A Research Agenda for Mixed-Criticality Systems

James Barhorst, Boeing; Todd Belote, Lockheed Martin; Dr Pam Binns, Honeywell; Jon Hoffman, AFRL; Dr James Paunicka, Boeing; Dr Prakash Sarathy, Northrop Grumman; Dr John Scoredos, Northrop Grumman; Peter Stanfill, Lockheed Martin; Dr Douglas Stuart, Boeing; Russell Urzi, AFRL

Mixed criticality systems **(def.):** ***a mixed-critical system is an integrated suite of hardware, operating system and middleware services and application software that supports the execution of safety-critical, mission-critical, and non-critical software within a single, secure compute platform.***

1.0 VISION

Imagine a time in the near future when in order to meet an emerging threat, the US Air Force needs to develop and field a new type of Unmanned Aerial Vehicle (UAV) that can operate in close proximity to large civilian populations and near politically sensitive assets. The vehicle needs to be developed and certified airworthy as rapidly and at as low a cost as possible, while building an UAV meeting particular mission payload requirements. At the same time, confidence and assurance in its vehicle's reliability and performance is paramount.

Software development, verification, validation, and especially certification would be the major costs and risks to be encountered. A key reason is that current certification practices do not leverage reuse of existing design, implementation, and certified components. Given today's technologies and development processes, it might take 3 to 5 years and 10s of billions of dollars to field an essential capability such as this for our war-fighters.

However, if a Design-for-Certification approach had been implemented, a better outcome could be achieved. A library of pre-certified, composable high confidence real-time operating system (RTOS) and middleware components would exist. Tools would exist that could compose and configure the RTOS and middleware layers of the software implementation such that recertification was minimized or eliminated. This infrastructure would feature the safe execution of mixed criticality software, ensure full fault tolerance and fail-safe operation, and have fully predictable run-time behavior and bounded airworthiness certification costs.

The objective of the Mixed Criticality Architecture Requirements (MCAR) program, part of the Air Force Research Laboratory (AFRL) Flight Critical Software and Systems Initiative (FCSSI), is to define requirements for this vision for Design-for-Certification of mixed criticality systems, and to engage the research community in solving the technical challenges that will be encountered.

2.0 OBJECTIVES

The objective of the Mixed Criticality Architecture Requirements program is to lay the foundation for a Design-for-Certification process by defining the architectural requirements for a set of pre-certified, composable building blocks for a high confidence real-time operating system and middleware. Safety and security are assumed system characteristics in future MCAR architectures where applications of multiple criticalities could safely and efficiently utilize the same computation resources. This design for certification process should reduce the effort, cost, and risk of attaining certification for our future applications.

From the USAF's statement of objectives for MCAR:

> *The goal of this program is to begin a new era of architecture definition and systems engineering that will enable future systems capabilities to incorporate a design for airworthiness certification philosophy.*
>
> *The objectives of this delivery order are to:*
>
> *1) Study, analyze and define requirements for a compose-able architecture capable of incorporating a mixture of safety-critical and non-critical systems/functions, while operating in a secured information environment.*
>
> *2) Extend the architecture requirements to institute a design for certification perspective, and to levy this perspective to the architecture's hardware, software and firmware components.*
>
> *3) Establish those necessary architecture requirements that will ensure systems' functional efficiency and overall affordability to certify and implement.*
>
> *4) Institute teaming and collaboration to capture the multi-disciplinary approach needed to ensure a safe and secure mixed critical architecture, and to engage collaboration and cooperation with National Science Foundation through the use of government supplied working group meetings.*
>
> *This effort provides the aerospace industrial community the opportunity to engage the academia to conceive, define, and investigate a process capable of establishing a set of Mixed-Critical Architecture Requirements (MCAR). This effort will entail identification of a mid-to-far term capability, and assembling a team of multi-disciplinary experts from industry and academia to support defining an architectural option to achieve this future capability.*

The primary objective of this white paper is to inform the reader about MCAR; in particular, to present a set of research challenges that would complement and enhance our current MCAR work. These technical research challenges are presented in Section 5.0, for your potential participation.

The reader should come away with an understanding of:

1. The issues associated with mixed criticality,
2. The USAF need and plan for Design-for-Certification, and MCAR's relationship to the FCSSI,

3. The current participants in MCAR,
4. Some characteristics of an MCAR system,
5. Some fundamental technical problem areas in MCAR, and
6. An approach for addressing these areas with your assistance.

3.0 RATIONALE

Future aerospace systems are expected to exhibit increased richness of function and meet increased levels of performance. All of these drivers are in some part responsible for the near exponential increase in size and complexity of software on such embedded systems. Increased software size and complexity for aircraft systems renders the task of certifying them significantly more challenging and costly. Particularly for smaller UAVs, an issue complicating certification is that Size Weight and Power (SWaP) considerations are leading to mixed criticality solutions, and current certification paradigms can require certification of co-resident functions to the level required for the highest criticality level present. One of the drivers for increased system functionality and software complexity is the need for increased levels of autonomy aboard military aircraft. UAVs are currently used by the militaries of the world, however, all of these semi-autonomous systems are only fore-runners of truly autonomous vehicles currently on the drawing boards of major aircraft vendors. A key requirement for the next generation of UAVs is the enabling - through onboard software - of higher cognitive functions normally performed by a pilot (in air /on ground). These higher level cognitive functions are hallmarks of true autonomous systems capable of self-determination, decision-making, self-awareness and self-assessment, giving rise to real problem-solving automata. Not all systems will need to have autonomy in every aspect of their operation. Autonomous and automated systems go hand-in-hand and can be considered as differing only in scope of their “autonomy”. Most military systems will without a doubt include some deference to a human-role; however the ability of this human to effectively *control* any onboard processes is unlikely given communication channel latencies. It is more likely these humans are supervisory in nature and provide some level of meta-reasoning capability.

