Low-Cost, Embedded Safety-Critical Systems

Mats Heimdahl, Sanjai Rayadurgam[*], Peter Seiler, and Gary Balas[†]

1 Overview

Fault tolerance is vital to ensuring the integrity and availability of safety critical systems. Fault tolerance encompasses all of the architecture design and algorithms required to guarantee proper system operation during both normal and failure conditions. The architecture design includes the number and placement of sensors, actuators, and processors as well as the data buses required to interconnect all of the system components. A fault tolerant system must also include the logic and algorithms for fault detection, fault diagnosis, fault containment, and reconfiguration to continue operation in face of failures. In some systems, we can ensure integrity by simply reverting to a back-up mode in the event of a failure. For example, in an autoland system on an aircraft we can revert to a manual backup—the pilot. On the other hand, there are critical systems that are absolutely essential and must always be operational; for instance, the flight control electronics of an aircraft are essential to maintain controlled flight; here both the system integrity (the functionality is correct) and availability (the functionality is there when it is needed) must be maintained.

The aircraft industry has many years of experience designing systems driven by extremely stringent safety requirements. The system availability and integrity requirements for commercial flight control electronics are typically on the order of no more than 10^{-9} catastrophic failures per flight hour [1, 2]. They have converged to a design solution that is based almost exclusively on physical redundancy at all levels of the design. For example, the Boeing 777 has 14 spoilers, 2 outboard ailerons, 2 flaperons, 2 elevators, one rudder and leading/trailing edge flaps [4, 5]. Each of these surfaces is driven by two or more actuators (the rudder surface is driven by 3 actuators) which are connected to different hydraulic systems. Moreover, the control law software is implemented on three primary flight computing modules. Each computing module contains three dissimilar processors with control law software compiled using dissimilar compilers. The inertial and air data sensors have a similar level of redundancy.

The designs used in the aircraft industry achieve extraordinarily high levels of availability and integrity. The use of physical redundancy, however, dramatically increases system size, complexity, weight, and power consumption. Moreover, such systems are extremely expensive in terms of both the design and development, as well as the unit production costs. There is an increasing demand for high-integrity, but at the same time low cost, software enabled control in other transportation domains including aerospace (e.g. unihabited aerial vehicles and fly-by-wire in lower end general aviation aircraft), automotive (e.g. autonomous full-authority braking and adaptive cruise control). Moreover, the need for low-cost high-integrity solutions impacts non-transportation applications such as medical devices (e.g. inplantable infusion pumps, cardiac phasing, and neuro-stimulation). **A basic design challenge is to bring high levels of reliability and integrity to these other domains that can afford neither the cost nor the extra power, weight, and size associated with physical redundancy.**

The following research questions form important sub-problems that address this basic design challenge:

1. **Analytical Redundancy:** Can model-based and data-driven monitoring techniques be used to reduce the reliance on physical redundancy? Flight control systems for commercial aircraft use disimilar components, e.g. processors from different manufacturers. The use of disimilar components is meant to avoid a common design flaw in a single component. We hypothesize that algorithmic disimilarity provided by model-based and data-driven techniques may also provide benefit in detecting different classes of failures.

[*]M. Heimdahl and S. Rayadurgam are with the Department of Computer Science and Engineering, University of Minnesota, (heimdahl@cs.umn.edu and rsanjai@cs.umn.edu)

[†]P. Seiler and G. Balas are with the Department of Aerospace Engineering and Mechanics, University of Minnesota, (seiler@aem.umn.edu and balas@aem.umn.edu)

2. **Software Fault Detection:** Can advances in model-based software development can be applied for fault detection of software and hardware components of the system? We hypothesize that using the artifacts from a model-based development process as well as formalized software and safety requirements [3] as models for fault detection will be an effective approach.

3. **Validation and Verification:** How can the reliability of new architectures be certified? There are two important issues to this question. First, analytically redundant detection algorithms introduce new fault modes in the cyber-domain in addition to existing fault modes in the components (physical domain). For example, the algorithm could incorrectly identify faults. Thus new design tools are required to assess the reliability of systems that rely on analyitcal redundancy. Second, complex systems are by necessity hierarchically organized; they are decomposed into subsystems for intellectual control, for enabling compositional verification and validation, as well as for the ability to have the subsystems created by separate development organizations or external contractors for subsequent integration into the system as a whole. Such systems are typically assembled from components purchased from subcontractors. Certifying the entire system requires addressing the assurance of interconnected black-box components. We hypothesize that this can be done through (1) compositional reasoning by determining what a black-box component must (and must not) do to operate safely in its intended environment, (2) rigorous metric-driven testing that can be objectively evaluated to determine weather or not is is adequate to thoroughly exercise the required component behavior as well as the delivered component itself, and (3) symbolic execution of the delivered object code, that is, automatic discovery of behaviors of the object code and construction of tests to exercise these behaviors.

References Cited

[1] R.J. Bleeg. Commercial jet transport fly-by-wire architecture considerations. In *AIAA/IEEE Digital Avionics Systems Conference*, pages 399–406, 1988.

[2] R.P.G. Collinson. *Introduction to Avionics*. Kluwer Academic Publishers, 2003.

[3] Steven P. Miller, Alan C. Tribble, Michael W. Whalen, and Mats P. E. Heimdahl. Proving the shalls: Early validation of requirements through formal methods. *Int. J. Softw. Tools Technol. Transf.*, 8(4):303–319, 2006.

[4] Y.C. Yeh. Triple-triple redundant 777 primary flight computer. In *Proceedings of the 1996 IEEE Aerospace Applications Conference*, pages 293–307, 1996.

[5] Y.C. Yeh. Design considerations in Boeing 777 fly-by-wire computers. In *Third IEEE International High-Assurance Systems Engineering Symposium*, pages 64–72, 1998.