ERF2010_004

CERTIFICATION CHALLENGES OF MIXED CRITICAL CYBER-PHYSICAL ROTORCRAFT SYSTEMS

Alex Boydston, MSEE, alex.boydston@us.army.mil; Dr. William Lewis, bill.lewis6@us.army.mil
United States Army Aviation Engineering Directorate

Abstract

Modern aircraft rely on complex and highly integrated hardware and software systems for safe operation and successful execution of missions. Flight safety requires these complex avionics systems be robust and reliable. Military avionics systems have evolved into highly integrated subsystems composed of computer hardware and software. As technology progresses, vehicles and their human pilots are relying on autonomous processing systems to a greater degree. Many critical and non-critical functions are becoming automated. Qualification and certification of such mixed criticality systems are problematic to schedule and budget constraints. These problems are well known in the industry and several groups have studied the issues in great detail. These complex systems are identified as "cyber physical systems" per the National Science Foundation and Defense Advanced Research Projects Agency.

Processes for good project management and engineering have traditionally been developed and promulgated through the defense acquisition guidelines and by various industry and government standards. These standards were thought to provide guidance that, if followed, improved the chances of developing safe, highly reliable and cost effective systems. The foundations of traditional program management emphasized well defined requirements, a meticulous approach to development processes, and reliance on extensive testing.

Industry recommendations and software incident investigation have led to the idea that complex system developers should have comprehensive requirements early in the lifecycle. Inadequate requirements, poor planning, and bad processes have been cited as causing poor design choices in the early stages that could only be addressed later by risk management. However, increasingly complex systems have outstripped traditional program management paradigms. Exhaustive testing of the most complex systems is not affordable. Project managers assert that these guidelines burden systems development to the point of making it very difficult to stay within budget and time schedule. They advocate a more streamlined approach. Complex systems certification becomes more difficult if programs adopt a more streamlined approach. Traditionally, under the Federal Aviation Administration (FAA) and other aviation authorities, qualification and certification of airborne systems relied on a rigorous test regimen.

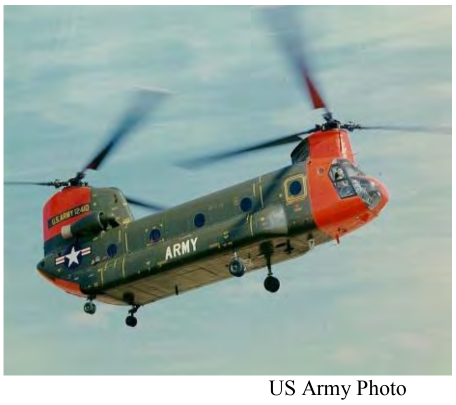
The question for certification authorities then becomes how can performance of the system under development be definitively demonstrated and established? There should be a concerted industry effort from all stakeholders to establish better design process guidance. These guidelines should quantify the reliability and safety characteristics of a design in its early phases. No matter what is adopted, demonstrated or derived performance is required for assessing airworthiness of these systems. The path and decisions that are made will determine the fate of future programs.

1. Introduction

Central to asserting the airworthiness of critical systems is proving they are safe to fly. Such systems are heavily supported by electronic, mechanical and human control systems. It is crucial that we adjust our processes and guidelines for qualifying and proving future systems are safe to fly within a reasonable budget and schedule else these systems will become too costly to design, develop, test, produce and maintain. This paper will discuss: (1) progression of complexity in US Army rotorcraft, (2) definition of mixed critical complex systems, (3) complexity challenges, (4) development and qualification problems, (6) notional solutions, and (7) current work in progress addressing these challenges.

2. Progression of Complexity in US Army Rotorcraft

Since 1941, with the first United States (US) Army helicopter (i.e., the Platt-LePage XR-1), rotorcraft technology has grown in complexity, both mechanically and electronically and, more recently, via software. Since 1950 some form of electronically controlled subsystem has been attempted in rotorcraft. In 1950, a Piasecki HUP-1 was flown with a modified Sperry A-12 analog computer controlled autopilot [11]. In 1960 the first hover coupler was developed to allow hands off hovering with a control feedback of Doppler radar and radar altimeter. It was demonstrated on an S-58, equivalent to the H-34 Choctaw [11]. From 1971 to 1974 the US Army Air Mobility Research and Development Laboratory and the Canadian Department of Industry, Trade and Commerce developed and demonstrated a digital-fly-by-wire (DFBW) flight control system called the Tactical Aircraft Guidance System (TAGS) in a modified CH-47B Chinook (see Figure 1). This system used a combination of redundant computing systems and incorporated advanced control laws. The technology that was developed and tested on TAGS was eventually used on the RAH-66 Comanche (see Figure 2) including computer synchronization, built-in-test, and majority voting and mid-value select. It proved that complex software and hardware DFBW for rotorcraft was achievable [11].



US Army Photo

Figure 1 - US Army CH-47B Chinook

US Army Photo

Figure 2 - US Army RAH-66 Comanche

The most complex rotorcraft to this day was developed as a joint aircraft, on a program initially led by the US Army in the 1980s, called the V-22 Osprey (see Figure 3). The Osprey has the ability with its tilt rotor system to fly as a helicopter or a plane. It's supported by triple modular redundant flight computer architecture with DFBW and glass cockpit.

US Navy Photo

Figure 3 – V22 Osprey

The UH-60M Blackhawk upgrade (see Figure 4) is the US Army's latest DFBW project. Like the Comanche and Osprey, the UH-60M upgrade employs a triple modular redundant computer system that uses a combination of majority voting and mid-value select. The control laws are implemented in software. More than 60 functions of varying levels of criticality interact with each other in the UH-60M aircraft with a large part implemented in software [12]. Thus, it can be considered a cyber-physical system.

US Army Photo

Figure 4 - US Army UH-60M Upgrade DFBW

More examples of present day, complex cyber-physical systems include the upgraded CH-47F Digital Automated Flight Control System (DAFCS) which replaced an older analog AFCS, the Full Authority Digital Electronic Control (FADEC) system on several rotorcraft, Stability Control Augmentation Systems (SCAS), Terrain Avoidance Warning System (TAWS) on some military aircraft and the glass cockpit systems such as the Common Avionics Architecture System (CAAS) on CH-47 and CH-53.

Additionally, stringent requirements are being levied on civil and military aircraft for Global Air Traffic Management (GATM), Required Navigation Performance (RNP) for area navigation (RNAV), and Traffic Collision Avoidance Systems (TCAS). The Primary thrust of these improvements is to place more reliance on aircraft autonomy and decrease the reliance on ground-based traffic management aids.

In an April 2010 Huntsville Chapter of the American Helicopter Society (AHS) briefing by Dr. James Snider, US Army Director for Aviation Development [13], areas were cited for future (circa 2025 and beyond) development for military rotorcraft which reemphasizes the growth in cyber-physical subsystems for rotorcraft. The new development includes focus on survivability, situational awareness, affordability, performance, communications and lethality. Survivability will include focus on signature reduction, aircraft hardening, redundancy, speed, and active protection. Situational awareness will involve virtual cockpits, more Unmanned Aerial Vehicle (UAV) teaming, UAV swarms, degraded visual environment control, sensor fusion, and foliage penetrating sensors. Affordable technologies will be promoted by adopting condition based maintenance, open-source software, code reuse and commonality. Enhanced performance is to be achieved via hybrid engines, active rotor control, removal of swash plates, variable geometry rotors, and sea based improvements. Communication networking will include global information grid compatibility, multilevel security, software driven waveforms and integrated assured communications. Last, research will be conducted for scalable and directed energy weapons, and artificially intelligent weapons along with target recognition and selectable yield warheads.

