Modulare Avionik als Grundlage für Systemdefinition, Systemkonfiguration und Systemkontrolle

B. Balser, M. Förster, G. Grabowski

EADS Deutschland GmbH
Postfach 801160
81663 München
burkhard.balser@m.eads.net
michael.foerster@m.eads.net

Abstract: ASAAC - Allied Standard Avionics Architecture Council - ist ein international ausgerichtetes Vorhaben, das die Standardisierung zukünftiger Avioniksysteme von Militärflugzeugen zum Ziel hat. Der Beitrag gibt zunächst einen kurzen Abriss über Ziele und Vorteile von ASAAC. Davon ausgehend wird das Softwareschichtenmodell für zukünftige, integrierte modulare Avionik-Architekturen dargestellt.

1 Modulare Avionik in der Militärischen Luftfahrt

Das Konzept der Integrierten Modularen Avionik (IMA) ist seit Ende der 80er/Anfang der 90er Jahre bereits Gegenstand einer Reihe von Veröffentlichungen geworden, die dessen mögliche Realisierungen und weitreichenden Auswirkungen zum Thema haben (z.b. [NW]). Daher soll an dieser Stelle nur dessen Grundprinzip umrissen werden.

1.1 Konventionelle Systemarchitekturen

Die Architekturen konventioneller Avioniksysteme sind gekennzeichnet durch nahezu unabhängige Teilsysteme, die bestimmte Funktionen bzw. Funktionsgruppen wahrnehmen wie z.b. die Flugsteuerung. Diese Teilsysteme (Sensoren, Aktuatoren, Systemcontroller und Anzeigen) bestehen aus Gerätesätzen, die als sogenannte Line Replaceable Units (LRUs) ausgeführt sind. Die LRUs sind über analoge und digitale Datenübertragungselemente miteinander verbunden. In [NW95] werden die wesentlichen Komponenten eines Systemcontrollers im folgenden zusammengefasst:

1. Hardware:
   - Eingangs-/Ausgangsinterface mit Datenaufbereitung
   - Steuerungsteil, welches die funktionsspezifische Software enthält
   - Stromversorgung
   - Selbsttest (Built-In Test Equipment, BITE)

2. Software:
   - Anteile, die den Betrieb des Controllers ermöglichen
   - Selbsttest (Built-In Test, BIT) mit Schnittstelle zum Wartungssystem
   - Funktionsspezifischer Anteil, der die Steuerung der Systemfunktion wahrnimmt

In den letzten 20 Jahren sind jedoch die ursprünglich eher lose miteinander verbundenen oder z.t. völlig unabhängigen Teilsysteme zu immer komplexeren Gesamtsystemen vernetzt worden. Demgegenüber ist, von wenigen Ausnahmen abgesehen – wie z.b. Displays aus Kathodenstrahlröhren –, eine querschnittliche Nutzung der mehrfach vorhandenen Baugruppen oder Komponenten nicht gegeben. Die Avionik des modernen Fighters markiert aus europäischer Sicht einen Höhepunkt dieser Entwicklung (Bild 1).

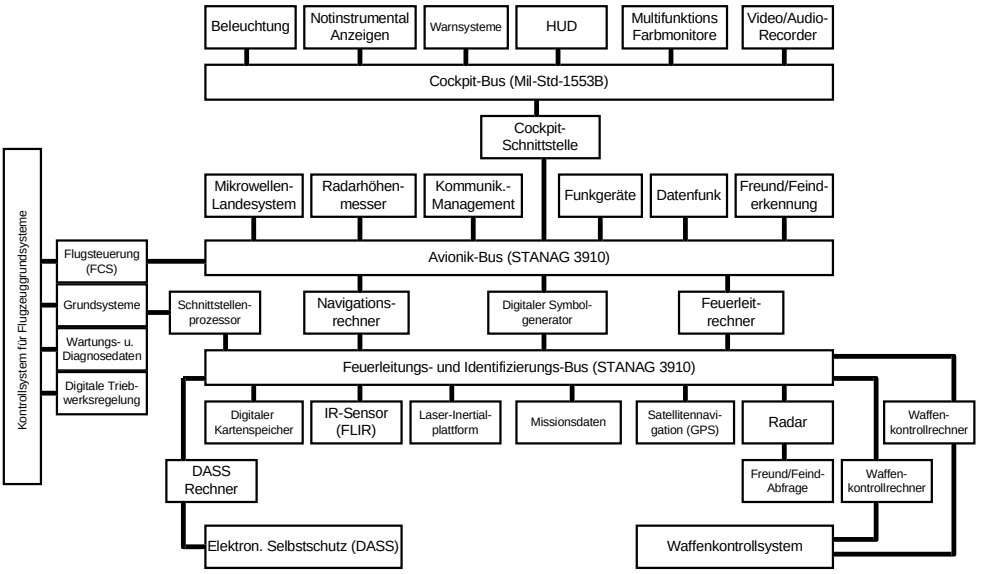

Bild 1: Avioniksystemarchitektur eines modernen Fighters

Die zunehmende Vernetzung der einzelnen Teilsysteme und die damit gestiegene Komplexität des Gesamtsystems führten zu stark steigenden Aufwänden für Test- und Integration sowie Logistik, die sich in hohen Kosten für Entwicklung, Beschaffung und Nutzung von Avioniksystemen niederschlagen.

