SERA. Again, an SRS was produced, a design was produced using Simulink/Stateflow® and subjected to formal analysis. Code was auto generated and a series of simulations were undertaken with increasingly complex scenarios stressing the software, including some that required formation flying; see Figure 2 - Successful Complex Scenario SimulationFigure 2. Note that no claims for certification credit for the software verification are made as a result of simulation. In all cases, the autonomous UAV avoided all obstacles wherever possible in accordance with SERA or, where this was not possible, through use of the Airmanship Package.

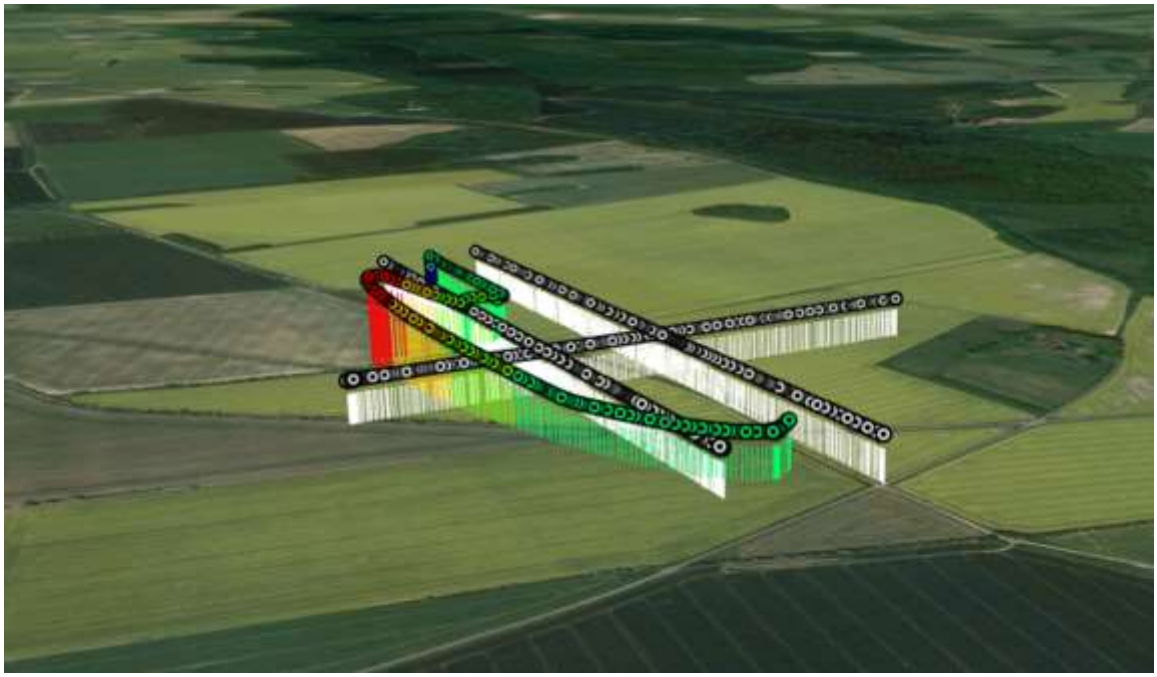

Figure 2 - Successful Complex Scenario Simulation

Certification - Software

The approach was heavily tool supported and because of the complexity of the problem, the tools were significantly enhanced throughout the project in order to cater for the breadth of required functionality. These tools were subjected to a ED-215/DO-330 development process to meet a minimum Tool Qualification Level 5 as they are Criteria 3 tools. The requirements tool, in its present form enables automated compliance to 3 of the 7 Table FM.A-3 objectives and partial compliance to a further one objective. A summary of the claims is in Table 1; note that the additional objectives are not shown. The following key gives the guide to colours.

Not met | Partially met | Fully met

	Objective		Activity	Claim
	Description	**Ref**	**Ref**	
1	High-level requirements comply with system requirements.	FM.6.3.a FM.6.3.1.a	FM.6.3.1	Manual review needed until Kapture® for System requirements is developed when it will likely be a partial compliance for derived requirements.
2	High-level requirements are accurate and consistent.	FM.6.3.b FM.6.3.c FM.6.3.1.b	FM.6.3.1	Basic functionality of Kapture supports accuracy claim; extra functionality gives consistency and unambiguity.
3	High-level requirements are compatible with target computer.	FM.6.3.d FM.6.3.1.c	FM.6.3.1	Kapture does not support this aspect: manual review
4	High-level requirements are verifiable.	FM.6.3.e FM.6.3.1.d	FM.6.3.1	Kapture requirements are verifiable due to the provision of semantics.
5	High-level requirements conform to standards.	FM.6.3.f FM.6.3.1.e	FM.6.3.1	Kapture encapsulates a requirements standard

