IST-004527 ARTIST2
Network of Excellence
on Embedded Systems Design

Activity Progress Report for Year 3

JPRA-NoE Integration / JPASE Dissemination and Industrial Liaison

ARTIST2 Workshop on Integrated Modular Avionics, PARADES, Rome, November 12-13, 2007

Cluster **Real-Time Components**

Activity Leader: **Albert Benveniste (INRIA)**
http://www.irisa.fr/distribcom/benveniste/

Today, the exponentially increasing diversity of airborne systems results in an ever increasing number of computers and controllers for system management, monitoring, and control. The development of specific ad-hoc solutions causes increases in costs, which in turn impacts purchase prices and operational costs. To overcome this, standardization principles and reuse of function units are now considered, via Integrated Modular Avionics. Integrated Modular Avionics (IMA) has set the principles of standardized components and interfaces of hardware and software in aircraft. These principles have been applied for the first time in the development of the Airbus A380. Further developing IMA raises a number of issues that require fundamental research efforts, in tight coordination with engineering needs.

ARTIST2, the European Network of Excellence on embedded systems has organized, as part of its activity on "scientific challenges in specific industrial sectors", a two-day workshop dedicated to Systems, Software, and Architecture, aspects of IMA.

The workshop analyzed:

- the issues and difficulties encountered by aircraft manufacturers and their suppliers,
- the specific research problems that result from the above issues, and,
- the recent advances in research that may contribute to overcoming the above difficulties.

Jean-Bernard Itier, Airbus: the AIRBUS approach to open IMA; Technology, Methods, Process and Future needs

JB Itier is Strategic technical Project Manager for Information Management for Avionics Platform and responsible for all R&T platforms subjects participating to IMA.

ARTIST2 – IMA & ADCN

Presented by

Jean-Bernard ITIER
STP Manager IMAP

A380 Integrated Modular Avionics

The history, objectives and challenges of the deployment of IMA on A380

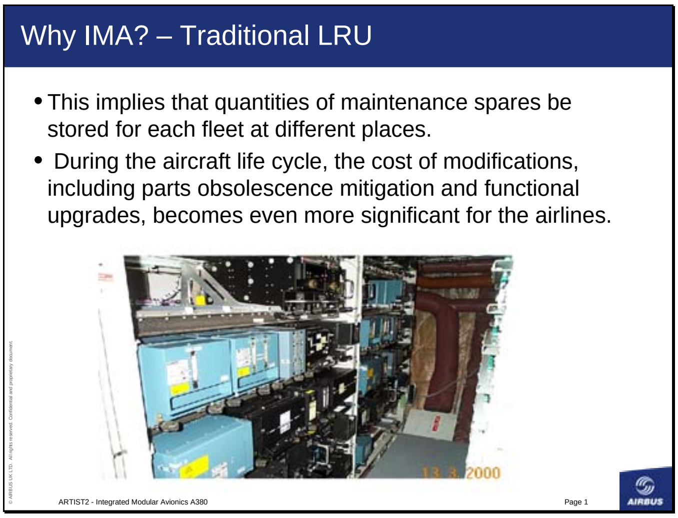
ARTIST2 - Integrated Modular Avionics A380

IMA and its background

Since A300 there is an increasing number for software controller systems, for performance, safety, improved maintenance, passenger comfort. This will increase.

Consequences so far:

- every system = 1 or more computer / controllers
- every aircraft = new computers
- every computer = lots of specific equipment



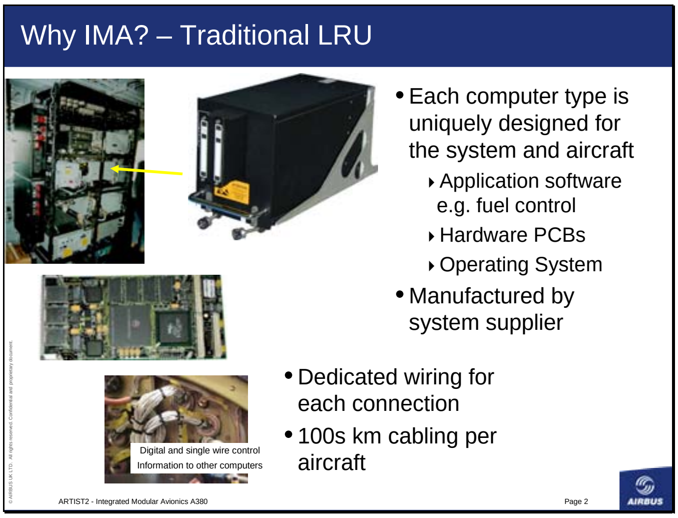
Why IMA? – Traditional LRU

- This implies that quantities of maintenance spares be stored for each fleet at different places.
- During the aircraft life cycle, the cost of modifications, including parts obsolescence mitigation and functional upgrades, becomes even more significant for the airlines.