Diese Kriterien treffen besonders auch auf Systemerweiterungen und -anpassungen zu, die während der Lebensdauer von Avioniksystemen üblicherweise vorgenommen werden und den Automatisierungsgrad durch eine Vielzahl missionsspezifischer Funktionen erhöhen sollen.

Wegen der enormen finanziellen Aufwände sind speziell in Europa nationale Eigenentwicklungen nicht mehr möglich. Moderne Flugzeuge und Hubschrauber werden in inter-

nationalen Konsortien entwickelt, die nicht nur nach technisch-konzeptionellen, sondern auch nach wirtschaftlichen und politischen Kriterien besetzt sind. Diese beeinflussen nicht nur die Realisierung von Avioniksystemen, sondern bereits deren Konzeptionierung bzw. Entwicklungsprozesse: Nationale Finanzierungsanteile müssen möglichst unter Berücksichtigung vorhandener Ressourcen in nationale Arbeitsanteile umgewandelt werden. Im Ergebnis dessen entstehen nicht selten künstliche Schnittstellen, die einem klaren Systementwurf entgegenstehen.

1.2 Modulare Systemarchitekturen

1.2.1 Künftige Herausforderungen

Sowohl für militärische als auch für zivile Avioniksysteme wird, bedingt durch wachsende operationelle Anforderungen, ein Funktionszuwachs vorhergesehen. Als Beispiele seien an dieser Stelle Sensorfusionsfunktionen für die Navigation militärischer Luftfahrzeuge oder Flight Management-Funktionen im zivilen Bereich erwähnt. Es kann damit erwartet werden, dass bedingt durch die unterschiedlichen Anforderungen und verfügbaren Ressourcen verschiedener Betreiber einerseits und durch die lange Lebensdauer insbesondere von militärischen Luftfahrzeugen andererseits die Funktionalität nicht mit „einem Mal", d.h. bei der Indienstellung, sondern schrittweise - nach je 7-10 Jahre - im Rahmen von Kampfwertsteigerungs- und anderen Upgrade-Maßnahmen in die Architektur eingebracht wird. Die gegenwärtigen, insbesondere wegen der Zertifizierung äußerst aufwendigen Verfahren müssen deutlich vereinfacht werden.

Luftfahrzeugbetreiber fordern bereits seit langem eine verbesserte Wartbarkeit. In diesem Zusammenhang wird häufig von geplanter Wartung gesprochen (Scheduled Maintenance). Modulare Architekturen lassen durch ein verbessertes Fehlertoleranzverhalten Verbesserungen erwarten, die im wesentlichen durch verbesserte Fehlerlokalisierung und -behandlung erreicht werden sollen. Querschnittlich nutzbare Komponenten gestatten die Erhaltung der Systemfunktionen als Ganzes trotz eventuell vorzunehmender Rekonfigurationen innerhalb der Systemarchitektur.

Zudem soll es mit Hilfe modularer Architekturen besser als in der Vergangenheit gelingen, die sich in immer kürzeren zeitlichen Abständen vollziehenden Technologiefortschritte im IT-Sektor für Avioniksysteme nutzbar zu machen.

1.2.2 Grundprinzip Integrierter Modularer Avionik

Die Prinzipien der Integrierten Modularen Avionik bieten geeignete Ansätze, um diesen Herausforderungen zu begegnen. Im Mittelpunkt dieses Konzepts stehen leistungsfähige Racks als Datenverarbeitungszentren. Diese enthalten querschnittlich verwendbare Hardware-Bausteine in Form von Modulen, die auf wenige Standardbautypen begrenzt sind, deren Qualität durch die größere Stückzahl im Vergleich zu den konventionellen, für einen Spezialfall gebauten Modulen verbessert ist. Die mechanisch und elektrisch abgeschlossenen Module sind als Line Replaceable Modules (LRMs) ausgeführt, die die LRUs als Basisbaugruppe ablösen sollen. Die LRMs und Racks untereinander sind durch Hochgeschwindigkeitsdatenbusse und/oder durch geschaltete Netzwerke hoher Bandbreite miteinander verbunden. Auch die Softwarearchitektur folgt einem modularen

Ansatz und trennt zwischen den speziellen systemspezifischen Anteilen und den Basisanteilen wie z.b. dem Betriebssystem eines LRMs.