6	High-level requirements are traceable to system requirements.	FM.6.3.g FM.6.3.1.f	FM.6.3.1	Manual review of manually entered data.
7	Algorithms are accurate.	FM.6.3.h FM.6.3.1.g	FM.6.3.1	Algorithm accuracy can be partially shown through the use of Kapture

Table 2 - Extract from DO-333 Annex A Table FM.A-3

It is assumed that the techniques would be applied to a Level A development, but of course they would equally be applicable to other levels. The design verification tool Modelworks enables automated compliance to 10 of the 13 Table FM.A-4 Objectives and partial compliance to the remainder. The partial claims are specifically because derived requirements and the non-formally provable requirements will need review and there are some aspects relating to the target computer which are better being reviewed.

Other Aspects of Certification

The main aim of the project was to develop and fly representative decision making software that had been verified as far as the design step and this was achieved. A secondary aim was to develop the safety case for the operation of the UAV in beyond visual line of sight conditions and to understand the art of the possible. This second aspect showed that we could develop software that would enable manoeuvres that are in strict compliance to SERA. While these were 'safe' and recognisable as being in compliance with SERA, they are, in some circumstances, not what a pilot would do. SERA is open to interpretation, hence the large amount of training a pilot receives to ensure that the intent behind rules are understood and can be rationally acted upon. It was consequently also shown that in a looser interpretation of SERA we could develop software that would give behaviour that would be recognisable to any pilot. This has given us the ability to have a useful discussion on 'required' behaviour with regulators. It has also given us a future development path for this type of software.

Cost Savings

As outlined earlier, a previous independent industrial scale experiment was carried out. Both prototype tools were subjected to cost analysis on a safety critical project. While this project was conducted under ISO26262, all tools and practices were also required to comply with a development that would support ED-12C/DO-178C and qualification under ED-215/DO-330. The project was blind seeded with 48 errors across the 6 modules which represented some 500 Simulink/Stateflow blocks; the participants did not know where or how many errors were inserted. Against a benchmark of using standard review processes as well as against another tool (SLDV), the time taken to undertake the equivalent processes to comply with the relevant objectives was measured and the results were compiled into the graphic at Figure 3.

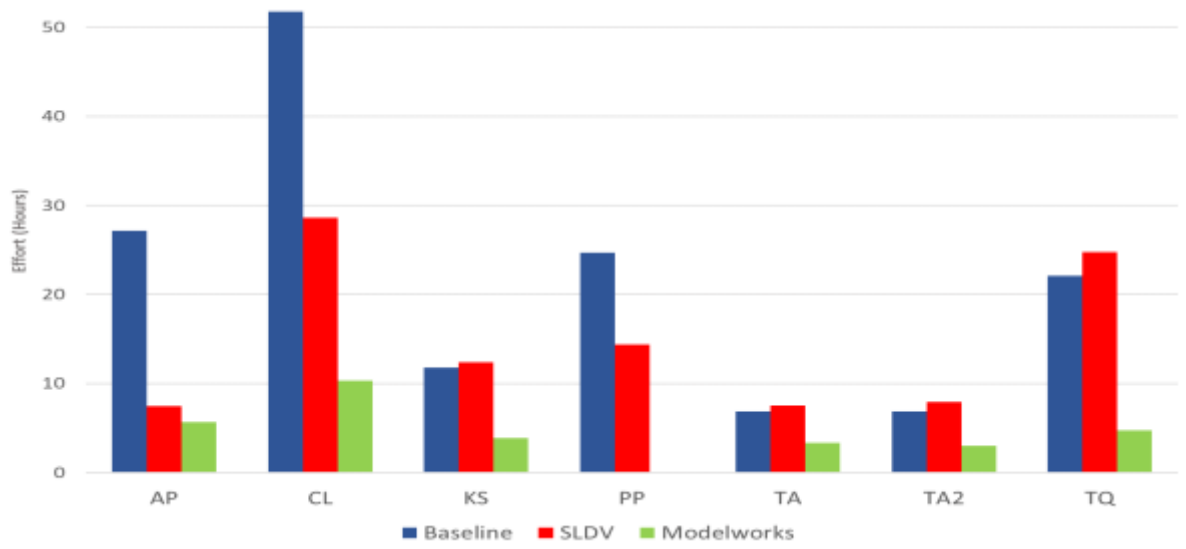

Figure 3 - ISO26262 Project Verification Metrics

This graph shows the following:

- In one instance (PP), no metrics were gathered for Modelworks as time to do the experiment had run out.
- The experiment that was repeated in a second company was TA2 which gave almost exactly the same results as the original experiment on TA. This shows that the process for measuring and the processes used for the verification were repeatable, which validates the whole experiment.
- There are instances where there is no benefit to using SLDV (KS, TA, TA2 and TQ).
- In all cases, Modelworks showed significant savings over the baseline (60-80%) and less but still significant savings (20-80%) over SLDV.

Since the experiment, Kapture has been produced which will automate away further aspects, and Modelworks has also been significantly improved. It therefore depends entirely on the existing processes as to what savings could be made in this part of the software life cycle. While consistent savings of were seen against the review process and the competitor tool, in half of the cases the competitor tool cost marginally more than undertaking a manual review. Furthermore, the formal review detected 49 errors; a mistake had been made prior to the blind error seeding. No claims for code compliance were made as this was out of scope for the project.

Comparable Approaches

We are aware of similar approaches that have been undertaken with respect to the use of formal tools. For example, a University of Iowa led project called CoCoSpec. However, this is not necessarily scalable for industrial use as expertise in the specifics of the formal language and the use of formal methods tools would be needed and this is expertise typically not found in most parts of the aviation industry. In the UK, the ASTREA project sought to understand how to develop UAVs with a specific focus on certification. The project found out a lot of useful information on various topics but again was not able to focus on how to gain software certification of autonomous operational software at a price that was affordable . Our approach is to make the powerful formal tools available to the typical user that are cheap to buy and easy to use without extra training, whilst also supporting the certification requirements.

Another approach might be to use a simulation based approach and to use ED-218/ DO-331, the Model Based Design and Verification supplement to ED-12C/DO-178C. There has been considerable discussion in various markets about the usefulness of simulation in verification. The problem is always how does an applicant claim that the fidelity of the simulation is sufficient in order to make any claims and then how many cases need to be simulated in order to achieve sufficient coverage of the requirements. This all has a cost and may not be much different from more traditional approaches. Certainly simulation has a great role in helping to show what the behaviour could be, in other words, in understand and eliciting requirements. For such a wide possibility of cases as could be encountered in the air, use of an automated formal approach would appear to give the prospect of significant cost reductions while achieving the required assurance.

Further Developments

While the requirements and design development and their verification aspects have been examined under this project, the source and object code verification were not. Technologies exploiting automated formal methods based independent verification for both source and object code are in development; these too will be verification tools. It is intended that these will be deployed on UAV, other aerospace projects and in other sectors such as maritime, in the near term and the cost impacts will be measured. The approach to the development of the decision making software will shortly be used and adapted for a specific use case as well as adapted for use in other domains such as underwater for off-shore and nuclear decommissioning. We have also used the results of this project as the basis for assuring the behaviour of swarming UAVs. The further development of tools and the mathematics behind them has been exploited to show that we can assure the behaviour of an arbitrary size swarm of UAVs. While this of course relies upon good data from sensors, the decision making and the control software are all highly assured at the requirements and design level. This has shown that the approach is scalable to very large and complex problems.

Conclusions

The use of automated tools for requirements can bring significant benefits as it assists a developer get requirements more right earlier in the development process. The development of such complex behaviour has to start with a description of the required behaviour in an easily accessible form so that not only developers but any regulatory body can understand what is required. Indeed, the conversation that is typically held between a software developer and the systems engineer can be significantly enhanced through use such a technology as it makes it very clear to both parties the implications of requirements. It consequently enables a clearer demonstration of why the requirements have to be written in the manner that enables both parties to agree. This is an oft overlooked area of Section 2 in DO-178C and its importance for the impact on cost should not be under-estimated.

The use of formal methods also has to be made accessible to non-experts in order to enable their regular use. The claims for certification credit are therefore made against the embodiment of the formal approach in tools and not against diagrams written in Simulink/ Stateflow®. By using powerful, automated, formal methods tools, it is possible to make considerable direct savings. It is also possible to make indirect savings because errors are not passed onto later phases of the software development.