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 1

If you look at the communications on past aircraft it is in most cases point to point even where digital data buses were used they were in the main based on ARINC 429 a standard where every sender of information has its own bus and in most cases because of data needs several buses



Why IMA? – Traditional LRU

- Each computer type is uniquely designed for the system and aircraft
  - Application software e.g. fuel control
  - Hardware PCBs
  - Operating System
- Manufactured by system supplier

- Dedicated wiring for each connection
- 100s km cabling per aircraft

Digital and single wire control
Information to other computers

ARTIST2 - Integrated Modular Avionics A380 Page 2

In the past we have one or more computers per aircraft system,

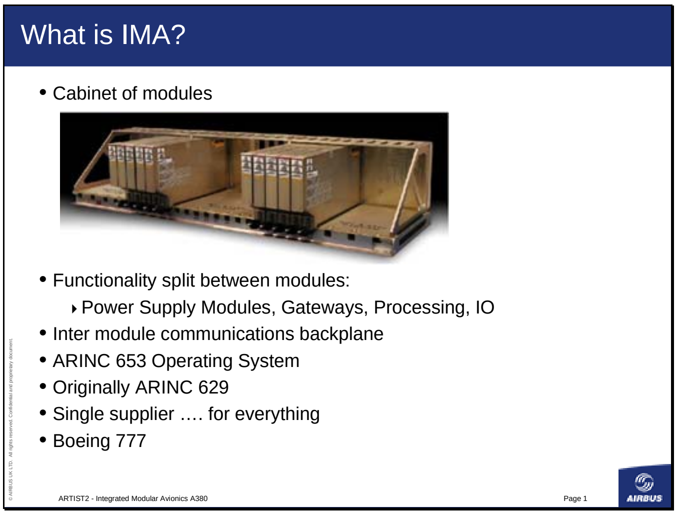
typically uniquely designed and manufactured for that system on that aircraft e.g. for fuel management I.e. you could not hold a fuel computer that could be used on an A320 and A330.

Since A300 the trend was to increase on board SW & HW. However this situation cannot continuous & it was necessary to stabilize volume, weigh & costs for on board electronics and to reduce it in the next generation of A/C.

The response: IMA

- IMA
  - concept
  - no specific technology or components
- Integration
  - multiple applications on same computer
  - data communications with multiplexed network

Several approaches were proposed by the different suppliers:

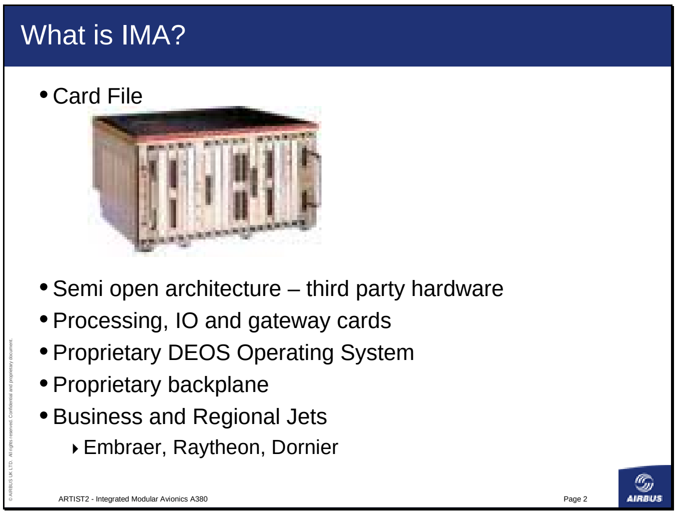


What is IMA?

- Cabinet of modules
- Functionality split between modules:
  - Power Supply Modules, Gateways, Processing, IO
- Inter module communications backplane
- ARINC 653 Operating System
- Originally ARINC 629
- Single supplier …. for everything
- Boeing 777

ARTIST2 - Integrated Modular Avionics A380 Page 1

AIRBUS



What is IMA?

