Requirements for Handling of Rare Conditions in Flight Control System Design

Herbert Hecht
SoHaR Incorporated
Culver City, California

1. Introduction

As part of research on design assurance for flight critical systems conducted for the FAA[1] it was found that aircraft incidents in which software played a part were very uncommon, and that those that did occur were caused by the software requirements rather than software implementation. The parts of the requirements that contributed to the incidents were invariably those that dealt with the handling of rare conditions. That exceptional conditions account for most failures has been recognized in other environments as well as shown by the following quotes.

> The main line software code usually does its job. Breakdowns typically occur when the software exception code does not properly handle abnormal input or environmental conditions – or when an interface does not respond in the anticipated or desired manner.[2]

> Therefore the identification and handling of the exceptional situations that might occur is often just as (un)reliable as human intuition. [3]

In practically all cases the handling of rare conditions is relegated to software that must first recognize the conditions that require exception handling and then invoke corrective or mitigating measures. To make the software designer aware of the need for these tasks they must be stated in the software requirements. Even if an exceptionally astute software designer includes these provisions in the absence of a specific requirement they may not get tested because the test specification is usually generated from the requirements. For these reasons all of the following material deals with assurance that software requirements for handling of rare conditions are completely specified prior to software development.

2. The Generation of Software Requirements

The conventional wisdom is that software requirements flow down from system requirements and that the resulting software requirements get parceled out into requirements for individual software modules as part of the software design process. This waterfall model is only partially correct and it is particularly deficient when is comes to the handling of rare conditions. The response to adverse events in the environment depends on details of the system and software architectures that are usually not known when system and software requirements were formulated. This is recognized in one of the SAE guidance documents for complex aircraft systems, SAE ARP 4754[4]. Section 5 of ARP 4754 deals intensively with requirements capture and the basis of requirements for the selection of development assurance levels. It also recognizes the importance of requirements that arise from design decisions:

> At each phase of the development activity, decisions are made as to how particular requirements or groups of requirements are to be met. The consequences of these design

choices become requirements for the next phase of the development. Since these requirements result from the design process itself, they may not be uniquely related to a higher level requirement and are referred to as derived requirements.

An example of a derived requirement arises from the selection of a specific radar altimeter that is to be used for the autoland function in a flight control system. This radar altimeter incorporates a number of self-checks that can be monitored by the flight control software prior to utilization of the radar altimeter for longitudinal control of the aircraft. The original system specification makes no mention of monitoring of these signals, and therefore a review based on the waterfall model would not detect that an available safety measure had been overlooked. The monitoring of the self-checks constitutes a derived requirement for the flight control system and its software. Failure to monitor the performance of a radar altimeter (or to replace it after previous instances of incorrect readings) contributed to loss of a Turkish Airlines B-737 near Amsterdam's Schiphol Airport in February 2009.[5]

The results of these added review requirements are diagrammed in Figure 1.

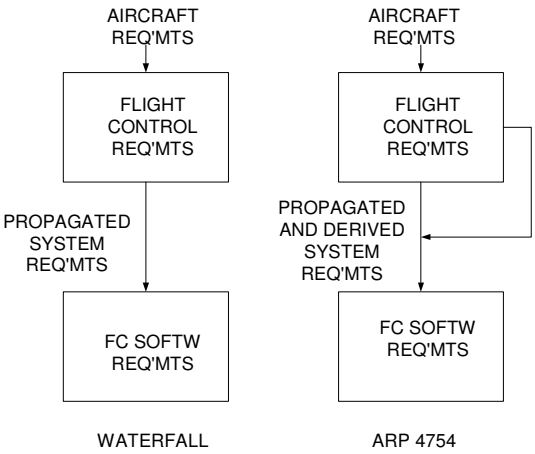

Figure 1 Requirements Generation

For all its deficiencies, the waterfall model enforces a discipline into the requirements review process and has inherent completion criteria that are difficult to extend to derived requirements. The next heading explores how to ensure capture of all flight control software requirements.