Avionics systems belong to a larger class of systems commonly referred to as *cyber-physical systems* (CPS). This alludes to the close binding between the *physical* phenomena and systems with the computational or *cyber* capabilities that control them. Most embedded systems can be considered as instances of CPS. Most CPS have some notion of criticality, namely there are aspects of the CPS whose failure leads to a negative impact on the system's environment. This loss can impact human *safety* as in the case of loss of control of aircraft systems (in flight). At the same time CPS can be characterized in terms of a *mission* or purpose, which the CPS achieves by suitable execution of its functions/activities. Some functions are critical for mission success, without affecting the safety impact of the CPS. This attribute of *criticality* is often used to characterize elements of CPS systems. For example, in avionics, engine control and control of flight surfaces are safety critical, but navigation and communications is (generally) mission critical. These avionics systems pose the highest levels of safety and security challenges in the CPS domain.

3.1. AFRL MCAR Program Roadmap

AFRL has been actively pursuing the FCSSI since 2004 with the following objectives: (i) unify UAV embedded system development and expand verification and validation and system certification processes, (ii) mature hierarchical model-based system development to ease integration, verification, and validation, (iii) enhance reusable software middleware in conjunction with upgradeable building block hardware and (iv) enforce security and safety requirements. This program has two phases; Phase I – improve the certification process with innovative techniques and methods; Phase II – design systems for certification. Phase I was primarily composed of the Technology for Affordable and Safe Software (TASS) Small Business Innovative Research (SBIR) programs, Verification and Validation of Intelligent and Adaptive Control Systems (VVIACS), and Certification Techniques for Advanced Flight Critical Systems (CerTA FCS) programs. Phase II started with the MCAR program and is designed to lead into the Mixed Criticality Architecture Design/Development (MCAD) program as defined in the roadmap in Figure 1.

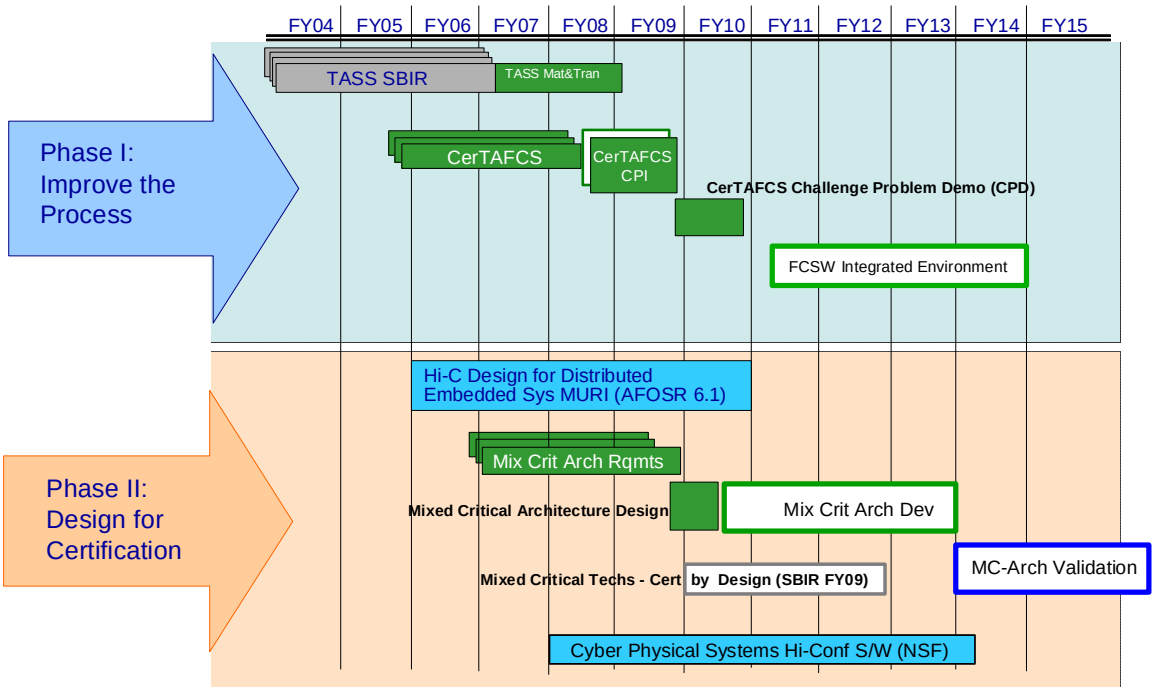

Figure 1: AFRL FCSSI Roadmap

3.2. Relevance to Other Efforts

As indicated earlier, avionics systems are a type of CPS. There has been significant activity since late 2005, championed by the National Science Foundation (NSF), the Networking and Information Technology Research and Development High Confidence Software and Systems Coordinating Group (NITRD/HCSS CG) and other organizations, to lay the semantic foundations of this area. Key application areas of interest are aerospace, automotive, chemical processes, civil infrastructure, energy / power generation systems, healthcare / medical devices, manufacturing, transportation, entertainment, and consumer appliances. Much work has been done under the aegis of NSF CPS workshops as well as active research funding by the NSF under the Computer and Information Science and Engineering (CISE) and Engineering (ENG) Directorates with focus on Foundations; Methods and Tools; and Components, Run-time Substrates, and Systems.

The AFRL MCAR program and community have sought to closely dovetail their efforts with the CPS researchers in various vertical domains including automotive and rail transportation.

This shared vision is spurred on by the need to attract focused research and forge ahead in a host of areas critical to furthering the mixed criticality agenda.

4.0 MIXED CRITICALITY – TECHNICAL BACKGROUND

To bring the technical background for mixed critical systems into focus, we will describe the domain of UAVs, then explore the emergence of mixed critical systems in this domain, follow that with a discussion on what many would call one of the most vital areas of this domain: system certification, and finish with some mixed criticality system characteristics that could help frame this problem for researchers and developers.

