Survey of State-of-the-Art in Software Health Management and V&V of ISHM Software

Johann Schumann
RIACS/USRA

Ole J. Mengshoel
Carnegie Mellon Silicon Valley

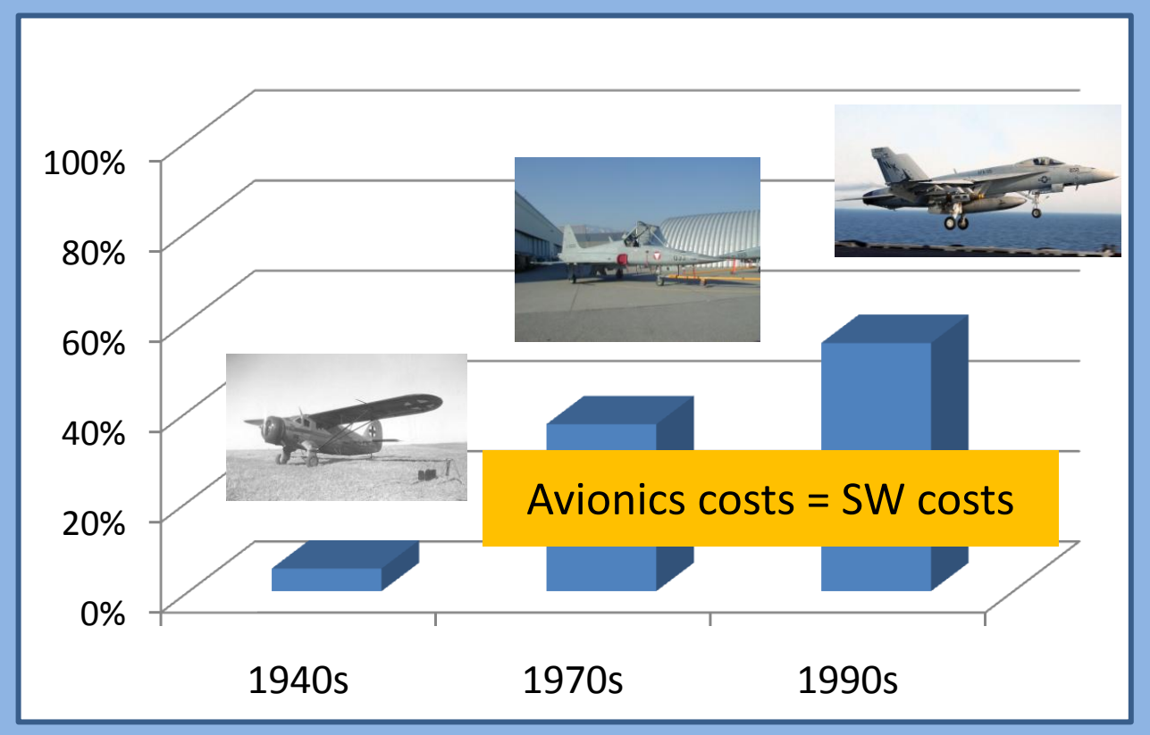
Adnan Darwiche and Knot Pipatsrisawat
University of California, Los Angeles

Abstract

Modern aircraft and spacecraft rely heavily on dependable operation of safety-critical software components, making the health management of software components (SWHM) extremely important. At the same time, assessing the health of a piece of software exhibits severe challenges, because software failure could be a result of faults (bugs) introduced at different stages of the software life-cycle. In this work, we survey and classify various techniques related to software health management and identify some technological gaps in this area of research. In addition, we discuss some practical techniques for V&V of integrated vehicle health management (IVHM) systems.

Survey of SWHM Techniques

Terminology



- Fault (bug): a cause of abnormal system behavior
- Error: a system state that deviates from what is intended
- Failure: an event in which the service becomes unavailable or deviates from the correct service

Classification Dimensions

Software Life-cycle: design, development, and post-deployment.
Purposes: the way the technique deals with faults.
Automation: the level at which the technique can be utilized without human intervention.
Resources: the amount of computational resources required.
Completeness: the extent of coverage the technique guarantees to provide.