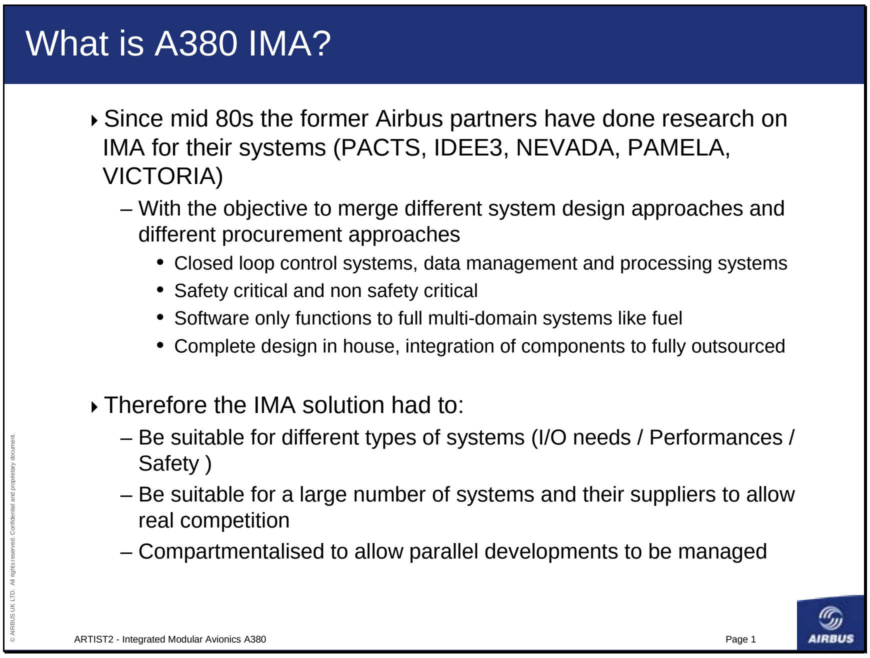
- Card File
- Semi open architecture – third party hardware
- Processing, IO and gateway cards
- Proprietary DEOS Operating System
- Proprietary backplane
- Business and Regional Jets
  - Embraer, Raytheon, Dornier

ARTIST2 - Integrated Modular Avionics A380 Page 2

AIRBUS

- Independent modules as LRU, provides processing, IO and PSU in one package

What is A380 IMA?

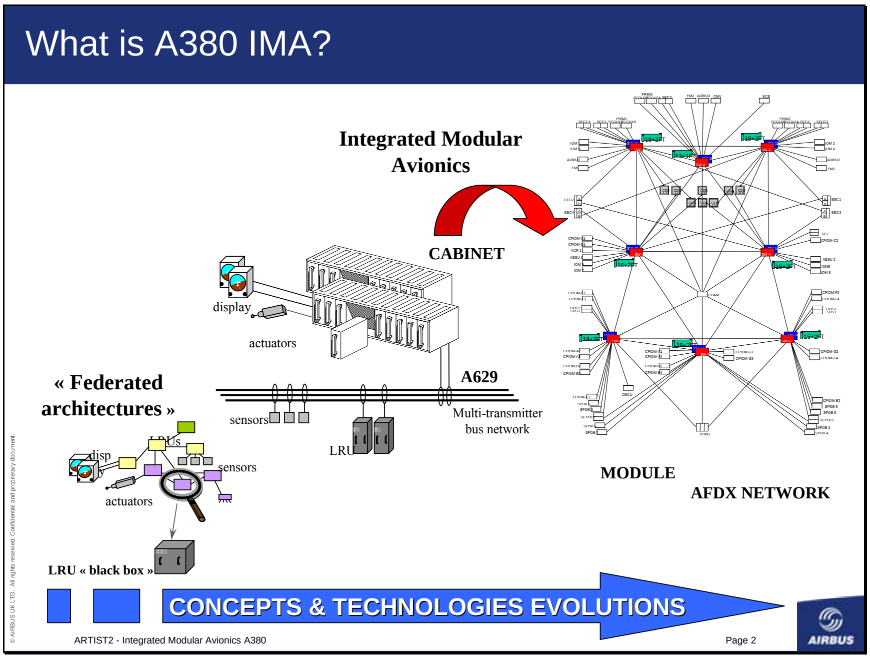


What is A380 IMA?