4.1. Technical Description of the Domain

In the development of military air vehicles, safety critical systems have historically been separated from mission critical systems. This is designed and implemented through various methods including but not limited to hardware, operating system, middleware, and application layer constructs. This separation is desirable from the standpoint of certification as it serves to delineate the higher critical processes from lower critical ones. The presumption is that this separation prevents a lower criticality function adversely affecting the higher criticality function leading to unpredictable behavior. Such interactions can compromise or, at the very least, significantly complicate the certification effort.

Historically, this separation has been considered an absolute, where early implementations were accomplished by separating safety critical from mission critical functions physically. This would separate the functionality, including all logic and processing. This created well defined divisions in UAV systems. These divisions manifest themselves as the two main groups of avionics in a typical UAV. The safety critical systems were grouped together under the vehicle management systems banner. These systems include; flight control, actuation control, engine control, utility and subsystem control, electrical power system control, and other critical vehicle systems. The mission critical systems were grouped under the mission systems banner. These systems include; mission management, weapons management, navigation, radar systems, countermeasures, multi-functions displays, communication (vehicle to ground and vehicle to vehicle), and other mission functions and systems. Not only have the safety critical functions been separated from the mission critical functions, but the similar components have been

separated within the different criticality groups. These criticality groupings of systems are the state of the art and the standard for the development of most UAVs. Safety-critical and mission-critical systems have traditionally been physically separated in multiple ways. Each system had its own physical hardware and software components and interfaced with the other systems via external buses, and I/O (analog and discrete). Over time, the inherent inefficiency of this approach led to the concept of consolidating like functions into a federated Vehicle Management Systems (VMS) computer and federated Mission Systems Computers. These exposed new internal issues and drove the need for process isolation in these federated systems. One way this process isolation has been implemented is ARINC 653 [1], a standard for time and space partitioning for avionics systems. In this approach all applications' safety critical and mission critical segments are allocated to separate processor containers and the data-flow between these containers was tightly controlled through a combination of middleware and OS / kernel level mechanisms.

Another driving factor in the UAV domain is the constant battle to reduce weight and volume of the embedded systems. Any reduction in the weight and volume of the UAV's systems translates directly to more range (more fuel capacity, or fuel efficiency), greater performance, or capability to carry more/larger payloads (sensors, weapons, etc.) These added capabilities are all desirable in the eyes of the customer and user communities. Any new technologies that could help enable future UAVs to reduce their footprint would be highly desirable.

4.2. What it Means to be Mixed-Critical

As mentioned before, the next generation of UAVs is placing demands on UAV systems in terms of software complexity and higher levels of cognitive functions. This increased complexity also introduces many changes to the current paradigm of what defines safety critical and mission critical. Some functions that were considered mission critical are now safety critical. Others blur the line because of the closer coupling between the safety critical and mission critical functions especially under off-nominal conditions. In manned aircraft the human pilot provides the ultimate decision point to de-conflict and separate different criticality levels. In autonomous systems, the cyber-pilot or the embedded UAV software has to thus bridge criticality levels. Consequently, the status-quo of process isolation is no longer adequate or feasible.

Add the fact that these systems continue to shrink into smaller and smaller systems. This continued growth in computational complexity (more functionality) and the weight and volume requirements that are also being levied on systems point to a system that will be a highly complex mixed critical system that has been forced into a smaller physical space. These systems will be required to support safety critical and mission critical functions in the same physical and logical space and require new methods and new tools to support affordable development.

4.3. Certification Considerations

Certification of aircraft systems software often follows guidelines such as RTCA DO-178 [2] or ISO/IEC 12207 [3]. In general the certification is based on examining the software life-cycle processes (requirements, development, configuration management, verification, and validation) used to produce the software, in conjunction with its test data. This analysis is then compared with a safety assessment that is made independent of the software system, usually based on the aircraft certification requirements and general software implementation. This is exemplified by the following excerpt from DO-178B:

> *The Certification authority satisfies itself that the software level proposed by the applicant is consistent with the outputs of the system safety assessment process and other life cycle data…*

Safety criticality levels play a crucial role in the certification process. For DO-178B, the criticality levels are A through E, with level A representing the potential for catastrophic failure including loss of life and/or the aircraft, with level E having no effect on safety. The required safety of a subsystem is typically a function of its criticality and is often stated in terms of the likelihood of mishaps of the corresponding severity. So requirements for criticality A may be to demonstrate a catastrophic mishap probability such that it is unlikely that such a mishap will occur over the planned service life of the aircraft fleet [4]. Producing evidence to demonstrate the required level of safety imposes an increasingly steep V&V burden as criticality increases, therefore affordability demands that system elements not be assigned inflated criticality levels.

The criticality of an element, however, depends not just on the inherent criticality of the function it provides, but it may also be inherited from the elements that depend on it. For software, this includes indirect dependencies, such as side effects due to sharing computational

resources. This is where certification requirements impact solutions for mixed critical systems. Absent separation mechanisms such as those provided by ARINC 653, software elements of lower criticality running on the same processor as a high criticality element would have to be treated as if they were also of the higher criticality. This can significantly increase the V&V effort required over a federated solution. Accordingly, one of the objectives for a mixed criticality solution is to provide separation between elements of differing criticality levels sufficient to guarantee that the lower criticality elements can not interfere with the functioning of the higher criticality elements.

Current methods, such as model-based design, test generation, and execution automation, formalization of some components (requirements, models, etc.), and rapid prototyping have been shown to be effective with the current class of problems. As the problem space of the mixed critical systems expands the software complexity, the future of affordable system certification of complex UAVs becomes a very risky proposition. New methods and tools are needed to make certification an engineering problem again. One such area that could be a true breakthrough is the concept of composable certification. If it could be shown that a particular system meets the desired level of safety by using an existing set of certification artifacts from pre-existing components, this could significantly reduce the testing needed to certify that system, and provide developers an affordable certification path forward. Having this type of composable certification scheme would be a highly desired method/tool for future complex mixed critical systems.