Die Module der Hardware und Software werden zu einem Kern zusammengefasst und durch spezielle Module, z.b. Graphikkarte, Massenspeicher, spezielle Input/Output Einheiten, an die vorgesehene Funktionalität angepasst. Die querschnittliche Nutzung der modularen Komponenten erlaubt Software weitestgehend beliebig zu verteilen oder umzuverteilen entsprechend den Systemanforderungen, ohne durch Komponentengrenzen zu behindert zu werden, und erlaubt so eine Rekonfiguration des Kernes zu einem weiterhin funktionsfähigen System beim Ausfall einzelner Module. Damit kann die geforderte hohe Verfügbarkeit erreicht werden und die Anzahl außerplanmäßiger Wartungsereignisse während des Betriebs können erheblich reduziert werden.

Das Zusammenwirken der modularen Komponenten in einem integrierten System erfolgt über standardisierende Schnittstellen für die Kommunikation auf der Systemebene. Weiterhin ist für modulare Avionik ein standardisiertes Design für Größen und Einbauvorrichtungen notwendig, um die gewünschte querschnittliche Nutzung von modularen Komponenten innerhalb einer Flotte von Flugzeugen bzw. für mehrere Flugzeugtypen zu ermöglichen. Diese Architektur-Standards sind ein wesentlicher Aspekt, um als Systemlieferant bzw. -integrator Module (Hardware und/oder Software) von verschiedenen Herstellern beziehen zu können[1].

1.2.3 Realisierung Modularer Avionik

Die Auswirkungen des modularen Avionikkonzeptes auf die Rollenteilung zwischen Betreibern, Systemintegrator, Subsystem- und Plattformlieferanten sowie Zertifizierungsbehörden können noch nicht endgültig vorhergesagt werden. Es gilt als sicher, dass der gesamte Systementwicklungsprozess stark modifiziert wird. Allein der durch die stark wachsende Anzahl zu integrierender Komponenten gestiegene Komplexitätsgrad erfordert eine verstärkte Nutzung von Softwarewerkzeugen. Eine weitere wichtige Frage, die mit den Zertifizierungsbehörden geklärt werden muss, betrifft die Nutzung von bereits „vorqualifizierten" Komponenten [NW95].

Zur Realisierung des Konzeptes der Integrierten Modularen Avionik wurden zahlreiche internationale Aktivitäten ins Leben gerufen. Für zivile modulare Avionik wurde Anfang der 90er Jahre ein Architekturmodell erarbeitet, welches 1991 als Standard [AR91] verabschiedet wurde. Für militärische Anwendungen wurden in den USA bereits erste modulare Avioniksysteme realisiert [GF96], [HM94]. In Europa werden derzeit nach einer Reihe von Konzeptstudien zu Beginn der 90er Jahre zumeist auf nationaler Ebene Experimentalsysteme und Prototypen realisiert. In den letzten Jahren fanden jedoch eine Reihe von multinationalen Aktivitäten und Programmen statt (z.b. [AS03], [CO96], [EU95]), um die verschiedenen nationalen Ansätze zur modularen Avionik zu harmonisieren. In diesem Zusammenhang sind auch die Aktivitäten des Allied Standards Avionics Architecture Council (ASAAC) einzuordnen. In dessen Phase I (1990-1994) wurde

[1] Systeme, die als LRUs ausgeliefert werden, haben bereits heute intern einen modularen Aufbau. Dieser ist jedoch - von wenigen externen Schnittstellen abgesehen - stark herstellerspezifisch und nach außen nicht sichtbar.

ein Architekturmodell für militärische modulare Avionik entwickelt [AR91],[AS94]. Innerhalb der Phase II, die im November 1997 anlief, sollen Standards für europäische modulare Avionik erarbeitet und mit Hilfe von Demonstratoren demonstriert werden.

Der Geschäftsbereich Militärflugzeuge der European Aeronautic Defence and Space Company (EADS M) nimmt zusammen mit nationalen und internationalen Partnern aus Frankreich und Großbritannien an diesem Programm teil.