- Since mid 80s the former Airbus partners have done research on IMA for their systems (PACTS, IDEE3, NEVADA, PAMELA, VICTORIA)
  - With the objective to merge different system design approaches and different procurement approaches
    - Closed loop control systems, data management and processing systems
    - Safety critical and non safety critical
    - Software only functions to full multi-domain systems like fuel
    - Complete design in house, integration of components to fully outsourced
- Therefore the IMA solution had to:
  - Be suitable for different types of systems (I/O needs / Performances / Safety)
  - Be suitable for a large number of systems and their suppliers to allow real competition
  - Compartmentalised to allow parallel developments to be managed

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 1

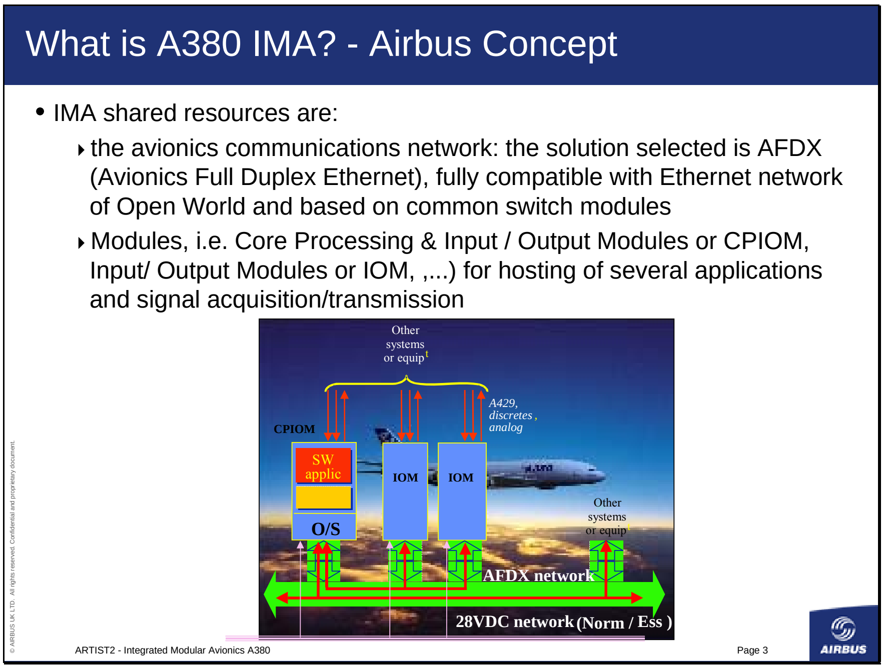


What is A380 IMA?

CONCEPTS & TECHNOLOGIES EVOLUTIONS

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

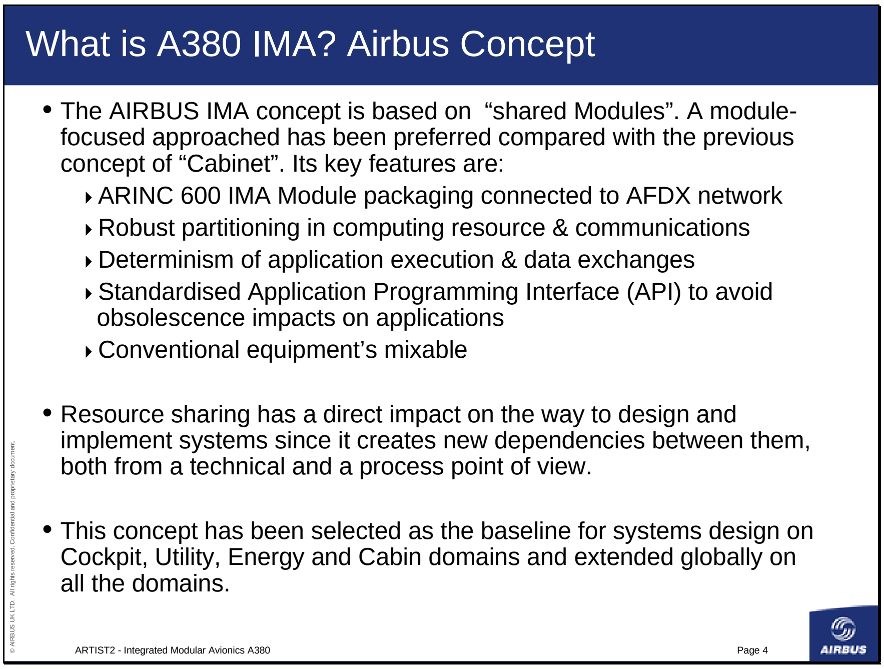
ARTIST2 - Integrated Modular Avionics A380 Page 2