Technological Gaps

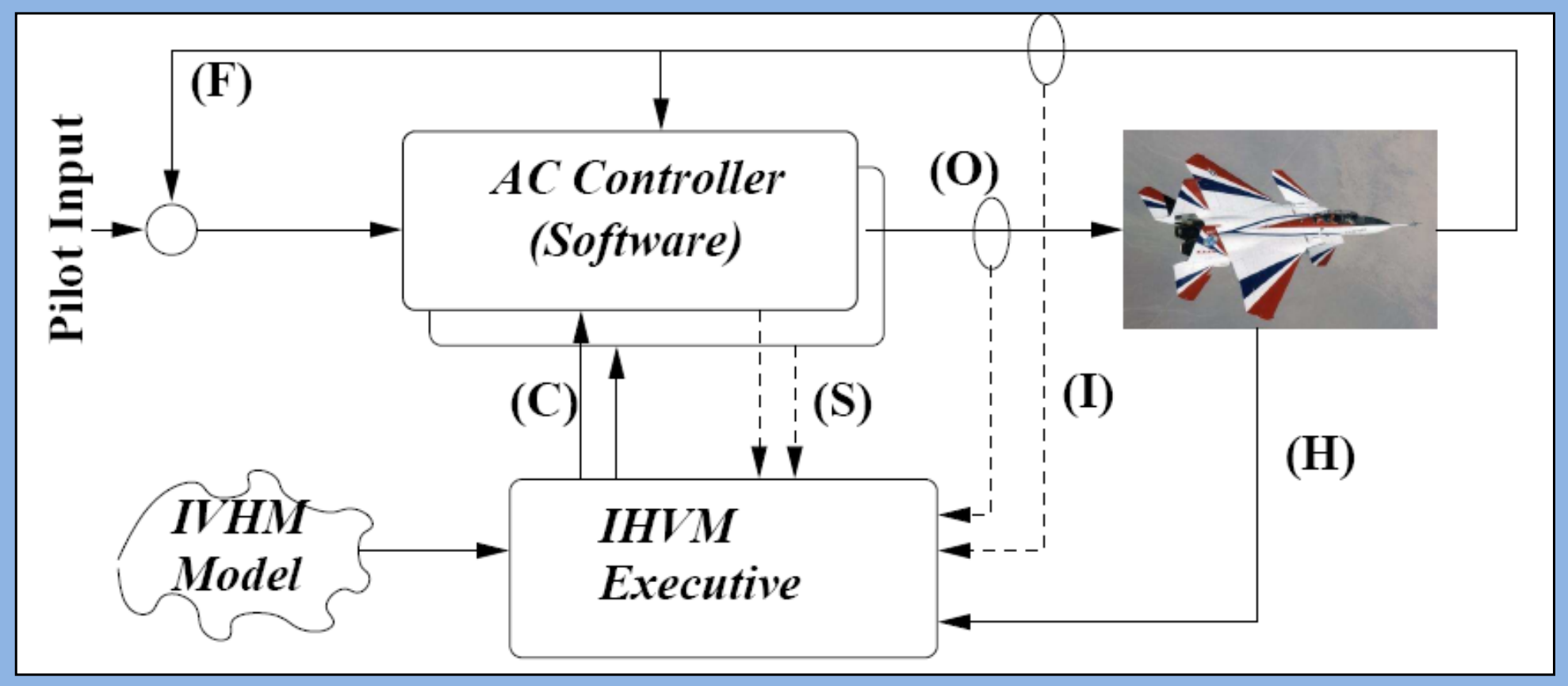
- Most approaches target discrete software rather than continuous calculations (as required for GN&C)
- There is no or little work in the area of prediction and prognostics for software
- Few approaches can be demonstrated to be correct and reliable, addressing the issue that the SWHM-software is a safety-critical piece of software itself
- Most techniques are for software and for software only, while many failures occur at the software-hardware interface
- There is not much integration between approaches from different phases of SW life-cycle.