2 Das ASAAC-Technologieprogramm

2.1 Überblick über das Programm

Die Phase II des ASAAC Technologieprogramms zielt auf die Erarbeitung und Validierung von Architektur-Standards für europäische modulare Avionik bis 2004. Mit Hilfe von Demonstratoren soll - sowohl für die Auftraggeber als auch für die Industrie - der Nachweis erbracht werden, dass modulare Avioniksysteme, die den ASAAC-Standards entsprechen werden, die erforderliche gestiegene Leistung bei signifikanten Kostenreduzierungen erbringen. Insbesondere sind die folgenden drei Punkte die Treiber des Programms:

- Die Kosten über die Lebensdauer (Life Cycle Costs)
- Die Erfüllung der Anforderungen während des Betriebs des Flugzeugs (Operational Performance)
- Die Erfüllung der beauftragten Mission (Mission Performance)

Die erste Stufe (Stage 1) der Phase II (Laufzeit bis Anfang 1999) diente der Verfeinerung des in ASAAC Phase I entwickelten Architekturmodells mit Blick auf mögliche Realisierbarkeit. Dazu wurden Konzepte für die Systemarchitektur als Ganzes sowie für die speziellen Aspekte Software, Standardmodultypen, Netzwerk und die Unterbringung der Standardmodule in Racks erarbeitet. Am Ende der Stufe 1 bildeten diese Konzepte den ersten Entwurf für die Architektur-Standards, welche mit Hilfe von Demonstrationen validiert werden sollen. Daher wurden parallel zur Konzepterarbeitung Demonstratoren und dazugehörige Szenarien entwickelt.

Die Stufe 2 der Phase II (ASAAC Phase II Stage 2, Laufzeit 48 Monate, Beginn 12.1999) dient der Umsetzung der in Stufe 1 erarbeiteten Konzepte, d.h. sie zu demonstrieren. Während der Demonstrationen wird der Nachweis erbracht, dass die ASAAC-Standards eingehalten werden können oder die Entwürfe der Standards werden ggf. anhand der Demonstrationsergebnisse entsprechend überarbeitet. Die endgültigen Entwürfe der Standards sollen zum Abschluss der Stufe 2 der Phase II, d.h. nach Ablauf der Demonstrationen validiert und den zuständigen Normungsgremien zur Annahme übergeben werden (ab 2003).

2.2 Das ASAAC-Architekturmodell für modulare Avionik

Das im Rahmen von ASAAC entwickelte Architekturmodell für modulare Avionik orientiert sich bei den Softwareaspekten am Modell des Virtuellen Rechners und ist schichtenförmig wie folgt gegliedert (Bild 2):