Obviously, the level of system complexity is increasing for US Army rotorcraft. All of these proposed advances in functionality will increase the human, hardware, and software interaction and will place a heavy burden on testing, qualification and certification unless something is done to mitigate and improve the current approach to system lifecycle processes.

3. Mixed Critical Complex Systems Defined

The word "complex" is derived from the Latin word *complecti* which means to interweave or entwine. A complex system has interdependencies on other parts of that system which can exhibit linear or non-linear behavior. "Complexity deals with interrelationships among parts or elements. The more dependent variables [and states] a system has the greater that system's complexity is" [2].

The International Council on Systems Engineering (InCOSE) defines a system as "a construct of collection of different elements that together produce results not obtainable by the elements alone" [3]. The European Complex Systems Society defines a complex system as "a system where the collective behavior of its parts entails emergence of properties than can hardly, if not at all, be inferred from properties of the parts" [4]. As the coupling of parts become tighter and more dependent within a system, the level of complexity rises. After components are integrated into a system unexpected behavior of systems often emerges which is not obvious at the component level. Thus, focusing on just the components of a system is not adequate when analyzing systems and a holistic systems engineering approach is required.

In military and aerospace systems the concept of mixed criticality exists in terms of safety, mission and security. A system where a failure or defect could cause risk to human life is called a "safety critical system". A system where the loss of capability causes reduced mission effectiveness is called a “mission critical system”. Military aircraft usually handles varying levels of security including clear, confidential, secret and top secret. Thus, these systems are “mixed security systems” [5].

Systems which feature a tight combination of, and coordination between, the system’s computational and physical elements are known as “cyber-physical systems”. DARPA defines cyber-physical systems as “systems that derive significant portions of their functionality from both software and electromechanical subsystems” [33]. The growth in Integrated Modular Avionics (IMA), software and the mixture and management of these varying critical items are driving complexity in design, development, testing, manufacturing and maintainability.

4. The Complexity Challenge

Aircraft control systems traditionally were comprised of analog pressure gauges and the transfer of control was via mechanical means. Since the advent of DFBW technology, aircraft have relied more on integrated circuit electronics and software to maintain control. Aerospace systems have required high reliability numbers to increase confidence in those systems. Tighter tolerances on traces have impacted aerospace electronics reliability. Integrated circuits have increased in complexity per Dr. Gordon Moore’s law which stated that transistor quantity per chip will double every two years [6]. In recent years that rate has leveled off. However, processor cores with multiple on-chip memory and internal buses have increased driving up software system complexity significantly.

The calculations to achieve the reliability Mean Time To Failure (MTTF) and Mean Time Between Failure (MTBF) has been based on known or observed physical failure modes or extrapolated and estimated from test data. Some argue that systems reliability gives confidence in system safety. However, a reliable system is not necessarily a safe system (e.g., a mower or nuclear power plant may be reliable but unsafe if not properly controlled or handled).

Hardware and software reliability are two distinctly different concepts. Software does not break like hardware over time. Hardware reliability tends to have a bathtub shaped curve as shown in Figure 5 [7]. Software is part of the systems equation, yet the methods to calculate software reliability have not been well established and agreed upon. Software reliability does not adhere to the bathtub failure rate curve like hardware, but empirically exhibits a curve more like that shown in Figure 6. Software errors are caused by human design and coding errors. Software does not wear out like hardware over time, but can cause problems by its operational context (i.e., heavy processor loading, collisions, bus errors, race conditions, etc.) As software is upgraded failure rates tend to increase but may decline with coding corrections [7].

Avionics systems are becoming more software intensive. Interlinking avionics systems may exacerbate software dependencies. Since software is largely based on abstractions of human thought and processes, its complex by its nature. Software does not break or decay like hardware. Thus, we have relied on process control through development, testing and reporting to ensure some confidence in the produced software.

Sadly, even with the current guidelines and standards, we often find errors in the software late in the system design costing programs lost time and money. Therefore, it is important to find and resolve problems early. Figure 7 shows where faults are introduced, found and the cost of fault removal [8, 9, and 10]. As can be seen, 70% of the faults are introduced early in the establishment of the requirements and only 3.5% of those faults are found in the same stage. If the fault is found in the requirements/design phase and removed, a minimal 1x (i.e., 1 times) cost impact is incurred. The cost continues to rise to greater than 110x the original cost if errors are not found until acceptance testing just before fielding.

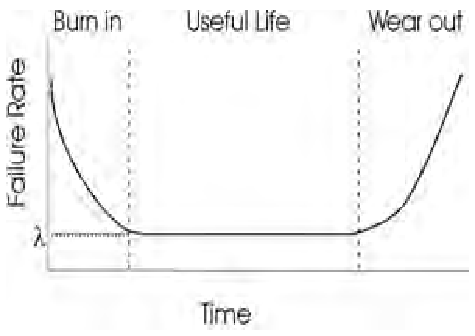


Figure 5 - Bathtub Curve for Hardware Reliability [7]

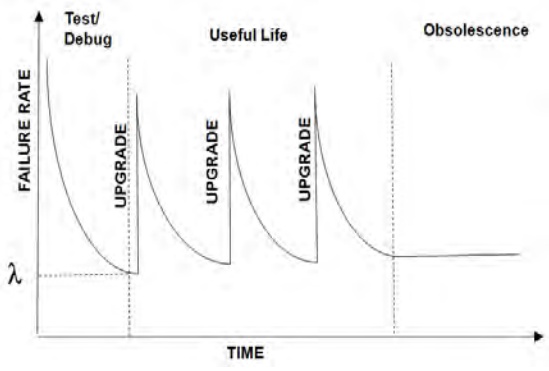


Figure 6 - Notional Curve for Software Reliability [7]

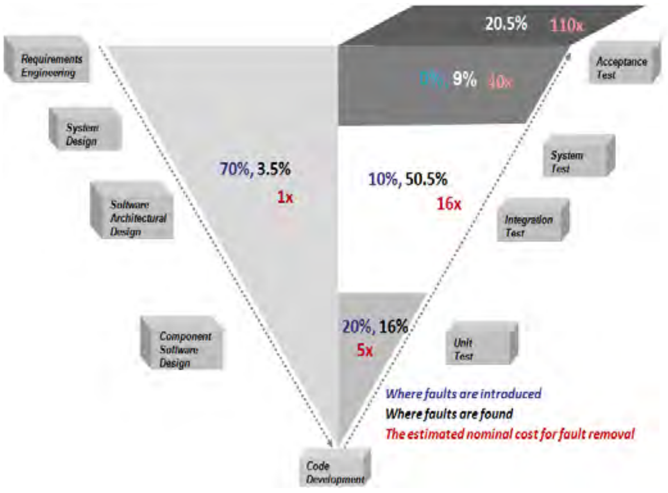


Figure 7 - Cost of Faults in the V-Curve [8, 9, 10]

Traditional methods of system design need to be restructured to address complex system development. Otherwise, systems of tomorrow will be too costly to develop. It is crucial that appropriate methods, processes and tools are utilized early in the requirements and design phase to mitigate errors early in the process and save costs and time in the overall system lifecycle.