What is A380 IMA? - Airbus Concept

- IMA shared resources are:
  - the avionics communications network: the solution selected is AFDX (Avionics Full Duplex Ethernet), fully compatible with Ethernet network of Open World and based on common switch modules
  - Modules, i.e. Core Processing & Input / Output Modules or CPIOM, Input/ Output Modules or IOM, ,...) for hosting of several applications and signal acquisition/transmission

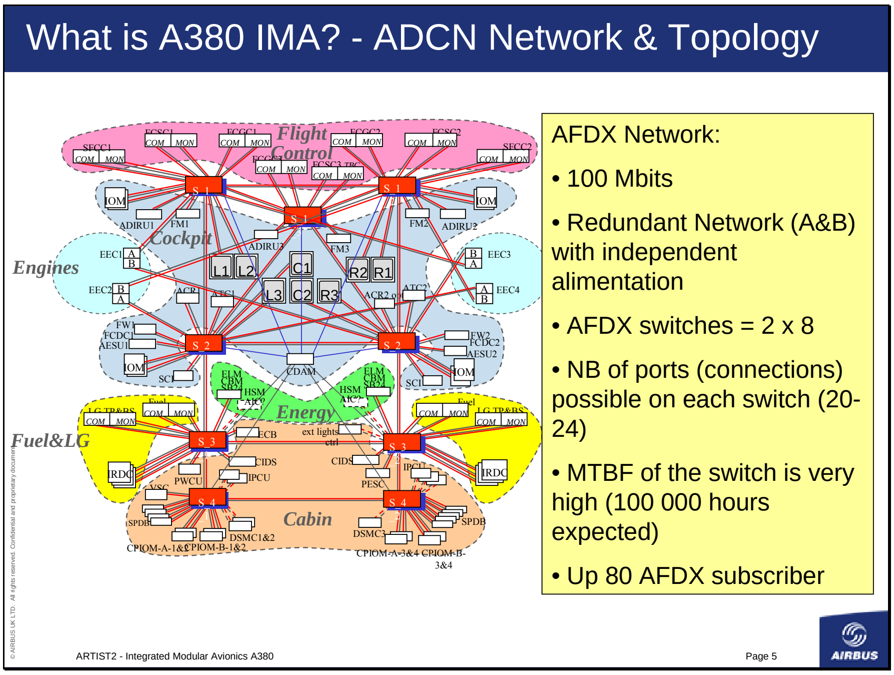
ARTIST2 - Integrated Modular Avionics A380 Page 3



What is A380 IMA? Airbus Concept

- The AIRBUS IMA concept is based on "shared Modules". A module-focused approached has been preferred compared with the previous concept of "Cabinet". Its key features are:
  - ARINC 600 IMA Module packaging connected to AFDX network
  - Robust partitioning in computing resource & communications
  - Determinism of application execution & data exchanges
  - Standardised Application Programming Interface (API) to avoid obsolescence impacts on applications
  - Conventional equipment's mixable
- Resource sharing has a direct impact on the way to design and implement systems since it creates new dependencies between them, both from a technical and a process point of view.
- This concept has been selected as the baseline for systems design on Cockpit, Utility, Energy and Cabin domains and extended globally on all the domains.

ARTIST2 - Integrated Modular Avionics A380 Page 4



What is A380 IMA? - ADCN Network & Topology

AFDX Network:

- 100 Mbits
- Redundant Network (A&B) with independent alimentation
- AFDX switches = 2 x 8
- NB of ports (connections) possible on each switch (20-24)
- MTBF of the switch is very high (100 000 hours expected)
- Up 80 AFDX subscriber