4.4. Characteristics of Mixed Criticality

The MCAR program has identified a number of characteristics which capture the challenges associated with defining, developing, and certifying mixed-criticality architectures and systems. The characteristics are not independent, and have a number of subtle and not so subtle interactions that impact the development and certification processes. The current set captures and extends concepts now used for developing and certifying avionics systems.

4.4.1. Safety Characteristics

Safety violations are initiated by components known within the system that attempt to effect unintended changes within the system. Safety violations need not be, and are often not, deliberate attempts of sabotage. Safety boundaries are defined to protect one portion of a system

from the (likely inadvertent) misbehaviors of other portions of the system. The purpose of safety boundaries are to ensure that a subsystem has available hardware resources when it needs them and that the data (and instructions) it generates and uses are not contaminated by other system components. Safety boundaries are defined for multiple dimensions of system resources (e.g., time, space, data flow, and temperature containment). System safety is often intimately related to fault containment and consequently, to fault tolerance. Demonstrable separability (as we have defined it) is frequently part of the demonstration of safety, but it need not be directly related to modularity, or composability.

4.4.2. Security Characteristics

Multiple levels of security are a critical need for flight systems. Data of several different security levels, i.e., CONFIDENTIAL, SECRET, and TOP SECRET, may be present on the same system. Care must be taken to ensure the correct interaction between security and other mechanisms, including mechanisms that separate safety criticality levels.

Security problems usually have their initial cause in an intentional act from outside the system, while safety problems usually have their root in an unintentional defect inside the system. This distinction has had a significant impact on the different approaches to security and safety mechanisms and consequently V&V. Security is focused on data flows, while safety looks at the unintended behaviors of software and systems. On the other hand, there are also significant commonalities and opportunities to leverage the two approaches.

4.4.3. Reconfigurability and/or Adaptability Characteristics

Reconfiguration and/or adaptation can be part of nominal system operation (e.g., mode changes in response to a detected set of triggering conditions, response to recognizing or learning about new situations), or a response to detected threats or faults.

Explicit rules are required to ensure that reconfigurations are timely, safe, and secure. The reconfiguration rules are captured as analyzable models in addition to being encoded within an implementation. Defining precise points or events at which reconfiguration should occur is not always straightforward, but must be encoded as such to be implemented as software. For example, if a gyro begins to drift, when is the drift sufficient to merit transition to use of a lower quality sensor? Finite State Machines (FSMs) are too limited to capture a realistic notion of

many system states. Vectors of real-valued or complex-valued numbers are also not sufficient to capture time evolving notions of system state. Sequences or their differences with time progressions somehow need to be represented. Based on our experience, most system state representations manifest themselves as ad hoc data structures which capture limited aspects of system behavior.

There may be multiple aspects of system state, and the reconfiguration might depend on all aspects. For UAVs that can learn, or even those designed to tackle complex operational scenarios, testing of all possible transitions from one configuration to another will not be possible, as the number of possible transitions will be prohibitive. Analysis must be used to show that transitions from one system state to another are safe in the sense that the newly configured system remains stable. In some sense system reconfiguration is dynamic when transitioning from one configuration to another that has never been tested before. In another sense, the configuration class can be analyzed at design time for assuring certain properties. Reconfiguration timeliness is a measure of how quickly the system can reconfigure itself relative to either the start of reconfiguration or the time by which reconfiguration is required to respond, and is generally also required for safe transition.

In general, one would tend to think that the more adaptable a system is to a diverse range of possible events, the more complicated its functionality might be to enable a greater variety of adaptation. A complicated system will often be more difficult to certify. Adaptability tends to improve system robustness, but often at the expense of efficiency.

4.4.4. Availability and/or Reliability Characteristics

Availability and reliability are really measures of how safe a system is in terms of our expectations for its continued safe operation over periods of time in the face of a set of possible faults. Availability and reliability can also be regarded as measures of a system's design for fault tolerance. Reliability and availability are such important measures, that an entire specialty has evolved to estimate these metrics. Numerous commercial tools are available for fault analysis. Many academic tools are also available, and they typically offer more complex fault models but are less robust.

In the traditional development process, there is only a loose coupling between architecture specifications, including fault tolerant design and safety mechanisms, and the various fault models that provide measures of availability and reliability. The connections are rarely explicit, and the models may lag the implementation and development efforts by several months. When a fault analysis produces an unacceptable value, there is rarely a direct, explicit linkage back to the architecture specification indicating the part of the design that is contributing significantly to the high fault probabilities. There is also a notable gap in the integrations of different error models. System failure models are often not linked to software or communication delay faults. Another significant gap is in availability and reliability metrics for real-time embedded software, and the contribution of software to overall system metrics such as probability of loss of control.

4.4.5. Distributed Characteristics

There are varying degrees of being distributed in terms of both physical proximity and amount of distributed control or coordination. Physical distribution ranges from under a few millimeters (for core-to-core communications) to possibly a few hundred kilometers (for inter-vehicle communications). Distributed coordination and control ranges from coordinating protocols for two nodes to possibly hundreds of nodes, with the degree of coordination ranging from distributed only to establish central control to highly distributed peer-to-peer communication protocols. The suitability of a distributed protocol will depend on the degree of physical distribution, which affects communication time scales.

4.4.6. Situationally-Aware Characteristics

Situationally-aware functions included in a vehicle mission management system will depend on the operational contexts in which a vehicle is planned to fly. Operational context defines rules for decision making. Providing a specification language to encode situations and states, along with tools to "check" the correctness (i.e., completeness and consistency) of the specification could benefit development and certification efforts by facilitating analysis earlier in the process.

Certification aspects and requirements may also depend on the operational settings of use. If certain functionality is deployed only in battle zones and never in commercial air space, the certification requirements for ensuring the correctness of these functions might be substantially reduced.