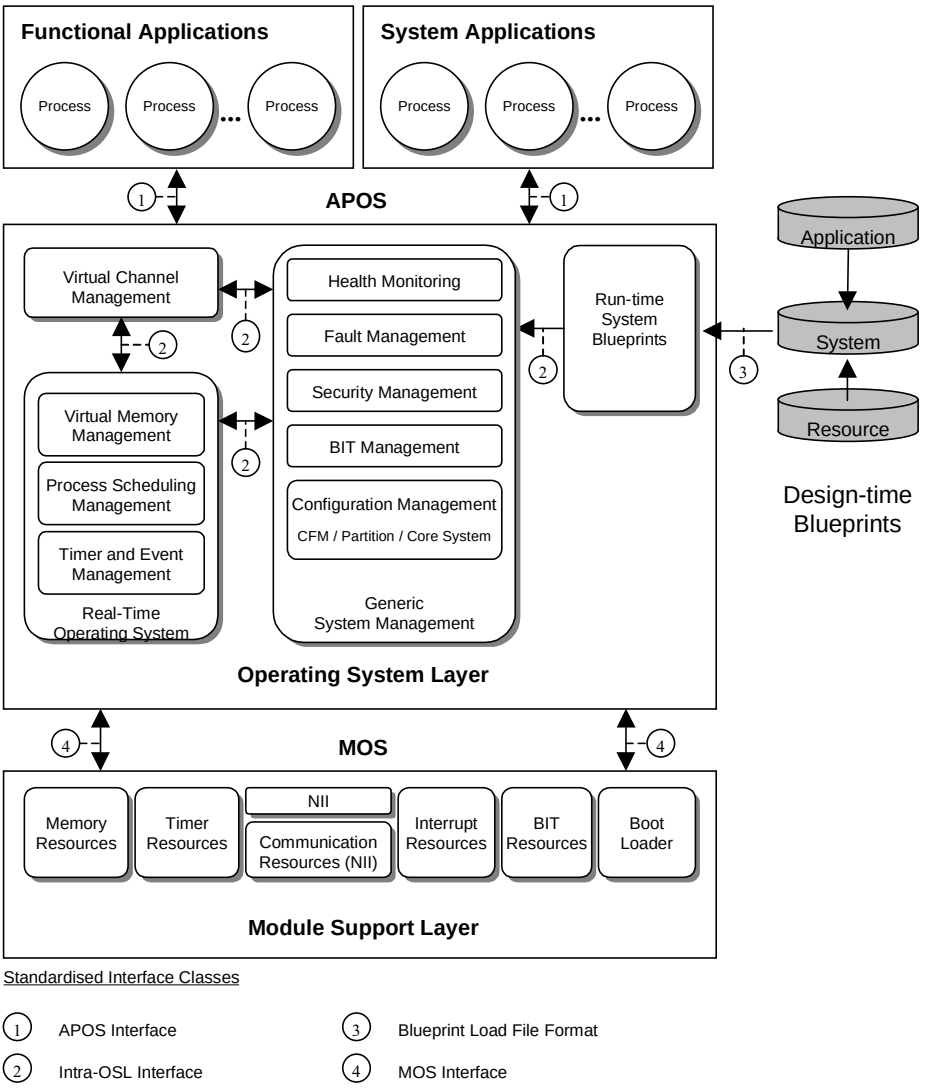

Bild 2: ASAAC Modell der Softwarearchitektur (Stand Juni 1998)

- Auf der obersten Ebene sind die Applikationen angeordnet. Hier wird unterschieden zwischen den Funktionalen Applikationen, d.h. Applikationen, die die „eigentlichen" Avionikfunktionen erbringen und speziellen Systemapplikationen, die Steuerungs- und Verwaltungsfunktionen auf der Systemebene erfüllen. Die Kombination von Funktionalen und Systemapplikationen ist für ein spezielles System bzw. ein Luftfahrzeug spezifisch. Unterhalb der Applikationen sind die Module bzw. Komponenten der Architektur angesiedelt, die querschnittlich, d.h. auch unabhängig von einem bestimmten Flugzeugprojekt, genutzt werden sollen:

- Eine Betriebssystemschicht (Operating System Layer, OSL), die aus einem Realzeitbetriebssystem sowie aus Erweiterungen für Kommunikations- und allgemeine Systemverwaltungsfunktionen (Virtual Channel Management, VCM bzw. Generic System Management (GSM) besteht. Die GSM-Funktionen wirken auf der Ebene eines Moduls bzw. Racks. Der Zugriff der Applikationen auf die Dienste der Betriebssystemschicht erfolgt über eine genormte Aufrufschnittstelle (Application/Operating System, APOS).

- Anpassungsschicht an die Hardware (Module Support Layer, MSL). Dies enthält den Zugriff auf die modulspezifische Hardware als auch den Zugriff auf das Netzwerk. Die Betriebssystemschicht greift ebenfalls über eine genormte Aufrufschnittstelle auf die Hardware zu: über die MOS-Schnittstelle (Module/Operating System) auf die Modulhardware[1] und über das Network Independent Interface (NII) auf das Netzwerk.

Ein wesentliches Mittel der Systemintegration stellen die Systemkonfigurationstabellen dar, die in ASAAC als Blueprints bezeichnet werden (Bild 2). Blueprints enthalten in einer formalen Darstellung Attribute, die die Systemelemente der Applikationen (Software), der Hardware (Resources) sowie das integrierte System beschreiben. Während der Systementwicklung werden Blueprints benutzt, um die Anforderungen der Software in Application Blueprints zu erfassen und diese auf die Kapazitäten der Hardware abzubilden, die ihrerseits in Resource Blueprints erfasst werden. Somit soll für jeden möglichen Betriebszustand eine Konfiguration gefunden werden. Mit Hilfe der Blueprints soll dieser Mapping-Vorgang zumindest teilautomatisiert werden, wofür spezielle Softwarewerkzeuge entwickelt werden müssen. Die Menge der zur Laufzeit zulässigen Systemkonfigurationen sowie die notwendigen Übergangssequenzen zwischen ihnen werden in den System Blueprints abgelegt und in einem geeigneten Format an das Laufzeitsystem übergeben. Dessen Anteile der Systemsteuerungs- und -verwaltungssoftware werden durch die System Blueprints in die Lage versetzt, das System entsprechend den Vorgaben zu konfigurieren bzw. - wenn erforderlich - zu rekonfigurieren. Letzteres kann z.B. als Reaktion auf einen Wechsel der Mission bzw. einer Missionsphase oder wegen des Auftretens eines Fehlers innerhalb eines Moduls erforderlich werden.