5. Development and Qualification Challenges

Since the US Army is a large user of rotorcraft it is faced with continuing challenges of growing complex system development and qualification. The US Army Aviation Engineering Directorate is responsible for ensuring airworthiness qualification for all US Army aircraft that is managed by the Aviation Program Element Office. Airworthiness is composed of design approval, production approval and continued airworthiness. To be airworthy means that the system is safe and reliable to operate and will perform the mission when delivered. Also, it means that the aircraft will continue to safely perform the mission if maintained and operated per its specifications. Any repairs or overhaul must maintain airworthiness. Flight control systems for the US Army must meet system reliability of $1x10^{-6}$ for tactical airspace and $1x10^{-9}$ for civil airspace. There is some variance in these reliability levels depending on if the aircraft is rated for Visual Flight Rules (VFR)/Visual Meteorological Conditions (VMC) or Instrument Flight Rules (IFR)/Instrument Meteorological Conditions (IMC). Determining the levels of reliability is difficult.

Current Qualification Process Needs Upgrading

Current qualification processes exist to address certifying avionics systems; however, these processes lag technology, making certification difficult. For years, a solid systems engineering foundation has existed for the US Army within the Aeronautical Design Standard Rotorcraft and Aircraft Qualification Handbook (ADS-51-HDBK). It encourages good requirements definition, forms the basis of good solid detailed system specification, outlines the establishment of key critical system and functional requirements, and prescribes the necessary design reviews, analyses, verification and validation within an Airworthiness Qualification Plan which is responded to by an Airworthiness Qualification Specification, providing guidance to an Airworthiness Release. This process is proven, but, as with all guidance handbooks and standards, can be upgraded to better handle cyber physical systems.

While formalized systems engineering has been around since the 1940s we are still using document-centric approaches to capture and trace requirements to test. Different modeling and analysis approaches are used across different organizations with no ability to set a single true model for a project and no way to communicate the modeled system to different engineering and management groups. This leads to disjoint or errant analyses, missed requirements, bad integration and test coverage. Program management and contractors conduct verification and validation per test plans and procedures to test components, electromagnetic interference, vulnerability, and compatibility (EMI, EMC, and EMV), environmental effects, and electrical power. These tests are conducted during bench testing, within System Integration Laboratories (SILs), and during ground and flight aircraft tests. From the reports and witnessing of this data the US Army Aviation Engineering Directorate (AED) determines the airworthiness substantiation by using a mix of qualitative and quantitative analysis coupled with best, but sometimes subjective, engineering judgment. While processes exist to address qualification, the processes are still geared to older federated systems with the goal to demonstrate meeting performance and safety criteria. A problem with this approach is that issues found at this stage are often too late to fix economically, thus allowed to fly with risk with warnings, notes and cautions. Qualification processes for complex systems are expensive and there is never enough time or money to exhaustively test every requirement. Thus, systems are apt to be released with unknown bugs still existing and unspecified in the airworthiness releases and impact statements.

Safety Requirements Determination is Late in Process

There is a need to determine completeness and validity of not only performance requirements, but also safety requirements early in a program. As Dr. Nancy Leveson of Massachusetts Institution of Technology (MIT) stresses, the original intent and rationale of requirements and their impact to safety in a system should be captured [1]. Early safety assessment is lacking to set assurance levels in military aircraft. Safety design and hazard analysis should be done early. Military aviation industry can draw from some civil practices such as from the Society of Automotive Engineering (SAE) Aerospace Recommended Practices (ARPs) 4754 and 4761 or from Systems Theoretic Accident Model and Processes (STAMP) Based Process Analysis (STPA) method [26]. This may seem straightforward to remedy; however, programs struggle with shrinking development budgets and shortening schedules causing them to detour from the recommended processes. While following industry standards and guidelines such as DO-178B, SAE ARP 4754 and 4761 encourages processes, they do not prevent poor system design decisions. Conversely, STPA does get involved in the requirements process but has only been used in a few programs showing promising results.

Need to Understand and Measure Complexity

Additionally, not knowing how to parametrically describe or assess system complexity makes it difficult to tailor the approach to such systems. There is no guidance on acceptable means to limit complexity to qualification requirements.

Legacy Systems Pose Challenges

The current US Army rotorcraft fleet is composed of airframe designs dating back to the 1950-80s. Introduction of new technology in this older aircraft frequently results in mismatches in architecture, software and integration. These aircraft are sometimes upgraded in piecemeal fashion making qualification difficult. While the US Army upgrades the fleets the technology evolves quickly before fielding leading to early obsolescence.

Military Aviation Must Comply with an Equivalent Level of Safety to Civil Airspace Entry Requirements

Military aircraft deals with more stringent environments and missions than civil aircraft and is increasing in complexity to handle those environments and missions. Military aircraft are required to qualify to an equivalent level of safety and show interoperability with civil infrastructure for operations in national and international civil airspace. There are recent requirements for GATM, which is a system that relies on the Global Positioning System (GPS) to create more accurate navigation. The military needs differ from civil needs, typically requiring superior performance and different qualification methodology. The burden is to show that military capabilities map into safe airspace entry. Military aircraft must take into account higher pilot workloads due to tactical conditions during combat and other military functions and be able to fly in harsh environments night or day. To cope with these challenges military aircraft are continually being upgraded to include complex fault tolerant DFBW systems, glass cockpits, air survivability equipment, multiple levels of security, manned-unmanned teaming and other upgrades that were previously mentioned.

Safety Critical Software Development Process Challenges

Current and future complex cyber-physical systems with mixed critical architecture requirements are on the drawing boards. The processes for developing the electrical and mechanical systems are long established through Computer Aided Design (CAD), Computer Aided Modeling (CAM) and description languages such as Very High Speed Hardware Description Language (VHDL). Software design and development processes have evolved from traditional structural language and flowcharting to more formal methods using methods such as Object Oriented Design (OOD) approaches, Unified Modeling Language (UML) and reliance on libraries such as the C++ Standard Library. However, these newer methods have not fully been established by the safety critical community. The commercial industry, in less safety critical product development, has embraced the use of OOD methods and included multithreading within their development regime. With military systems adopting more commercial off the shelf (COTS) items more products that have used these methods exists. Likewise, the workforce that is being hired out of college is trained with OOD methods and mostly in C/C++ and Java. Also, safety has not been included in general engineering curricula. This adds to the challenges for the development of safety critical systems. Development of deterministic code is a must with safety critical systems. Rigorous testing is attempted to flesh out all issues but often lacks complete results some of which may be safety critical. Waiting until the test phase to determine safety issues with complex systems can spell programmatic disaster.

Absence of Modeling and Formal Methods

Our approach to systems engineering has not changed drastically over the span of 50 years. We still generate our requirements in document form and quite often struggle with tracing the requirements down to the test level and back up to the original requirements. This is very difficult when multiple groups are working on a complex project. It is possible for requirements to change and not get updated properly or not be fully understood as their impact on other parts of a system until it's too late in the development. Some programs try hard to utilize modeling and simulation; however, the tools used by various engineering teams involved on a project may differ and do not easily translate to other models. Additionally, the ability to perform early analysis of a design for critical items such as worst case execution time, latency and data bus loading is needed which impact safety or performance. Waiting until integration to discover issues consumes significant program resources (see multipliers in figure 7). 80% of the problems are being discovered in the integration phase of a program or later. Not being able to discover and resolve these problems early in a program can be detrimental. It is paramount that we verify and validate requirements as early as possible. Thus, formal methods, modeling and simulation are needed to flesh out the bugs at the beginning of a program and carry a continuous verification and validation process.

Lack of Certified Test Labs

Testing and qualifying these types of systems is critical. SILs equipped with real systems and simulated flight exercises the system before release to ground and flight testing are in the critical path to achieving qualification. Sometimes, programs employ laboratories that have not been verified by the developer and validated by an independent reviewer to match the system that is being built which results in low confidence results.