4.4.7. Affordability Characteristics

The primary focus of MCAR is affordable certification of UAVs. However, the cost of the certification effort can be linked to costs of all phases of the system software life cycle, including design, development, deployment, and upgrades. We recognize that some of these lifecycle processes may change, but the core aspects of specification, design, and verification will not vanish. Several of the earlier characteristics will contribute to affordability of the life cycle processes, including certification. In particular, we anticipate composable, yet demonstrably separate modules will significantly aid in the design, analysis, development, and certification processes. Further, we see fault tolerance, security, and safety characteristics as assisting in the certification process by providing analyzable capabilities, and thereby supporting evidence that the assumption of a certain degree of separability holds. Another affordability characteristic is the existence of techniques and methods that can support synthesis and automated testing in ways that will be acceptable to the certification agencies. Existing techniques and methods, such as model-based design tools, compiler environment tools, and middleware synthesizers, have been used to speed up the development processes. In some cases, the automated output is accepted, but typically not without undergoing careful scrutiny and some modifications. Tools also can assist in design analysis. Included in this category are model checkers, scheduling analyses, fault system analysis, and general resource use analysis tools.

5.0 FUNDAMENTAL TECHNICAL PROBLEM AREAS

As a result of our MCAR requirements gathering activities and current MCS (Mixed-Critical Systems) R&D, we have identified six broad, partially overlapping categories of MCAR challenges that offer areas of research into fundamental technical problems. Finding solutions for these challenges is key to achieving the objectives of the FCSSI and need to be addressed jointly by the MCAR program teams and the research community.

5.1. Technical Problem Area: Certification of Composable and Configurable Design, Software Implementation, and Hardware Components

MCAR envisions future systems will be constructed using composable and configurable components with pre-determined levels of certification. Components and associated metadata will comprise the basic initial set of building blocks for future UAV development and

certification. The overarching challenge is to identify, develop, and implement both a certification process and a composability framework that will support composable and incremental certification. A systematic means to identify and assess component *certifiability* levels in designated contexts, even before they are implemented, is necessary to support composable and incremental certification. Formal and analytical methods for embedding attributes of certifiability within components (or their specifications) are envisioned to play a key role in the certification process. These certifiability attributes will have to include the system context (hardware, etc.) that characterizes the applicability of the attributes.

All avionics systems are designed with dynamic reconfiguration capabilities which, for example, might be activated in response to detected fault conditions or to a change in contingency plans. Currently, certification costs grow with the number of possible configurations of a system. As the set of UAV mission scenarios become increasingly diverse and challenging, design trends will call for more dynamic reconfiguration capabilities. Classes of reconfiguration techniques, including conditions and monitoring mechanisms for reconfiguration initiation need to be defined and abstracted to a core set of standardized approaches to dynamic reconfiguration. The developed core set must span the majority of circumstances requiring reconfiguration as well as balance system complexity and system development and certification costs. Tools and techniques to certify classes of reconfiguration designs, mechanisms, and their implementations are needed to make the certification process more uniform and more cost-effective.

5.2. Technical Problem Area: Certification of MCAR System Architecture Families

MCAR requirements evaluations with potential users indicate a need for a *core* system that is configurable to a wide range of UAVs with differing missions and flight durations, differing embedded computer platforms including multi-core, and differing mixes of criticality levels depending on where piloting is performed. The structure of a core system adds further semantics and refinements to the MCAR composable components model. The notion of leveraging a core system architecture across a range of vehicle systems and mission can be viewed as defining a *Product Line Architecture* (PLA), which defines a *product family*. Systems within the family share a set of common core capabilities, called *commonalities*, and each is differentiated by a

unique set of product-specific capabilities, called *variabilities*. These systems present significant challenges for "modular analysis" in support of V&V and certification activities. More generally, capturing the range of target vehicle platforms with cross-cutting requirements, identifying their commonalities and variabilities, and subsequently determining software and hardware component selection and binding is a significant challenge. This is because the selections and bindings are subject to many potentially conflicting multi-dimensional requirements, where the goal is construction of a composable and incrementally certifiable system architecture.

In practice, development organizations that adopt PLAs do so because significant reductions in system *building* development costs using PLAs have been observed. Many organizations simply treat a system built using a PLA as any other system where they analyze and test a PLA system as any other individual system. However, the state-of-the-practice in V&V of PLA-based systems fails to exploit the product line structure to reduce V&V costs because the reusable aspects of the core architecture do not receive certification credit. The research literature on testing of PLA-based systems is extremely limited and we are unaware of any literature that addresses reusing the results of testing for certifying multiple product line products. This is a missed opportunity because product lines, by definition, define the space of possible systems as combinations of variabilities.

PLAs define a skeletal structure for all products in the family. Can safety and security analyses be aligned with PLA structure to enable reuse of portions of an analysis? In a mixed-critical PLA, different analysis techniques might apply to elements of different criticality levels. Also, can criticality be exploited to vary the cost and effectiveness of analysis across a PLA?

5.3. Technical Problem Area: Unification of Safety and Security in MCAR Systems

In future configurable aerial systems, the distinction among criticality levels for components will be blurred and variable depending on which function a component provides relative to the particular set of mission or operational requirements. In UAV systems with mixed criticality levels, safety is still likely to be the predominant concern yet security impacts safety and some UAV mission operations may mandate security and survival over safety. Techniques and tools have been developed and studied that characterize both system safety and security levels, but

little work appears to have been done that leverages now disjoint security and safety certification credit by identifying the common approaches used to address both characteristics.

A unified approach to safety and security that combines the views of threats and hazards might leverage separation mechanisms to satisfy both safety and security requirements. Separation of tasks with different criticality levels is currently accomplished through a federated architecture (physical separation) or through the use of run-time enforcement mechanisms coupled with design time analysis, such as those found in a partitioned RTOS. Information flow analysis tools have been developed for verification of software in the security domain. Worst case execution time (WCET) analysis tools have become quite sophisticated and can incorporate CPU caching and pipelining. Cache sharing in modern multi-core processors is a source of interference in both the safety (time and space partitioning) and security domains.