---

[1] In der ASAAC-Systemarchitektur sind folgende Modultypen vorgesehen: Data Processing Module (DPM), Signal Processing Module (SPM), Graphics Processing Module (GPM), Network Switch Module (NSM), Mass Memory Module (MMM)

2.3 Elemente der Standardisierung

Wie im vorigen bereits gesagt, ist das Ziel des Programms, Standards zu erstellen (siehe [AS03]), die die Erstellung von Anwendersoftware erleichtern sollen und die Portabilität auf andere Plattformen ermöglichen sollen. Ein erfolgreiches Beispiel für einen Standard für Software Anwendungen ist POSIX (IEC99). Da ASAAC neben der Applikationsentwicklung als einen wesentlichen Punkt auch die Systemsteuerung und die Interprozesskommunikation (IPC) im Blicke hat, sind noch weitere Interfaces nötig, die im folgenden kurz zusammengefasst seien:

- APOS ist die Schnittstelle für den Anwendungsprogrammierer, ihre Funktionalität ist an POSIX orientiert und umfasst die Elemente Threadmanagement, Timemanagement, Synchronisation, Fehlerbehandlung und Kommunikation über Virtuelle Kanäle (VC). Sonderfunktionen sind für einige IMA Module vorgesehen: Filehandling für das Speichermodul (MMM, Mass Memory Modul), Funktionen zum Ein- und Ausschalten von Modulen in einem Rack für das Power Conversion Module (PCM) und OpenGL als Graphikinterface für das Graphikmodul (GPM, Graphic Processing Module).

- SMOS abstrahiert die Elemente der Systemsteuerung (GSM) vom Operating System. Es enthält die Funktionen, die es ermöglichen, Prozesse zu starten, zu steuern und zu beenden, die Kommunikationsverbindungen zwischen den Prozessen aufzubauen, zu kontrollieren und zu beenden, und der Hard- und Software zu beobachten: Fehlererkennung bei Hard- und Software, Built In Tests (BIT) und Monitorfunktionen. Diese Funktionen sind nicht für normale Anwendungen vorgesehen, sie werden nur im GSM genutzt, um dessen Portabilität zu gewährleisten.

- Die Blueprints dienen der Spezifikation, dem Design, der Konfiguration und Kontrolle des Systems.

    - Designtime Blueprints spezifizieren das System und legen die Funktionalität und Verbindungen und Beziehungen der einzelnen Softwareelemente fest. Durch sie kann der Nachweis erbracht werden, dass das gewünschte System machbar und richtig ist.

    - Aus den Designtime Blueprints werden die Runtime Blueprints errechnet. Runtime Blueprints sind die Elemente der Systemsteuerung während der Laufzeit des Systems. Sie enthalten Informationen über die Elemente einer Konfiguration, d.h. der Prozesse und ihrer Kommunikationsverbindungen, und über mögliche Fehler, deren Erkennung und Behandlung, d.h. notwendige Systemübergänge und Aktionen, die beim Auftreten eines Fehlers zu vollziehen sind. Ihr Standard wird in einer formalen Sprache beschrieben. Den einzelnen Projekten ist es überlassen eine geeignete Implementierung zu definieren.

    - Das SMBP ist das Interface zwischen der Implementierung der Runtime Blueprints und der Nutzung im GSM.

- Neben diesen Softwareschnittstellen werden auch noch logische Schnittstellen benötigt. Ein Beispiel: Ein Modul fällt aus, es wird von einem GSM bemerkt. Diese Information muss an andere Module und ihre GSM's verteilt werden um eine neue Konfiguration einzunehmen, sprich eine Neuverteilung der Prozesse und Kommunikationswege, damit das System weiterhin seine Aufgabe erfüllen kann. In den Blueprints ist der Fehler beschrieben und die Mittel zu seiner Entdeckung, ebenso die Aktionen, die durchzuführen sind, es werden weiterhin aber noch die Formate der Messages benötigt, die zwischen den einzelnen Systemkomponenten ausgetauscht werden, seien es triviale Elemente wie die Header oder komplexe wie die Übermittlung des Fehlers, seiner Ursache und seiner Behebung. Ein weiteres Eingehen auf diesen Teil sprengt den Rahmen dieses Vortrages.

Ziel der Bemühungen ist es, die einzelnen Teile des Systems, Hardware, Betriebssystem, Systemkontrolle (GSM) und Anwendung von einander zu trennen und ihre Wiederverwendbarkeit in anderen Systemen zu ermöglichen.