Need to Better Understand Human-Machine Interaction

The majority of aircraft accidents are attributed to human error. Human factors engineers rightfully argue that the load of information that the pilots must deal with necessitates quick access to the appropriate information to reduce accident rates. According to [35], "80% of the [US Army's] accidents are caused by human error". Modeling and safety of the human cognitive aspects of cyber-physical systems needs focused research. More can be found on this subject in [36, 37, 38].

Declining Expertise

Other challenges in development and qualification exist. With declining engineering school enrollments, as with other technical fields, the gap for engineering expertise on software intensive systems is growing. There is a lack of trained personnel in safety, qualification and test for complex cyber-physical systems.

6. Notional Solutions

In examining the challenges in the aerospace industry and current practices to face the challenges, it is evident that a shift in the design, qualification and certification approach is needed for the complex cyber-physical systems. The following are some suggestions to address the aforementioned challenges.

Assign a Chief Software Engineer to Programs

US Department of Defense (DOD) acquisition guidelines mandate that a Chief Engineer resides within each Program Management Office (PMO). With the increasing prevalence of software-reliant systems, the complexity of embedded software systems and continued hardware needs, either a Chief Software Engineer should be appointed to the PMO or it be required that the assigned Chief Engineer also possess software, as well as hardware and systems, expertise. This expertise will ensure that some focus is placed on embedded software systems in programs and that this critical area is not just treated as a peripheral item with no importance.

Promote Simplicity and Establish a Complexity Metric

While feature rich designs grow in complexity we should still promote simplicity in design as much as possible. By determining such a metric we can set the design, qualification and certification path accordingly. Dr. Eberhardt Rechtin quoted in [2], "Purely analytical techniques, powerful for the lower levels, can be overwhelmed at higher ones. At higher levels, architecting methods, experience-based heuristics, abstraction, and integrated modeling must be called into play. The basic idea behind all of these techniques is to simplify problem solving by concentrating on it is essential. Consolidate and simplify the objectives. Stay within guidelines. Put to one side minor issues likely to be resolved by the resolution of major ones. Discard the nonessentials. Model (abstract) the system at as high a level as possible, then progressively reduce the level of abstraction. In short, Simplify!"

Earlier identification of problems in requirements and an understanding of the level of complexity are needed. In order to objectively gauge simplicity there has to be a measure of complexity on a system. If some way that the complexity level of new or modified systems could be determined early in a program then the expectations for development, testing and production may be better understood and complex programs could be managed accordingly.

Complexity metrics are not new. One such metric is the cyclomatic complexity developed by Thomas J. McCabe, Sr. in 1976 for the complexity of a software program exists [35]. It measures the quantity of independent paths through a program's source code and is determined by using a control flow graph. McCabe encouraged programmers to limit the complexity and increase simplicity as much as possible. This thinking should be extended from software to systems. The number and type of interfaces, dependencies, and human-machine interaction complexity could factor in. While the idea of a metric sounds good, we should also analyze if true benefit would be achieved by reviewing the return on investment of metrics such as the Capability Maturity Model Index (CMMI).

Emphasize Coding Pattern Guidelines

Safety critical and non-critical programming guidelines should be addressed. The programming languages for safety critical applications should be chosen carefully or at least bounded on the features used. For instance, C and C++ have become popular languages, over the previously DOD mandated Ada. The use of pointers, polymorphism and inheritance patterns should be minimized or appropriately applied. Experience shows that if not properly applied, these patterns can lead to errant behavior.

Another issue that should be handled with safety critical code is the use of standard libraries that are used with no understanding of what "dead" or "deactivated" code results. In the operation of a program, if an errant pointer points to dead code faults could occur. Lastly, in cases where multithreaded code controls concurrent process and threads using semaphores and mutexes, it could become impossible to determine the behavior for all possible states if not properly implemented. Further guidance on programming and programming language patterns exists in [14].

Encourage Code Reuse and Certified Composable Software

In order to promote real code reuse, stable, certified standard code libraries for real-time languages and operating systems should be established across the industry. Reusable code should be analyzed and understood via modeling and analysis for specific computing environments before implementing. Also, auto code generation of certifiable code along with certified static code analysis tools coupled with functional tests should be standardized. Obviously, time and space partitioning schemes (e.g., ARINC 653) for operating systems, middleware, and virtual machines to allow mixed functionality at different criticality should coexist on an integrated or distributed system [15]. With these concepts in mind for reuse, certified libraries, code generation, analysis tools, and partitioning, the idea of a composable certification as referenced by Dr. John Rushby [16] and the Air Force Research Laboratory's Mixed Critical Architecture Requirements (MCAR) effort may be realized (see Figure 8), but keeping in mind that testing of emergent behavior as a system is still mandatory.

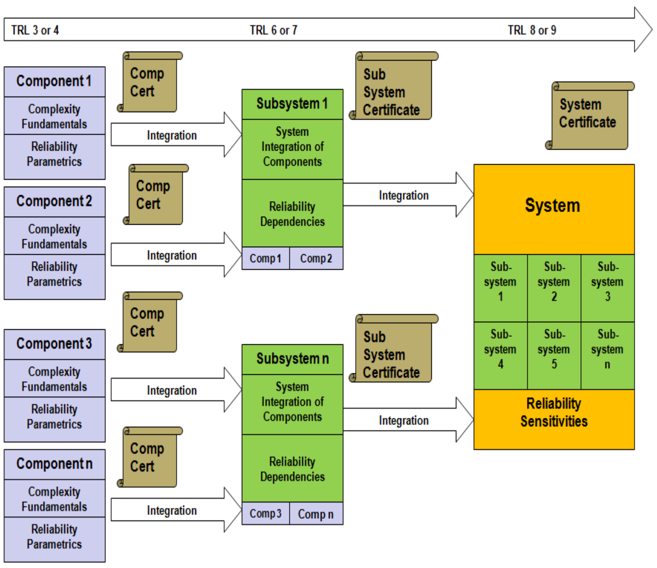


Figure 8 - Composable Certification

Provide Guidelines for a Federated Component Mix

While systems are becoming more integrated, some argue that the mixing of non-critical and safety-critical functionality should never happen. There are standards such as ARINC 653 that provides for partitioning of operating systems (OS) and mechanisms to partition middleware (i.e., above the OS and below the application software). Furthermore, for distributed networked avionics systems, deterministic time-triggered protocols should separate data flows into different levels of priority or quality of service. Most data bus architectures are mixed within Army rotorcraft platforms to include MIL-STD-1553, ARINC 429, RS-422, discrete signals and standard non-time-triggered Ethernet. Bus architectures that control time sensitive critical and non-critical data flow should be considered in aviation safety critical systems to attain proper system partitioning and promote determinism.