Current analysis and measurement-based methods need to be extended to establish, to an acceptable level, all required separation guarantees among software functions that share resources. Information flow pattern analyses are needed to determine all possible interactions between functions and shared memory, whether directly communicating or not. Similar analysis techniques might be useful in determining non-interference run-time guarantees required for constructing WCET estimates. The highest levels of safety often have stringent timing requirements that might not be applicable to security requirements. A standard set of core approaches that simultaneously satisfy system safety and security isolation guarantees at varying levels of criticality could reduce system development and certification costs.

5.4. Technical Problem Area: Modularity Trade Studies of an MCAR System with Respect to Hardware, Software, and Certification Gains

Modern systems are designed so they can be semi-automatically configured with platform and application-specific data. For example, tool generated configurations may include the binding of communication ports between applications, the mapping of applications to processors, or defining accessible memory limits for an application. It is often the case that configuration of certain information is *assumed* independent (i.e., it does not affect) other aspects of system behavior, but in fact configuration decisions can lead to obscure and hard to find errors in a deployed system. For example, (mis)mapping of local application data to a shared address space can lead to unintended interactions.

In practice, detecting unintended interactions is often limited to V&V on a fixed set of configuration scenarios identified during requirements analysis, often representing common use-cases of the system undergoing certification. The space of possible configurations grows exponentially with the number of configuration parameters and their assignment ranges. To systematically explore interaction effects among configuration parameters, some researchers have applied Design of Experiment (DOE) testing techniques. For example, DOE techniques have been applied to investigate feature-interaction in telephony systems to guarantee coverage over all combinations of configuration settings. More recently, combinatorial DOE interaction testing approaches have been applied to widely-deployed configurable middleware.

Prior to final allocation and bindings of hardware and software components for a particular system configuration, it is desirable to optimize the modularity and placement of mixed-criticality components to balance hardware, software, and certification requirements. In the abstract, this is a constrained multi-dimensional optimization problem. In practice, no single optimization tool can solve such a complexly constrained system. The system integrator typically uses a sequence of analysis tools that iteratively lead to a feasible configuration with respect to many diverse requirements. For example, many design and analysis optimization tools have been applied to Architecture Analysis and Design Language (AADL) [5] specifications.

System hardware must be reconfigurable for a variety of UAV platforms families and upgradeable to reduce life-cycle cost. The computer architecture may support multiprocessing allocated to multi-core processors. Application, middleware, and operating system software should be reconfigurable, even dynamically reconfigurable at some level to support a variety of target platforms and missions. Research is needed to determine how system modularity balances between reductions in performance and certification costs, and into effective testing strategies for detecting and validating the absence or presence of interaction effects among components.

5.5. Technical Problem Area: Resource Management, Sharing, and Separation in Multi-Core Mixed-Criticality Systems

Time partitioning refers to scheduling mechanisms that guarantee that each thread or process always receives the amount of CPU execution time that it was promised, no matter what the other threads or processes in the system do. Space partitioning refers to memory management mechanisms that prevent a process from interfering with memory that has been allocated to other

processes. This is normally accomplished via the CPU memory management unit (MMU). Multi-core processors are rapidly becoming the industry standard. What mechanisms are required so multi-core processors can provide support for run-time partitioning?

In multi-core processors, many cores concurrently compete for access to different hardware resources ranging from on-chip caches to off-chip memories and I/O devices. This is analogous to distributed computing, except in distributed systems the time scales are orders of magnitude larger (and possibly with much less covariance) and the sources of contention are identifiable because the hardware devices are physically distinct. In multi-core processors, when a thread allocated to one core has a fixed amount of time, there are no guarantees that the time will not be spent in busy-waits for memory accesses while threads on other cores execute. The problem is exacerbated by the diversity of emerging multi-core processor architectures. Novel approaches to both space and time partitioning in multi-core processors are needed for the highest criticality levels. No commercial multi-core RTOS, certifiable to DO-178B level A, is currently available.

RTOS vendors currently support symmetric multiprocessing (SMP) approaches that allocate tasks to cores to balance workload and maximize throughput. Asymmetric multiprocessing (AMP) RTOSs are also being developed which may be suitable for hosting different applications with distinct computational needs or computing paradigms. For example, consider rehosting two legacy applications on a dual-core processor that were originally hosted on different RTOSs (e.g., an ARINC 653 compliant RTOS and LynxOS). The availability of separate computing cores on a single substrate provides some inherent level of separation (especially for CPU time) that may be utilized to support novel approaches to partitioning at lower criticality levels. It may be possible to leverage new chip architectures to satisfy separation requirements. This may allow lighter weight real-time schedulers to be used instead of a full time and space partitioned RTOS, reducing overhead and further increasing performance.

5.6. Technical Problem Area: Next-Generation Design Analysis and V&V Environments.

Mature well-integrated toolsets and methodologies are needed to aid in the development of cost-effective, certifiable UAV platforms. Tools are needed to compose pre-certified building block components in ways that will produce certifiable subsystems. For example, composition rules with well understood and analyzable behaviors are required for composing guardians

(trusted monitors) and sets of fault configuration alternatives. Rules, acceptable to the certification authorities, for extending architecture subassemblies are needed. For example, when defining a variation on the common core of a product line, what rules govern the incremental certification of the product-specific architecture extensions? Is it sufficient to certify only the glue-code when the product-specific functions are built using pre-certified components? Tools and techniques to qualify components as certified at various levels need to be identified. Assembled collections of components might qualify at higher levels by architectural design (e.g., triple modular redundant (TMR) systems). Individual components may need to be formally verified, or only extensively tested, depending on use. A qualification acceptance standard for off-the-shelf certification artifacts is needed.