2.4 Realisierung und Demonstration der Elemente

Die Standardisierung im Softwarebereich, auf die anderen Bereiche kann im Rahmen dieses Vortrages nicht eingegangen werden, umfasst die Interfaces, nicht jedoch die Implementierung ihrer Funktionen und ihre Nutzer. Die Nachweisführung über Demonstrationen benötigt gerade diese Elemente. Es sind:

- die Implementierung notwendigen Funktionen der Interfaces,
- das OSL, bestehend aus RTOS, GSM, VCM und den notwendigen Systemanpassungen
- die Demonstrationsprogramme und ihre Blueprints

Die Demonstrationen werden in drei Stufen durchgeführt:

- Stufe 1 die Implementierung der Interfacefunktionen, der Kommunikation und eines rudimentären GSM für eine VMEbus Plattform. Die Interprozesskommunikation ist hier auf das Board beschränkt. Dies gilt auch für das GSM.
- Im zweiten Schritt wird die IPC über die Boardgrenzen hinweg ermöglicht. Ebenso erhalten die GSM die Funktionalität, Meldungen auszutauschen. Parallel werden Prototypen der IMA Module gefertigt.
- Auf den Prototypen wird im dritten Schritt dann die Funktionalität portiert und die Funktionsfähigkeit des ASAAC Systems gezeigt.

2.5 Zusammenspiel der Elemente

Bisher wurde immer wieder das OSL und das GSM erwähnt als Komponenten der Ver-

mittelung zwischen Applikation und dem System. Im folgenden sei etwas auf ihre Arbeit eingegangen.

Das OSL teilt sich in die Komponenten RTOS, GSM,VCM und die Anpassungsschicht für die Interfaces, über die oben berichtet wurde.

2.5.1 Real-Time Operating System (RTOS)

ASAAC verlangt den Aufbau der Applikationen in Form von POSIX complient Prozessen, der Ablauf wird durch Threads übernommen, die sich über APOS Funktionen synchronisieren und Nachrichten austauschen. Die Prozesse benötigen einen Adressraum, der disjunkt zu anderen Prozessen ist, d.h. Virtual Memory Management. Diese Aufgaben übernimmt das RTOS, das in zwei Formen realisiert wird:

- Einmal unter Verwendung von COTS Technologie, die Wahl fiel auf das Realtime Unix LynxOS von Lynux Works, zum anderen

- implementieren die Firmen aus Großbritannien ein eigenes Operating System.

2.5.2 Generic System Management (GSM)

Das Wort „generisch“ bedeutet, dass es weder vom zugrundeliegenden Operating System noch von den Anwenderprozessen abhängig sein soll. Ziel ist, dass das GSM sowohl auf allen Hardwarebausteinen einsetzbar ist, als auch, dass verschiedene Implementierungen zusammenarbeiten können. Seine, bereits oben erwähnten Aufgaben teilen sich mehrere Elemente:

- Der Health Monitor (HM) ist zuständig für die Überwachung der Hard- und Software. Er nimmt die Fehlermeldungen auf, unternimmt, wenn nötig, erste Erhaltungsmaßnahmen und übermittelt dies

- dem Fault Manager (FM), der weitere Maßnahmen durchführt und dann die Information weiterreicht sowohl an andere Instanzen des GSM im System als auch zu

- dem lokalen Configuration Manager (CM), der dann den lokalen Umbau übernimmt.

Alle diese Handlungen werden durch die Informationen aus den Runtime Blueprints gesteuert, die der Blueprint Manager dem anderen Einheiten des GSM zur Verfügung stellt.

2.5.3 Virtual Channel Management

Wie oben ausgeführt wird die Interprozesskommunikation über

- Virtuelle Kanäle (VC) durchgeführt. Unter Verwendung von LynxOS sind sie in Device Drivern implementiert, eine Fähigkeit von UNIX und UNIX ähnlichen Systemen.

- Aufbau, Konfiguration, Routing und Abbau der VC steuert der Virtual Channel Manager (VCM). Er ist die Vermittlerstelle zwischen dem System unabhängigen GSM und der systemabhängigen Implementierung der VC. Seine Wiederverwendung ist von der Implementierung der Prozeduren der SMOS, dem eingesetzten Operating System und der Implementierung der VC abhängig.

2.5.4 Einsatz der Blueprints bei Konfiguration und Kontrolle eines Systems

In den vorherigen Abschnitten wurden die Blueprints als Elemente der Systemkonfiguration und -kontrolle genannt. Im folgenden sei ihr Aufbau und ihre Anwendung in den einzelnen Phasen eines Projektes näher beschrieben.

Während der Entwicklungsphase eines Systems dienen die Blueprints zum einen als Sammlung softwarespezifischer Elemente, z.b. die Interfaces der zu entwickelnden Prozeduren, ihres Zeitbedarfes, ihrer Kommunikation etc., zum anderen stellen sie diese Informationen für bereits entwickelte Elemente dem Entwickler zur Verfügung. Sie werden „Application Blueprints“ genannt. Die verwendete Hardware wird in ihren Eigenschaften, z.b. Prozessor, Memory, Bussysteme ebenfalls beschrieben. Ihr Name ist „Resource Blueprints“. Die in diesem Arbeitsabschnitt entwickelten oder genutzten Blueprints werden unter dem Begriff „Design Time Blueprints“ zusammengefasst.