Technique	Purpose	Automation	Resources	Completeness
Design and programming methodologies (design/development phase)				
Model-based design	Fault prevention	Manual	N/A	No
Goal-based operations	Fault prevention	Manual	N/A	No
Aspect-oriented programming	Fault prevention	Semi-automatic	N/A	No
Recovery-based computing	Fault prevention, fault tolerance (recovery)	Manual	N/A	No
Verification and Validation (V&V) (testing phase)				
Testing	Fault removal	Manual, semi-automatic	Adjustable	No
Simulation	Fault removal	Automatic	Moderation-high	No
Debugging	Fault removal	Semi-automatic	Varied	No
Numerical analysis	Fault removal	Manual	Low	No
Model checking	Fault removal	Automatic	High	In some cases
Theorem proving	Fault removal	Automatic	High	In some cases
Runtime techniques (post-deployment phase)				
Redundancy-based fault tolerance	Fault tolerance (compensation, isolation, reconfiguration)	Automatic	Varied	No
Checkpointing and rolling back	Fault tolerance (rollback)	Automatic	Varied	No
Runtime monitoring	Fault tolerance (concurrent detection)	Automatic	Minimal	No
Trace analysis	Fault tolerance (concurrent detection, preemptive detection, diagnosis)	Automatic	Varied	No
Built-in tests	Fault tolerance (concurrent detection, preemptive detection)	Automatic	Minimal	No
Software rejuvenation	Fault tolerance (rollback)	Automatic	Minimal	No
Computer immunology	Fault tolerance (concurrent detection, isolation)	Automatic	Minimal	No
Self-healing software	Fault tolerance (concurrent detection, compensation, rollback, reinitialization)	Automatic	varied	No

Table 1. Classifications of software health management techniques

V&V of IVHM Systems

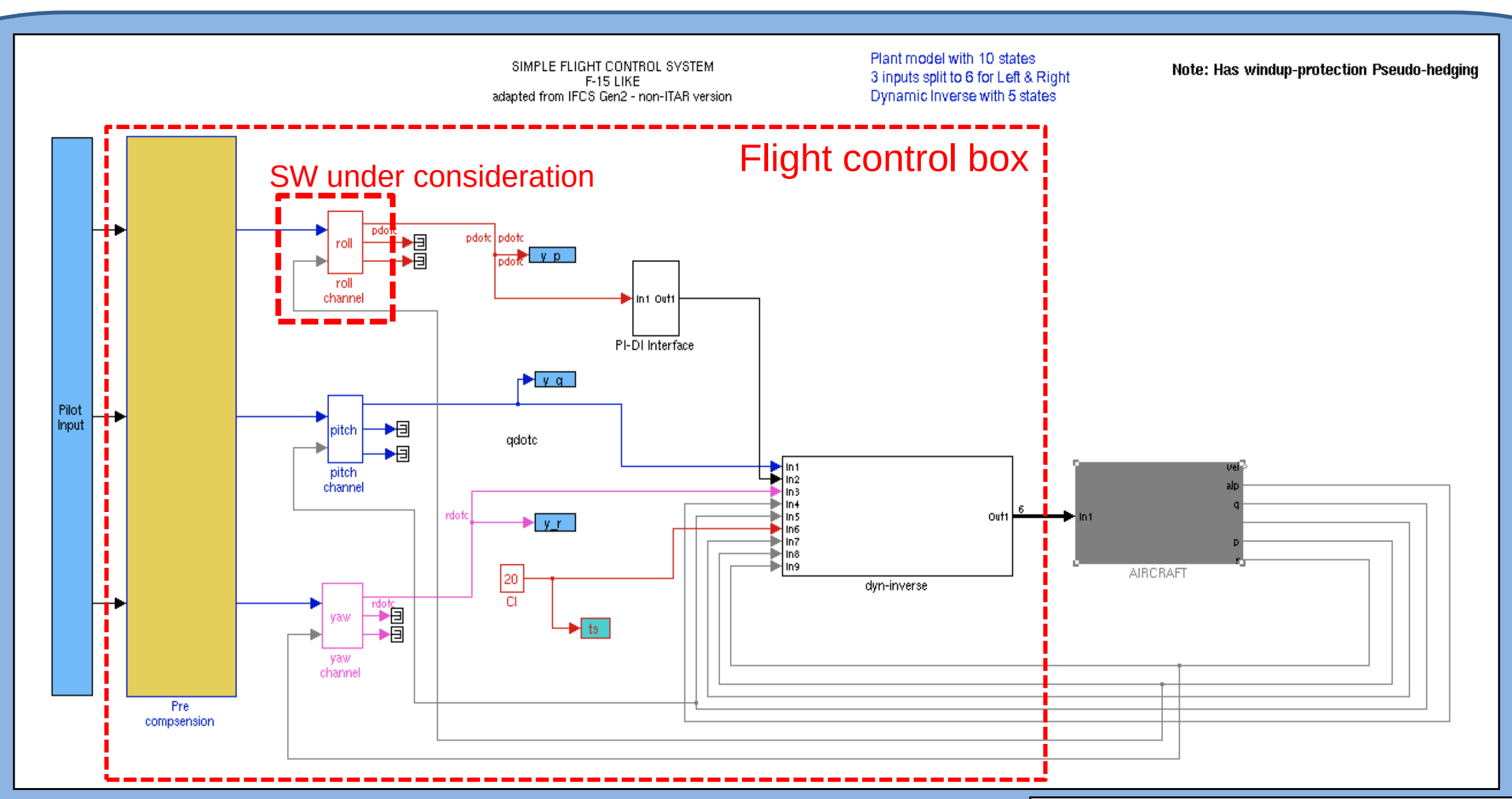
We divide the process of V&V of IVHM systems into two main parts:
Model-level: make sure the model (i) correctly reflects the system and (ii) that it represents the faults/errors/failures of interest.
Code-level: make sure the actual implementation/associated data structure of the IVHM system can execute without problems. This needs to be done both in the development phase and during the execution.