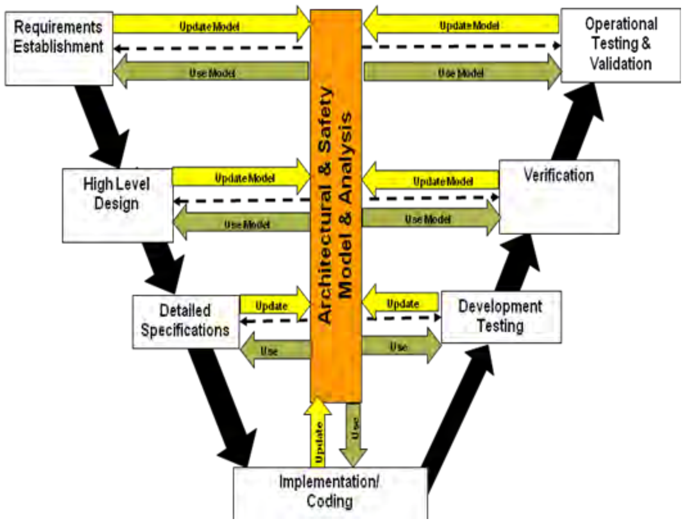


Figure 9 - Lifecycle Use of Architectural Model

Perform Early and Lifecycle Modeling and Simulation

Another innovation includes the early modeling of systems so that the architecture can be analyzed and communicated across different teams as the "single source of truth" for a complex project. The model could be used, not only in the early stages in the design, but throughout the lifecycle of a program to allow early virtual verification of a system design. As shown in Figure 9, as the system is realized the model is continuously updated and the realized system could eventually be used to verify and validate the model. An advantage of using such a framework would allow for the ease of analysis when changes have to be made to the architecture at anytime in the lifecycle. Timing analysis, modal analysis, reliability analysis, human-machine interaction and other analyses could be conducted with the model. Integration analyses and some verification could be executed before real equipment is built. Also, future modifications to the aircraft could be incorporated with the living model providing a better quantitative understanding of the impacts before embarking on a detailed development and implementation. The model could aid in generating the tests. Additionally, SIL, ground and flight tests could be used to verify and validate not only the realized system but also the model.

Standardize a Common Modeling Tool Exchange Protocol

Since various companies and groups within organizations use different modeling tools, a common, standardized translation of the models, using such mechanisms as Extensible Markup Language (XML) Metadata Interchange (XMI), could allow ease of communication of the models to promote a common model exchange. The Object Management Group (OMG) is encouraging such reform in modeling languages such as Systems Modeling Language (SysML), Architectural Analysis and Design Language (AADL), MatLAB/Simulink and others.

Require Certified Test Labs

Test labs or SILs will always be instrumental in verifying and validating complex systems in the future. With the volume of testing required and the need for repeatability, it is important that more automated tests are conducted in the lab before expensive ground and flight tests are executed. Likewise, systematic manual and stress testing targeting critical operations is crucial. It is important that a program realizes that all bugs cannot be found no matter the level of testing performed in such complex systems. Nevertheless, testing will always be essential for performance demonstration. Well established rules for verifying and validating the labs, the tools, simulations, and automated, manual and robust methods must be established to increase the confidence in systems before executing expensive ground and flight tests. PMOs and contractors should account for this validation and verification effort in their contracts up front.

Develop Better Automated Testing Capabilities

Some problems can be circumvented by using early, virtual modeling of the system. Yet, airworthiness qualification will always require developmental and demonstrated capabilities through test. The upfront cost for automating tests is expensive but the payback occurs in the lifecycle of a system. While we need to get out of the mode of thinking that we can test bugs out of a system, testing will always be essential. Automated testing with fault injection and robustness testing should be considered to allow for repeatability and determinism in the testing regime.

Improve Current Guidelines and Standards

One last notional contribution is in the area of standards improvement. As shown in Figure 10, systems and software engineering standards have evolved over time. In the period of 1994 to 1996 a separation of standards occurred, in part due to the encouragement for the DOD to accept Commercial off the Shelf (COTS) solutions. After migrating from military standards, different standards organizations adopted their own standard or handbook which loosened the requirements. Some became merely guidelines with no requirement to follow. It has been found that the level of rigor once desired by military standards has not been enforced and sometimes results in technical cost and risk to programs. In 2005, the US DOD eliminated the waivers to cite military specifications and standards in solicitations and contracts per DOD Policy Memo 06-3. This allows the government to once again cite military standards as part of contracts. However, the older military standards do not fully apply to later, more complex systems of today or to the even more complex cyber-physical systems of tomorrow. It benefits the governing standard bodies to revisit these standards to incorporate the guidelines and standards needed to handle complex systems. Considerations for the verification, validation, qualification and certification should be made while these updates are being made to the standards.

Many of the standards that exist within the military rotorcraft community are based on older federated architectures and need to be updated to align and exceed current commercial aviation standards. Some consideration for complex systems standards may consider incorporation of better upfront safety analysis, complexity metrics, and improvements in system reliability and airworthiness metrics to aide in the qualification and certification release process. The methodology and theory will not be easily achieved but are desperately needed.

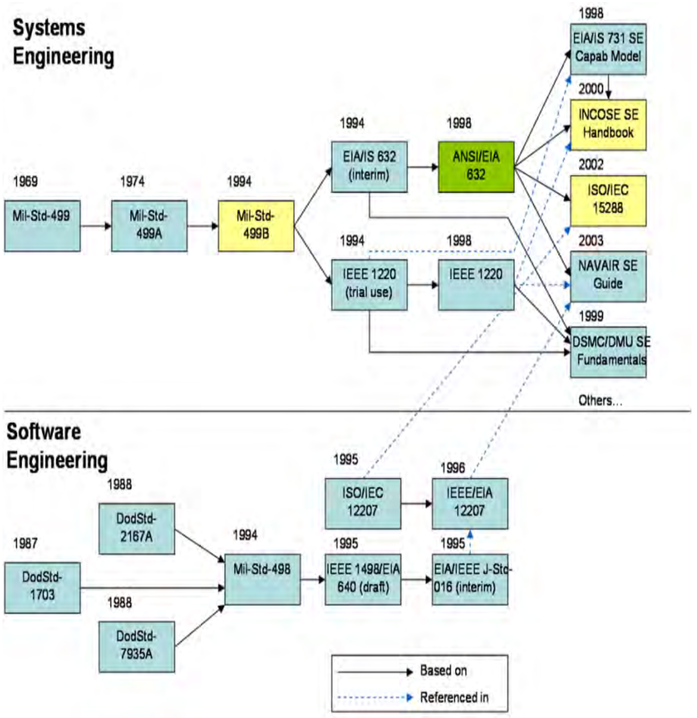


Figure 10 – Systems and Software Standards Evolution [17]

7. Current Work Addressing the Challenges

It should not be misconstrued that no work being done to address these challenges. The following discusses some current programs or approaches addressing the complex system challenges.

The S-92 and FAR Part 29 Update

The FAA Advisory Circular (AC) for Federal Advisory Regulation (FAR) Part 29 was based on precedents of helicopter certifications over the past 35 years. This made the certification process difficult for DFBW projects such as the Sikorsky S-92 project [19, 20, 21]. The S-92 is a DFBW aircraft with Advanced Flight Controls (AFC) that the advisory circular did not address. The challenges identified included: (1) complex, highly integrated systems and functions, (2) management and reliability of redundant components, (3) pilot-vehicle interface and human factors, (4) environmental effects on critical electronic components, and (5) system-structure interactions. With such innovations and the lack of a complete FAA advisory circular, it was difficult for aircraft manufactures to plan for certification. As a starting point, they used examples of the FAR Part 25 for fixed wing DFBW used on the Boeing 777 and Airbus 340. Modifications were made to the FAR Part 29 to handle the system development and qualification procedures. They had to incorporate the guidance of SAE ARP 4754 and ARP 4761 and DO-297 to execute certification for the helicopter while the project was ongoing. Also, they had to address the safety assessment process to show compliance, address AFC consideration for complying with human factors, handling qualities, pilot-vehicle interface, system and flight interaction, and list the rules for special conditions. A similar exercise needs to occur for military rotorcraft guidelines with consideration of the updates in FAR Part 29.