3. Assuring Completeness of Requirements

The approach for derived requirements that is described in the following is essentially one of divide and conquer. The major sources of derived requirements for a flight control system are:

- Aircraft environment (electric power, thermal control, communications)
- Aircraft operations (landing gear & flap positions, subsystem self-checks and outages)
- Computer environment (hardware failures, memory errors, executive and middle ware problems)
- Self-checks, calibration and monitoring of flight control components (waiting for completion, who checks the checker?)
- Flight control system software design and test (including response to all the exception handling requirements arising from the previous bullets)

These sources are arranged here in the order in which their implementation is completed in a typical project. Scheduling the review of requirements in this order permits reviewers to concentrate on one area at a time and avoids the schedule crush when all requirements have to be reviewed at the same time. The staggering of the review activities also facilitates participation of specialists in the functions that are to be reviewed at a given time. A review for completeness of requirements in the aircraft environment category might consider the following questions for electric power. Standards cover many of these conditions but a thorough review will want to verify that the electric power specialist and the flight control system engineer interpret the standards in the same way.

1. When the electric power system reconfigures in response to a failure, are there transients that have to be tolerated by the flight control system. What is the magnitude and duration of these transients?
2. Does the load shedding strategy for electric power outages require reduced capability modes for the flight control system, e. g. reversion to pure attitude control or rate stabilized fly-by-wire?
3. Does the flight control system need local energy storage for orderly power-down under some conditions?

Under thermal control the review will want to consider how the response to failure of the primary heating/cooling system will affect the flight control system and whether there is timely warning of thermal control malfunctions that may lead to abnormal operation of the flight controls. In the communications area a typical review topic is whether data flow from back-up instruments is fully independent of that from the primary instruments.

These questions can be organized into checklists that are partly generic and partly specific to the aircraft environment and flight control system being developed. Similar checklists can also be constructed for the other bulleted functions shown above.

4. Requirements Formats

As described above, derived requirements for handling of rare conditions will arise over most of the system development cycle, and the current heading addresses the format in which the requirements can be expected to be expressed. Evolutionary steps in the generation of a given requirement usually include

Objectives – these describe the conditions to be prevented or to be achieved; they may reference regulatory or system level requirements. The operating modes to which the objectives apply also need to be stated. The objectives are typically formatted as a natural language document, frequently in a hierarchical form in which a top level requirement is numbered *X* and the next lower level *X.1* and below that *X.1.1* etc. The objectives are the platform from which algorithmic descriptions are generated.

Algorithmic descriptions – these identify how the anomalous condition is to be detected and the action(s) to be taken subsequent to detection, usually in a format such as

If $Y > C$
Substitute F2 for F1
Monitor Y for t seconds
If Y(t) > C1
Substitute F4 for F3

In the above fragment, *Y* represents the output of a specific sensor, *C* and *C1* represent signal levels, and *F1..F4* are flight control modes or functions, e. g., filter functions or flight control channels. An

algorithmic description permits validation of the requirements by means of a simulation, an important step in assuring correctness as well as completeness. Following the generation and validation of algorithmic requirements, the assignment of the algorithm to a specific software function can be undertaken.

Assignment to a software function – this identifies the software module (and sometimes also hardware component) responsible for the execution of the algorithm. Requirements for sampling frequency, execution time for exception handling, and other implementation constraints (e. g., use of a specific filter algorithm) are usually included in this step. The document generated at the end of this step is an input to software design.

Reviews can be conducted on each of these formats. Review of the objectives documents will detect omissions or inconsistencies at an early stage and is therefore likely to avoid the cost of revisions and rework that will be incurred by detection at later stages. On the other hand, findings at this stage may not be considered conclusive because of the imprecision of natural language. Review of algorithmic description documents, particularly if they include the results of simulations, are much more conclusive but they take place later in the development and may require more expensive revisions than reviews conducted earlier. Review after assignment to a software function is the most conclusive, because it encompasses precision and timing issues that could not be fully evaluated in the previous formats.

A conclusion of the evaluation of the requirements formats is that reviews can and should be conducted in each format, with reviews in later steps concentrating on possible omissions and inconsistencies inherent in the earlier format. As an example, the objectives statement may refer to airspeed. By the algorithmic stage this will be refined to output of a specific sensor, transmitted over a given data bus, and possibly filtered over a stated interval. And review of the assigned requirements will be prioritized toward investigation of timing, precision and related problems.