Aus den für das Projekt relevanten „Design Time Blueprints“ wird das Ablaufverhalten des Systems errechnet, ein Prozess, der gegenwärtig untersucht wird. Das Ergebnis sind Blueprints, die zur Laufzeit das Systemverhalten bestimmen, die „Runtime Blueprints“.

Wie oben erwähnt, dient eine formale Sprache zur Beschreibung der „Runtime Blueprints“. Zum einen beschreiben sie Konfigurationen, d.h. die Elemente, die auf einer Hardwareplatform benötigt werden, die Prozesse, ihre Elemente, die Threads, ihren Resourcenbedarf in Form von Speicher und Zeit und die Kommunikation zwischen den Prozessen lokal auf dem Operating System oder systemübergreifend, das Routing, die Nutzung von Interfaces etc. Zum anderen beschreiben sie Ereignisse, die Übergänge zwischen Konfigurationen erzwingen, und die Aktionen, die dabei durchgeführt werden sollen.

Ein Beispiel soll dies verdeutlichen: Ein Prozess erzeugt einen nicht intern behandelten Fehler, in einem UNIX System äußert sich dies durch ein Signal mit einem entsprechenden Errorcode. Dies wird durch den HM abgefangen. Aus den Blueprints wurden ihm Informationen übermittelt, auf welche Signale er zu reagieren hat und welche Aktionen er durchzuführen hat, in dem Fall z.b. den Prozess anzuhalten und den FM zu informieren, der dann ebenfalls Aktionen gemäss den Blueprints durchzuführen hat.

Der Vorteil eines solchen Ablaufes ist zum einen, dass das Design eine Beschreibung von Fehlern im Ablauf eines Programms auf eine systemweite Basis stellt, zum anderen die Fehlerbehandlung zur Laufzeit vorbestimmt ist und im System abgestimmt durchgeführt wird, so dass ein kontrollierter Übergang von einer Konfiguration zur Nächsten das Gesamtverhalten so wenig wie möglich beeinflusst.

3 Zusammenfassung

Avioniksysteme mit modularen Architekturen bieten vielversprechende Potentiale, um auch in der militärischen Luftfahrt künftigen operationelle und technologische Erfordernisse begegnen zu können. Der Erfolg, d.h. die erwarteten Kostenreduzierungen, wird jedoch wesentlich davon abhängen, ob es gelingt, tragfähige und langlebige Architekturstandards für die europäische Luftfahrtindustrie zu etablieren, um in diesen engen Markt eine von Wettbewerb geprägte Struktur zu etablieren. Diese sollte sich jedoch nicht erst mit der Entwicklung neuer Flugzeugprojekte herausbilden: wegen der in den letzten Jahren verschlechterten Situation bei der Verfügbarkeit von elektronischen Komponenten, die speziell militärischen Erfordernissen genügen, und der extrem langen Lebensdauer militärischer Luftfahrzeuge müssen Wege gefunden werden, um das Potential modularer Architekturen auch für Upgrade-Maßnahmen existierender Luftfahrzeuge auszuschöpfen.

Zudem müssen geeignete Zulassungskonzepte gefunden werden, um Systemmodifikationen mit Hilfe von querschnittlich verwendbaren bzw. wiederverwendbaren Komponenten vornehmen zu können.

Literaturverzeichnis

[AR91] ARINC Report 651,Design Guidance for Integrated Modular Avionics, Nov. 1991

[AS94] ASAAC Phase I: ASAAC Feasibility Study: Core Architecture Concept Definition, ASAAC: 02, Issue 4, Feb. 1994

[AS02] ASAAC Phase II: First draft of proposed standards for software, ASAAC2-STA-31131-001-SWWG-I01 Software Concept Refinement Report, ASAAC2-TYP-WSBN-SNo-ORIG-Snn

[CO96] Carbonell, J., Ostgaard: Impact of COTS on Military Avionics Architectures AGARD CP-581/29, Oct 1996 (published 1997)

[EU93] EUCLID RTP 4.1: Modular Avionics Harmonization Study, 1995

[GF94] Grasshoff, M.; Förster, M.: An Approach Towards Integration of a Modular Core Avionics System Kernel, AGARD CP-581/30, Oct 1996 (published 1997)

[HM94] Howard/Madden/Hall: The F-22 Avionics Computing System Architecture, 1994 IEEE National Aerospace Electronics Systems Conference, Tutorials, IEEE 1994

[NW95] D. Niemeyer, R. Wolckenhauer: Integrierte Modulare Avionik – Eine neue Generation der zivilen Flugzeugsystemarchitektur DGLR-JT95-040, DGLR Jahrestagung 1995