Advanced analysis and composability tools must capture the essential compositional information about each component as metadata. Ideally, design tools will use this information to automatically integrate and generate the software system, retaining the modular certification, and automatically generate the certification tests. Tools should be able to predict system behavior and performance to enable both engineering trade studies and rapid development. Similar tasks have been performed in the past by R&D teams consisting of a lead integrator and tool developers (sponsored by DARPA and/or AFRL), including using AADL, SysML [6] or other similar techniques. The collaborative development teams of the future will be distributed and consist of multiple integrators and suppliers (of the MCAR components). New tool developments must be cooperatively integrated and tested.

6.0 APPROACHES TO TECHNICAL PROBLEM AREAS

AFRL Air Vehicles Directorate (AFRL/RBCC) has chosen a two pronged approach to reaching its vision for the future of mixed criticality systems development. First, as part of the FCSSI technology thrust, they have funded the Mixed Criticality Architecture Requirements program to define the requirements for a composable mixed criticality architecture where “safety and security are assumed system characteristics,” including identifying the critical unresolved technical issues discussed above. Second, they are seeking to establish a broad based community across the government, industry, and academia to address the breadth of challenges identified here that must be addressed before that vision can be achieved.

6.1. MCAR Program

The MCAR program is an AFRL/RBCC sponsored research program involving three major airframe suppliers, Northrop Grumman, Lockheed Martin, and the Boeing Company. MCAR kicked off in July of 2007 and will conclude as part of AFRL's Safe and Secure Systems and Software Symposium (S5) in June of 2009. The statement of objectives of the program covers developing requirements for a mixed criticality system made up of composable building blocks designed for certification to safety and security requirements; requirements for a high confidence real-time operating system that provides the separation required for mixed criticality systems, and its interfaces to hardware and middleware; requirements for mixed criticality middleware and its interfaces, including middleware providing communication and redundancy management services; and requirements for design, analysis, and V&V environments.

Each of the contractors is part of a team that brings additional expertise that contributes to the requirements development process. The Northrop Grumman team includes Honeywell, the Lockheed Martin team includes Rockwell Collins, Barron and Associates, Carnegie Mellon University and the University of Nebraska, and the Boeing team includes LynuxWorks, Wind River Systems, Green Hills Software, Objective Interface Systems, and Real-Time Innovations. This combined expertise will allow the contractor teams to develop the requirements for composable and certifiable mixed criticality systems for UAVs, and to identify the research areas necessary for those requirements to be satisfied. The teams have actively participated in a number of MCAR Working Groups which have served as forums for identifying and clarifying many of the issues presented here.

Also as part of the MCAR program, AFRL has chartered the MCAR contractor teams to engage the research community, in particular the researchers sponsored under a number of NSF research initiatives related to the emerging discipline of Cyber-Physical Systems. Next generation UAVs clearly belong to the CPS domain, and many of the research challenges identified in this white paper are CPS challenges. As part of this NSF sponsored researcher engagement, the MCAR contractor teams have reviewed a number of NSF sponsored research programs and invited selected NSF sponsored researchers to present their research at MCAR working groups. As part of this researcher outreach, AFRL and the MCAR contractor teams have also participated in CPS Week 2008, hosting a birds of a feather session laying out the MCAR

vision and presenting an early look at some MCAR research challenges, and participated in the November 2008 HCSS National Workshop for New Research Direction for High Confidence Transportation CPS organized by NITRD. These activities have laid the groundwork for the second prong of the AFRL approach to realizing their vision for future mixed criticality systems, building a broad based community aimed at addressing the full scope of mixed criticality research challenges.

The MCAR program will conclude this spring with the Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification as part of CPSWeek 2009 in San Francisco in April and the June 2009 FCSSI S5 in Dayton. The focus of the Workshop will be on disseminating the MCAR community consensus on multi-criticality challenges, as discussed here in Section 5.0, as well as further promoting a community based approach to addressing those challenges. The focus of the MCAR portion of S5 will be on presenting the results of the MCAR requirements efforts. Both events encourage non-MCAR team participation.

6.2. Broader Community Outreach

The MCAR program is only one step on the road to achieving the AFRL vision for the development of future mixed criticality systems. As stated in Section 1.0, an objective of AFRL, and of this white paper, is to inspire a community of funding agencies; researchers, academic and industrial; and air framers to collectively address the challenges that all face, and that none can solve alone. The previous section discussed how AFRL has begun the process of forming such a community through interactions with NSF and the NITRD CPS Community.

Although AFRL is approaching the mixed criticality challenge from the perspective of safe and secure flight software for next generation UAVs, many of the same challenges are faced by rotary winged and manned fixed winged aircraft. This has motivated AFRL to seek collaborators in the broader safety-critical avionics domain, including the Army, NASA, and NAVAIR.

Further consideration of the problem areas identified in Section 5.0 makes it clear that most of these are not unique to the aerospace domain, though there are domain specific aspects to each. As the challenges are not limited to AFRL's domain, neither are the solutions. Accordingly, AFRL is seeking to identify and exploit common ground with other domains. One thing that was clear from the November 2008 HCSS Workshop is that there is a great deal of

commonality across the various transportation sectors represented: aerospace, automotive, and rail. Likewise, many of the challenges facing the medical device and nuclear regulatory domains also apply to the MCAR domain. AFRL is seeking to collaborate with those organizations funding research in all of these areas to coordinate research efforts to jointly address the common aspects of these challenges.

MCAR itself represents a step in this collaboration, as AFRL and NSF explore how they can respond to their individual imperatives while still making progress toward the common research objective. Future extensions of that collaboration could include references to identified MCAR research challenges in future solicitations, and availability of AFRL described challenge problems for NSF sponsored researchers.