5. Conclusions

In the introductory part of this paper it was seen that software causes very few flight control failures, and that those that do occur are predominantly due to flaws in the requirements, particularly requirements for handling rare conditions. The latter group typically does not flow down from aircraft requirements but is derived from design decisions. The immediately preceding two headings then showed that there is a normal progression of derived requirements, and particularly derived requirements for handling of rare events, among the areas with which the flight control system interacts, and also a normal progression of the formats in which the requirements are stated. It is now proposed that we recognize these progressions and use them to formulate a review schedule that encompasses a large part of the flight control system development cycle and thereby offers these advantages

a. Recognition that the waterfall model does not adequately represent the requirements formulation and flow-down in highly integrated and complex systems
b. Utilization of specialists in aircraft functional areas over a relatively short and predictable time for review of areas of their responsibility
c. Restricting reviews to specific issues for which data can be expected to be available at the time of the review
d. Avoidance of scheduling conflicts and resource limitations when all requirements are reviewed at the same time
e. Increased assurance that requirements for handling of rare conditions are complete and correct.

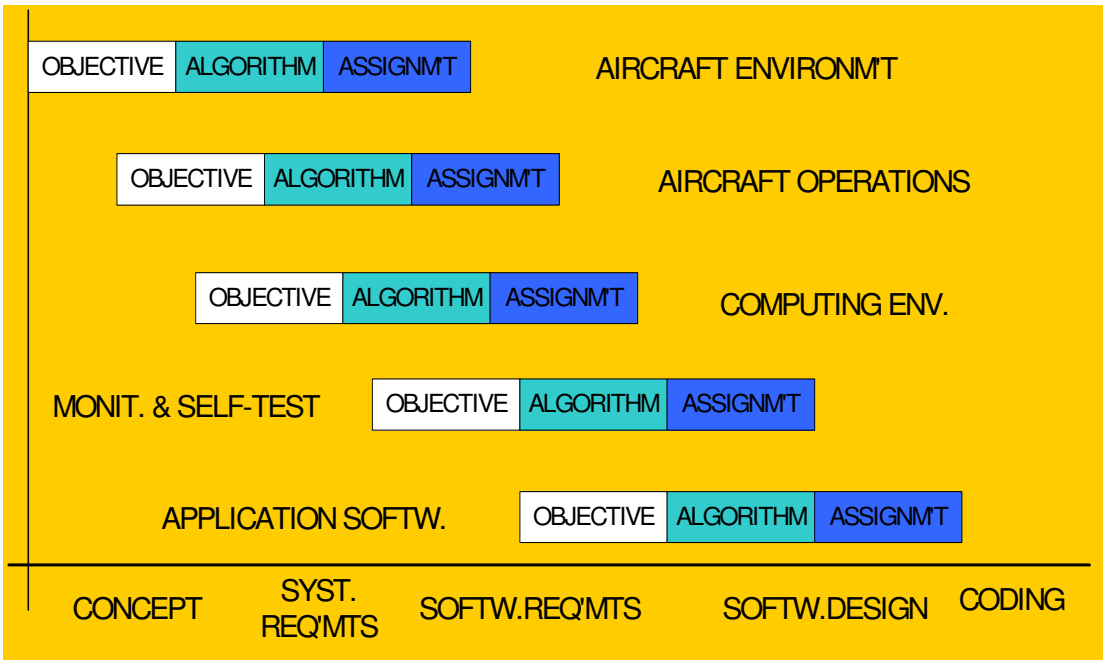

Figure 2. Example Review Schedule

In the above figure the horizontal axis represents the flight control development cycle. Obviously, a schedule of this type needs to be adapted to the specific circumstances of each project. There is increasing recognition that even legacy hardware and software requirements need to be reviewed, and particularly how these existing designs carry with them derived requirements for the developed part of the system.

Regardless of the specific schedule adopted, the emphasis on review of derived requirements for handling of rare conditions will be essential for removing the last remaining causes of failures in the flight control system that are associated with software.

References

[1] Flight Critical Systems Design Assurance, Contract DTFACT-08-C-00003

[2] C. K. Hansen, *The Status of Reliability Engineering Technology 2001*, Newsletter of the IEEE Reliability Society, January 2001

[3] Flaviu Cristian "Exception Handling and Tolerance of Software Faults" in *Software Fault Tolerance,* Michael R. Lyu, ed., Wiley, New York, 1995

[4] Society of Automotive Engineers, Certification Considerations for Highly Integrated of Complex Aircraft Systems, ARP 4754 dated April 10, 1996

[5] Onderzoeksraad voor veiligheid (Dutch Safety Board), "Preliminary Report, Accident [of] Boeing 737-800, TC-JGE, April 2009