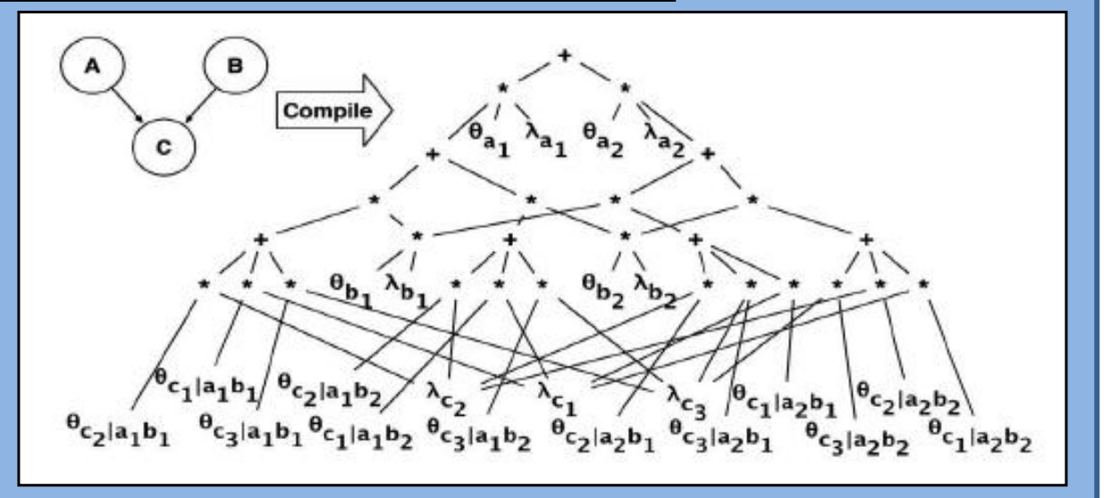


Analysis at Model Level

The model needs to be complete (covers all possible nominal and failure behaviors), and sound (i.e., failure indicated by the model corresponds to a failure in the actual system). Some desirable properties of models: determinism, fidelity, scalability.

A developed model can be tested with different test scenarios in various simulation environment. For example,
- IVHM test bed for RLV
- ERIS simulation environment for ARES-I FDDR





Model Compilation

Represent model as a Bayesian Network (BN), compile into an Arithmetic Circuit. Benefits:
- Compilation can exploit structure to provide a compact representation
- Demanding computation is delegated to an offline phase
- Fast and predictable response times are suitable for runtime monitoring

V&V of IVHM at Code Level

Must verify and validate both the (compiled) data structure and the code that will utilize it. Correctness as well as running time guarantee should be ensured for system deployed during runtime. Some approaches:
- Automatic test-case generation
- Static analysis
- WCET analysis
- Dynamic monitoring