DO-297 IMA Considerations

In conjunction with the Boeing 787 and Airbus 380 projects the RTCA and FAA realized that a need to address the certification of IMA. In 2005, DO-297 IMA Considerations was published by RTCA and adopted by the FAA in 2007 as DOT/FAA/AR-07/39 to provide guidance on these systems. It promotes incremental acceptance of IMA via a 6 step process including: (1) module acceptance, (2) application software and hardware acceptance, (3) IMA system acceptance, (4) aircraft system integration, verification and validation, (5) module or application modifications, and (6) module or application reuse.

DO-178B Update to DO-178C

DO-178B, "Software Considerations in Airborne Systems and Equipment Certification", has been used since the 1990s as a guideline. Software development processes, languages, patterns and tools have since evolved. DO-178C is being worked to allow for early modeling and testing in the System-V cycle. Major supplements to DO-178B with the DO-178C recommendations will include: (1) formal methods which provides a more formalized process coupled with testing to reveal issues early, (2) model based development to allow for early stage prototyping on and executable model to effectively identify defects, (3) object oriented technologies to focus on C++, Java, Ada 2005 that support polymorphism, inheritance, generics, and function dispatch, and (4) tool supplement to address static code testing, auto code generation and other aspects of development and testing. Its expected completion date is by the end of calendar year 2010 [30].

Mixed Critical Architecture Requirements (MCAR)

The AFRL MCAR [22] and its follow-on Small Business Innovation Research (SBIR) activity is concentrating on methods to more quickly develop, test, qualify and certify UAVs of the future such as the Global Hawk (see Figure 11). While their struggle is to resolve problems for UAVs, it is very applicable to manned aircraft too. They stress a "design-for-certification" approach utilizing a library of pre-certified, composable, high confidence, real-time operating system and middleware components. These ideas are an outgrowth of Rushby's idea of composable certification in [16]. Their goal is to expand verification and validation and system certification processes, mature hierarchical model-based system development to ease integration, verification and validation, enhance reusable software middleware in conjunction with upgradeable building block hardware and enforce security and safety requirements. Their idea is to be able to compose a system with underlying certified subsystems similar to the idea shown in Figure 11 while keeping in mind the emergent behavior of the system as a whole. A system can be composed of correct and certified parts but still be unsafe. Therefore, attention must also be placed on the end-to-end emergent behavior of the system.

US Air Force Photo

Figure 11 - Global Hawk UAV

Another key element in mixed critical complex systems is the use of time and spatial partitioning [15]. Modularity is desirable and should be incorporated, but should not be confused with partitioning. ARINC 653 partitioning ensures that operating system process scheduling and memory are not violated. Older federated systems were physically partitioned so that one system could not directly affect another. This is no longer the case with IMA, since multiple software modules may share the same physical memory and processor. To ensure that safety critical items are group together in the same logical partition a schema in compliance with ARINC 653 with an Application Executive is provided to ensure the execution time and memory of a partition is not violated. Theoretically, if a certified partition is left unchanged when adjacent partitions are revised, only minimal testing of the intercommunication between the partitions is needed for the unchanged partition.

Systems Architecture Virtual Integration (SAVI)

The Aerospace Vehicle Systems Institute (AVSI) SAVI project is a consortium of aerospace companies including Airbus, Boeing, Rockwell Collins, British Aerospace Engineering (BAE) Systems, FAA, NASA, and the US Army who are investigating innovative approaches to more quickly design, develop and test complex systems [23]. It's objectives are to (1) reduce cost/cycle-time and risk by using early and frequent, virtual integrations, (2) detect errors as early as possible with minimal leakage to later phases, (3) create and integrate models from the integrator and suppliers using well-defined semantics that support component based, quantitative and formal analysis, (4) facilitate auto-analysis and code generation to identify and eliminate inconsistencies, (5) provide automated compatibility analysis at the architecture level, and (6) develop an acquisition process that supports this virtual integration approach. The SAVI paradigm necessitates an architecture-centric, multi-aspect model repository as a "single source of truth". This will reduce the confusion of having multiple architectural models developed at different times generated in the analyses performed by separate work groups in a project. Also, a component-based framework in support of model-based and proof-based engineering will provide the design and test engineering community with the tools necessary to accomplish better engineering of complex systems. To ease the interchange of information among different organizations or groups within an organization that want to use different model tools, a model exchange protocol concept is supported for consistent model interchange between repositories and tools. SAVI breaks the current mode of thinking so that an architecture-centric, as opposed to a document-centric, acquisition process throughout the system lifecycle will be supported by industrial standards and tool infrastructure. SAVI is using the SAE AS5506 Architecture Analysis & Design Language (AADL) [42] to capture the single source of truth model and is investigating additional languages to integrate into additional domains of specification for a cooperative system engineering approach.

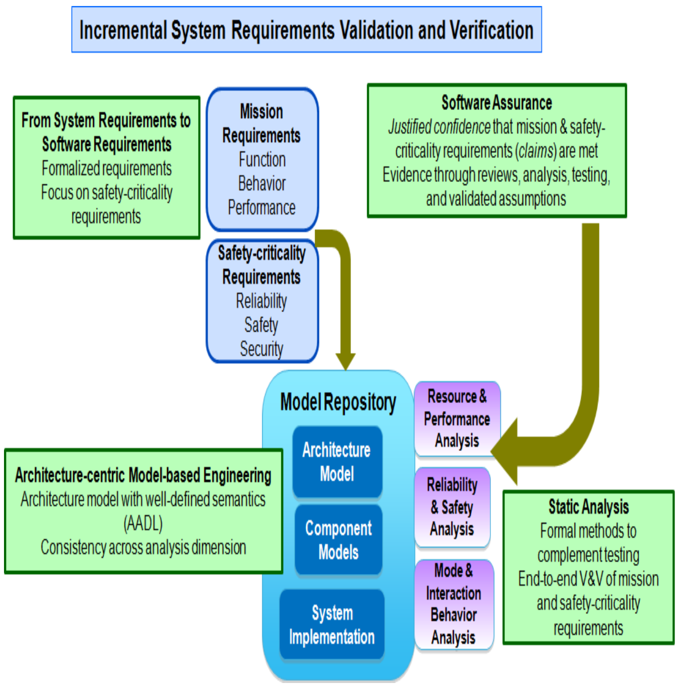

Incremental System Requirements Validation and Verification
From System Requirements to Software Requirements
Formalized requirements
Focus on safety-criticality requirements
Mission Requirements
Function
Behavior
Performance
Safety-criticality Requirements
Reliability
Safety
Security
Software Assurance
Justified confidence that mission & safety-criticality requirements (claims) are met
Evidence through reviews, analysis, testing, and validated assumptions
Model Repository
Architecture Model
Component Models
System Implementation
Resource & Performance Analysis
Reliability & Safety Analysis
Mode & Interaction Behavior Analysis
Architecture-centric Model-based Engineering
Architecture model with well-defined semantics (AADL)
Consistency across analysis dimension
Static Analysis
Formal methods to complement testing
End-to-end V&V of mission and safety-criticality requirements

Figure 12 - CM-SEI ReliabilityValidation and Improvement Framework

<u>Reliability Validation Framework</u>

CM SEI and the US Army are working another effort in conjunction with the SAVI project called the Reliability Validation Framework (see Figure 12) [41]. This involves research on an approach to establish an industry standard practice of reliability validation and qualification for software-reliant mission-critical and safety-critical systems by: (1) establishing an engineering framework for reliability validation and improvement, (2) demonstrating its feasibility of reliability validation through model-based architecture analysis, and (3) proposing a set of metrics for cost-effective reliability evaluation and improvement. It will integrate state of the art technologies including formalized functional and non-functional requirements. A system-focused safety analysis is to be included. Architecture-centric model based analysis, as with SAVI, will be included. Assurance cases are used to capture the rationale and type of evidence needed to support system reliability and safety levels. Future work could develop assurance case patterns for use on similar systems. Lastly, formalized static analysis to complement build-then-test practice so that testing can be reduced saving manpower and time will be employed. For this framework AADL is at the center of the model with a machine processable, single source of truth, annotated architectural model with well defined, well formed semantics for quantitative assessment (see Figure 13). Properties integrated into the model provide the data to drive specific analyses, each using the architecture definition, such as safety, reliability, interaction behavior, security, resource consumption, real-time performance and data quality.