For this to succeed, the MCAR challenges must capture the attention of prospective researchers. One of the purposes of this document is to do just that. Not only define research areas rich enough for premier researchers to be drawn to, but to provide the mixed criticality context that both makes the challenges real, and enhances the transitionability of any future research results. This white paper and the spring workshops begin to attempt to do so.

NSF and AFRL by themselves still can not establish a community with the critical mass and scope to lead to MCAR success, though they represent a nucleus around which such a critical mass could coalesce. For success, additional participants are necessary to support a broader community, both research breadth (SBIRs, industrial research teams in addition to universities and not-for-profits) and depth (basic research, technology maturation, proof of concept demonstration, transition).

From an AFRL perspective, today, MCAR itself has focused on requirements definition and defining the 6.1 research space. Existing demonstration programs are expected to encounter mixed criticality challenges over the next 3 to 4 years yielding additional lessons and further refining requirements. New solutions developed and matured to 6.2 within 5 to 7 years will enable affordable development and certification of those systems. One of the challenges for the broader community is to reconcile the individual domain roadmaps, a process that is being undertaken as part of the NITRD sponsored CPS workshops. The current Aerospace and Automotive roadmaps were presented at the November 2008 HCSS National Workshop.

7.0 SUMMARY

AFRL's MCAR program is laying the foundation for a Design-for-Certification process that can serve as a basis of iterative specification, development, analysis and testing phases for cost-effective, certifiable mixed criticality architectures. MCAR is the start of the second phase of FCSSI, a broader effort to expand and improve the flight critical system software development and certification processes, which began in 2004. Three major air framers are contributing their experience to collaboratively define the newly emerging system level certifiability requirements for next generation mixed-criticality UAVs. These airframers, along with their subcontractors, are collectively defining and documenting requirements for certifiable mixed criticality embedded software architectures and their development environments. Fundamentally new and efficient methods and tools are needed for the specification, design, analysis, generation, test and ***certification*** of UAV software that hosts both safety-critical and mission-critical functions.

In military UAVs, the federated architecture model places safety critical and mission critical functions on physically separate boxes. Data exchanges among processes of different criticality flow over a monitored physical bus, ensuring that lower criticality processes can not inadvertently contaminate safety-critical control processes. This federated architecture model reduces certification complexity since all applications on a box are certified to the same level and explicit data flow monitoring interface functions are relatively understandable from a certification perspective.

Pressures for improved vehicle performance via reduced size, weight, and power coupled with increases in needed functionality, led to the integrated avionics architecture model, where functions with multiple criticalities coexist on a single processor. Logical, rather than physical, separation mechanisms such as time partitioning techniques were developed and architected in portions of the hardware, middleware, operating system, and application layers to enable co-hosting the multi-criticality systems, ensuring safety.

The next generation of UAVs increases avionics systems software complexity by adding higher level cognitive functions. This also introduces a fuzzier distinction between safety critical and mission critical functions, especially under off-nominal conditions. In manned aircraft the human pilot provided the ultimate decision point to de-conflict and prioritize incongruous inputs

from different criticality functions. In autonomous systems, the cyber-pilot, or equivalently, the avionics software must dynamically manage conflicting function inputs. Hosting of multi-criticality functions on many-core microprocessors, with undocumented internal architectures and resource sharing policies, further exacerbates implementation of separation principles. The status-quo of physical or logical process isolation is no longer adequate or feasible.

MCAR has identified several core technical challenges. Among the fundamental challenges is the specification of a new definition for criticality levels, since the separation of safety and mission functions becomes blurred in the absence of a pilot. With advanced dynamic reconfiguration capabilities, components might also have variable criticality defined by their context or use. Traditionally, safety criticality has had a one dimensional value. In future systems, safety criticality may be multi-dimensional corresponding to different system safety behaviors. Analysis capabilities to ensure that systems have been safely and securely composed in all possible use configurations are needed. These methods and tools must look at multi-faceted models to ensure that competing requirements, such as safety and security, or timeliness and capacity, are simultaneously satisfied. It is a significant challenge to host a predictable software system on hardware with undocumented behaviors. It is also a challenge to provide certification artifacts, acceptable for certification credit, when composable pre-certified components are assembled to build a subsystem. Finally, the challenge of coalescing the design, development, testing, and V&V processes and environments to the maximum extent possible, while maintaining required independence for certification is a task that awaits the growing community addressing these issues.

We cannot afford to solve the mixed-criticality certification problem alone. We cannot afford to solve it multiple times. And we cannot afford not to solve it. The MCAR program is actively cooperating and recruiting participation from NSF's CPS, CISE, and ENG directorates, from NITRD/HCSS CG, Army, NAVAIR, and other government organizations, from standardization efforts communities such as SAE's AS5506 (AADL), from academia and research communities at large, from SBIRs and from automotive and rail transportation sectors. Please join us in finding a solution to the mixed criticality architecture certification challenges that lie ahead.

8.0 REFERENCES

[1] Airlines Electronic Engineering Committee. Avionics application software standard interface part 1 - required services. Technical Report 653P1-2, Aeronautical Radio, Inc., March 2006.

[2] RTCS SC-167 and EUROCAEWG-12. Software Considerations in Airborne Systems and Equipment Certification. Number RTCS/DO-178B. RTCA, Inc., Washington, D.C., December 1992.

[3] International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 12207 Systems and Software Engineering – Software Life Cycle Processes, International Organization for Standardization, March 2008, 138 pages.

[4] System Safety Division Headquarters, Air Force Material Command. Department of defense standard practice for system safety. Technical Report MIL-STD-882D, Department of Defense, February 2000.

[5] Peter H. Feiler, David P. Gulch, and John J. Hudak. The architecture analysis & design language (AADL): An introduction. Technical Report CMU/SEI-2006-TN-011, Software Engineering Institute, February 2006.

[6] OMG systems modeling language (SysML), version 1.1. Technical Report formal/2008- 11-02, Object Management Group, November 2008.