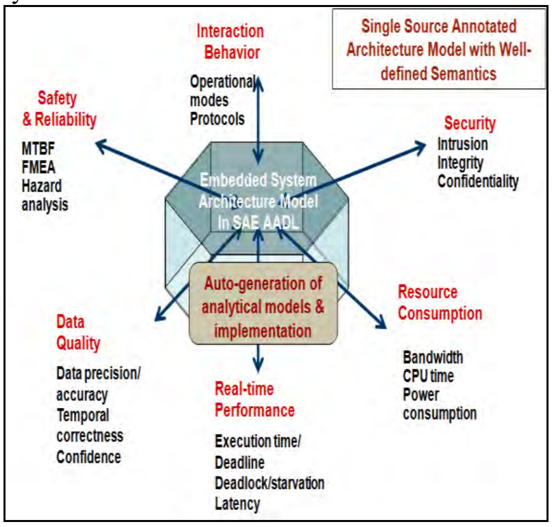


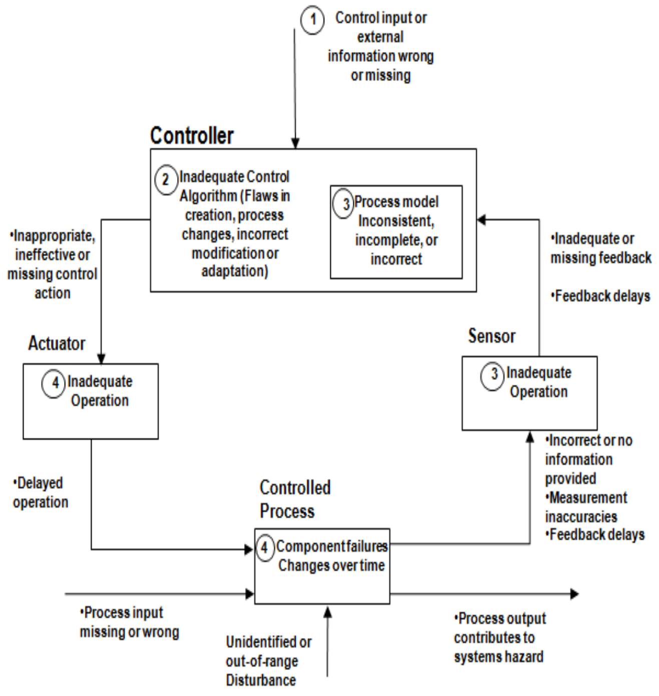
Figure 13 AADL with Extensions

<u>STAMP and STPA</u>

System-Theoretic Accident Models and Processes (STAMP) -Based Process Analysis (STPA) [26, 27, 28, 29] is a hazard analysis technique for complex software intensive systems that is based on systems theory instead of reliability theory. It is work that originated from Leveson and has been automated in a tool by SafeWare Engineering called SpecTRM-RL.

Rather than approaching hazard analysis from an event based analysis such as Failure Modes Effects Criticality Analysis (FMECA) and Fault Tree Analysis (FTA), STPA uses system theory. As mentioned earlier, safety in programs tend to get inserted after a substantial part of the design is completed. With FTAs and FMECAs much of effort is placed on proving, after the design is initially completed, that the design is safe rather than designing in the safety up front in the requirements stage. Systems are becoming too complex to wait that long. STPA provides a method to take the early requirements and conduct the hazard analysis early during the requirements establishment. STPA is based on Leveson's STAMP approach for accident analysis. The philosophy behind STAMP is that accidents result from inadequate control of safety related constraints on the system design, development and operation. STAMP and STPA have been applied to several projects including the Ballistic Missile Defense program [30], 1994 Blackhawk fratricide incident in Iraq, Space Shuttle Thermal Tile Processing System, the Japan Aerospace Exploration Agency H-IIB Transfer Vehicle for International Space Station [29].

STPA is a four step formal mode process (see Figure 14). It includes: (1) identifying system hazards and translating those to top level system safety constraints, (2) defining a control structure for the system being analyzed using the template shown in Figure 16, (3) from the control structure, identifying the potentially inadequate controls, and (4) determining how potentially inadequate control actions could manifest in the system and developing mitigations. In step (3) inadequate control can fall into one of four areas: (a) a required control action to maintain safety is not provided, (b) an incorrect or unsafe control action is provided that induces loss, (c) a potentially correct or adequate control action is provided too early, too late or out of order, and (d) a correct control action is stopped too soon. STPA provides a rigorous, formal guidance for focusing on the system as a control system and allows the safety analyst to conduct a top down analysis. Unlike FTA/FMECAs the analyst knows when they are finished by considering all possible contributors to hazardous operation from the control loop.



1 Control input or external information wrong or missing
Controller
2 Inadequate Control Algorithm (Flaws in creation, process changes, incorrect modification or adaptation)
3 Process model Inconsistent, incomplete, or incorrect
•Inappropriate, ineffective or missing control action
•Inadequate or missing feedback
•Feedback delays
Actuator
4 Inadequate Operation
Sensor
3 Inadequate Operation
•Delayed operation
Controlled Process
4 Component failures Changes over time
•Incorrect or no information provided
•Measurement inaccuracies
•Feedback delays
•Process input missing or wrong
Unidentified or out-of-range Disturbance
•Process output contributes to systems hazard

Figure 14-STPA Model [27]

Time Triggered Protocol Implementations

With some avionics architectures employing distributed components and an ever increasing requirement for bandwidth a need exists for deterministic data busses within mixed safety critical and non-critical systems. Some bus architectures such as ordinary Ethernet are subject to non-determinism, but with the protocols of time triggered or rate constrained policies as provided with Time Triggered Protocol (TTP) and ARINC 664 this concern over latency and jitter is mitigated. Rushby encouraged the use of time triggered deterministic busses [25]. Dr. Hermann Kopetz of Vienna University of Technology has performed extensive research on Time Triggered Ethernet to replace non-deterministic Ethernet. Standard commercial Ethernet provides ample margin for growth for most avionics systems but does not provide for determinism that can be achieved with MIL-STD-1553 [31]. A variation of Kopetz's ideas has been adopted by Honeywell on the NASA Orion manned space vehicle and the Boeing 787 on Ethernet. This architecture addresses the prioritization and quality of service application for best effort, rate constrained, and time triggered data. Interest in this new data bus technology has been expressed by various rotorcraft manufacturers and the automotive industry as a viable future data bus for safety critical applications.

DARPA Meta Solicitation for Cyber-Physical Systems [33]

In December 2009 the Defense Advanced Research Projects Agency (DARPA) provided a Meta solicitation for industry to respond to obtain research proposals to substantially improve the design, integration, manufacturing, and verification of complex cyber-physical systems, targeting aerospace and defense systems including rotorcraft. The requirement for this is to create a complete shift in thinking on the current approach to complex systems design and verification. Proposals from air and land vehicles were expected to provide a new approach to design and verification of complex cyber-physical systems and demonstrate a dramatic improvement in development time and cost. The use of model-based design method and quantification of levels of complexity and formal methods and probabilistic approaches to enhance verification to reduce expensive and non-achievable exhaustive testing and redesign are key challenges to be addressed by the Meta effort [39].

8. Summary

In summary, we have struggled with the complexity of systems since the beginning of systems engineering in the 1940s. Now is the time to start rethinking the approach to these complex systems. There is a need to quantify the complexity of systems to allow the programs to adjust their approach and expectations of a system. Development and testing challenges exist with complex cyber-physical systems and new approaches such as those captured in DO-178C need to be further established.

Use of modeling through the lifecycle of a program, partitioning, composition of qualified or certified components are crucial for the development of complex cyber-physical systems. We should simplify systems as much as possible instead of growing complexity; although, performance requirements make this difficult. Industry standards need to be revisited and adjusted to these new systems with the understanding that systems of tomorrow will not be feasible in cost or schedule using today's standards and processes. Modeling and analysis with a single source of truth is a potential contributor to the solution. Current efforts are ongoing to address these challenges, and further research should be conducted by industry to converge on a set of best practices, tools and standards to adopt for complex cyber-physical systems with consideration of the human-in-the-loop.

It is of urgent and utmost importance that the qualification and certification challenges are addressed for mixed critical cyber-physical systems. If these challenges are not addressed the ability to design, test, manufacture and maintain future complex, software intensive rotorcraft and beyond will be too expensive and too schedule intensive to realize. The fate of future programs will depend on the path and decisions made today.

References:

[1] Leveson, "SafeWare: System Safety and Computers", Addison-Wesley, 1995

[2] 1991, Systems Architecting, Creating and Building Complex Systems, Prentice-Hall

[3] http://www.incose.org/practice/fellowsconsensus.aspx

[4]http://css.csregistry.org/tiki-index.php?page=What+are+Complex+Systems+%3F&highlight=complex systems

[5] Executive Order 12356, Fed. Reg., 47, 14874 (Apr. 6, 1982). Hereafter cited as "EO 12356."
[6] Moore "Cramming More Components onto Integrated Circuits," *Electronics Magazine*, April 19, 1965
[7] http:// www.ece.cmu.edu/ ~koopman/ des_s99/ sw_reliability/#bathtub
[8] NIST Planning Report, 2003
[9] Galin, "Software Quality Assurance: From Theory to Implementation", Pearson/Addison-Wesley, 2004
[10] Boehm, "Software Engineering Economics", Prentice-Hall, 1981
[11] Stiles, Mayo, Freisner, Landis, Kothmann, "Impossible to Resist: The Development of Rotorcraft Fly-By-Wire Technology", AHS 60th Annual Forum, June 8-10, 2004.
[12] Morse, "Cockpit Design and Redundancy Management for the UH-60M FBW System with Active and Passive Controllers, revision 1, 28 March 2005
[13] Snider, "Future Directions in Tactical Vertical Lift", Approved for Public Release 28 April 2010, FN 4617 http://web.mac.com/stressed/AHS_Home/Past_Luncheons_files/AHS_4_29_Snider_Release_copy.pdf
[14] Kroedl, "COTS Real-Time Operating System and Architectural Consideration", DOT/FAA/AR-03/77, 02/04
[15] Rushby, "Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance", DOT/ FAA/AR-99/58, NASA/CR-1999-209347, March 2000
[16] Rushby, "Modular Certification", SRI International, September 2001
[17] http://www.incose.org/practice/standards/heritage.gif
[18] Federal Aviation Administration (FAA) Advisory Circular (AC), System Design and Analysis, AC 25.1309-1A, 06/21/1988
[19] Osder, "Fly By Wire System Architectures Computer Redundancy and Reliability Issues and Mythology", 15 April 2004
[20] King, McCallister, Arifian, "Civil Certification Methods for Advanced Rotorcraft Control Systems", AHS Forum, June 2004
[21] Boczar, Hull, "S-92 Fly-By-Wire Advanced Flight Control System", AHS, June 2004
[22] Barhorst, Blote, Binns, Hoffman, Paunicka, Sarathy, Scoredos, Stanfill, Stuart, Urzi, "A Research Agenda for Mixed-Criticality Systems", Wright-Patterson AFB
[23] Feiler, Hanson, de Niz, Wrage, "System Architecture Virtual Integration: An Industrial Case Study", Carnegie-Mellon Software Engineering Institute Technical Report, November 2009
[24] Boydston, Lewis, "Qualification and Reliability of Complex Electronic Rotorcraft Systems", AHS conference, October 2009
[25] Rushby, "A Comparison of Bus Architectures for Safety-Critical Embedded Systems", SRI International, September 2001
[26] Leveson, "A Systems Theoretic Approach to Safety in Software Intensive Systems"
[27] Dulac, Leveson, "An Approach to Design for Safety in Complex Systems"
[28] Ishimatsu, Leveson, Thomas, Katahira, Miyamoto, Nakao, "Modeling and Hazard Analysis Using STPA"
[29] Pereira, Lee, Howard, "A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System"
[30] St. Clair, "Growing Complexity Drives Need for Emerging DO-178C Standard", COTS Journal, 11/2009
[31] Kopetz, Ademaj, Grillinger, Steinhammer, "The Time-Triggered Ethernet (TTE) Design, IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2005
[32] Kopetz, "The Complexity Challenge in Embedded System Design", IEEE, 2008
[33] DARPA, "Broad Agency Announcement META Tactical Technology Office DARPA-BAA-10-21", https://www.fbo.gov/index?s=opportunity&mode=form&tab=core&id=dd65d945682e6a95b474bd8d33e21660&_cview=0
[34] McCabe, "A Complexity Measure", IEEE, 1976
[35] Rosenberg, "Army's fatal aviation accidents rise despite push to cut number of wrecks", Hearst Newspapers, Nashville Tennessean, 17 July 2005
[36] Best, Schopper, "Effects of System Delay on Aviator-Related Performance", Naval Air Warfare Center, 6 October 1995
[37] Best, Schopper, Thomas, "State-of-the-Art Glass Cockpits and Human Factors Related Issues", Loral Federal Systems – Owego, 29 Sept 1995
[37] FAA Memo 8110.98, "Addressing Human Factors/Pilot Interface Issues of Complex Integrated Avionics as Part of the Technical Standard Order (TSO) Process", 10 May 02
[38] Francis, Rash, Adam, LeDuc, Archie, "A Comparison of AH-64D and OH-58D Pilot Attitudes toward Glass Cockpit Crew Station Designs", USAAFL, Nov 2002
[39] Bellman, "Making DARPA META Goals Come True: How do we Revolutionize Verification and Validation for Complex Systems?", Aerospace Corporation, Presentation given on 17 June 2010 at S5 Conference, http://www.azimuth-corp.com/conference/S52010/agenda.htm
[40] Airbus data source: J.P. Potocki De Montalk, Computer Software in Civil Aircraft, Sixth Annual Conference on Computer Assurance (COMPASS '91), Gaithersberg, MD, June 24-27, 1991. Boeing data source: John J. Chilenski. 2009. Private email.
[41] http://www.sei.cmu.edu/newsitems/ amrdec_roadmap.cfm
[42] http://www.sei.cmu.edu/library/ assets/AADL_Fact_Sheet.pdf