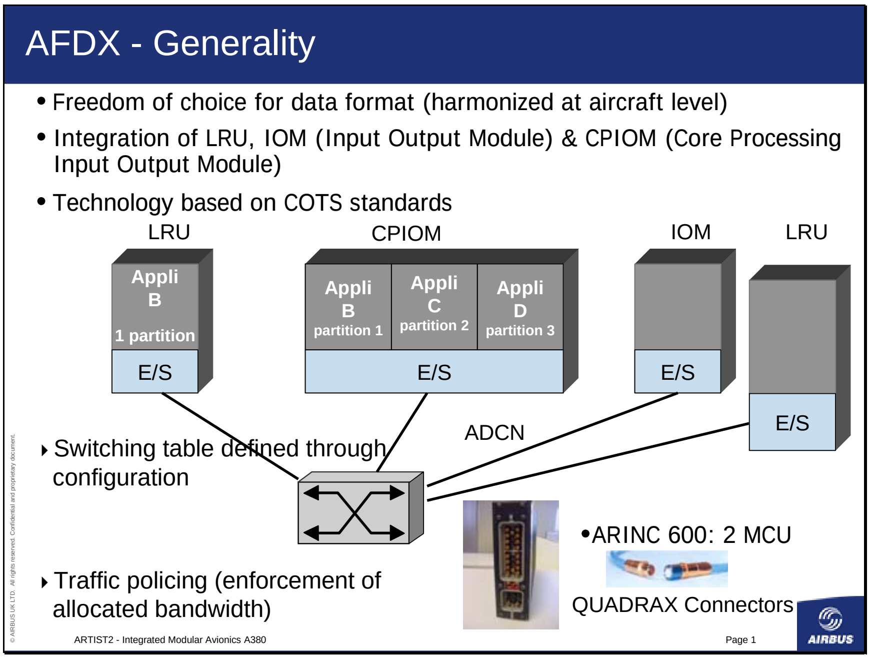
© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 5

AIRBUS

CPIOM + IOM + network + switches, AFDX (Avionics Full Duplex Ethernet), fully compatible with Ethernet; CPIOM allow for uniform handling of all applications; no power supply modules, part of networks or CPIOM → scalability.

AFDX



AFDX - Generality

- Freedom of choice for data format (harmonized at aircraft level)
- Integration of LRU, IOM (Input Output Module) & CPIOM (Core Processing Input Output Module)
- Technology based on COTS standards

- Switching table defined through configuration
- Traffic policing (enforcement of allocated bandwidth)

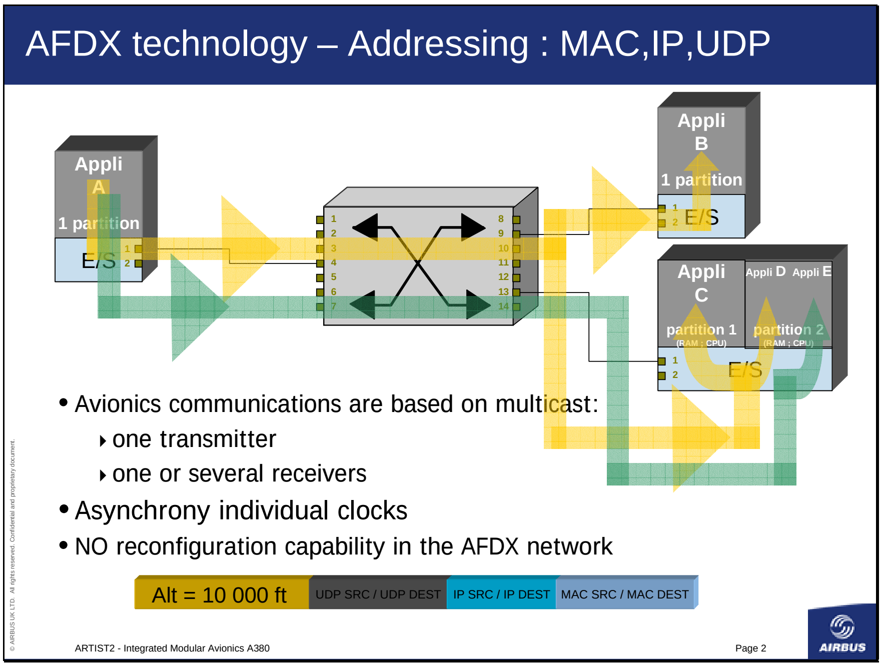
•ARINC 600: 2 MCU

QUADRAX Connectors

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 1

AIRBUS



AFDX technology – Addressing : MAC,IP,UDP

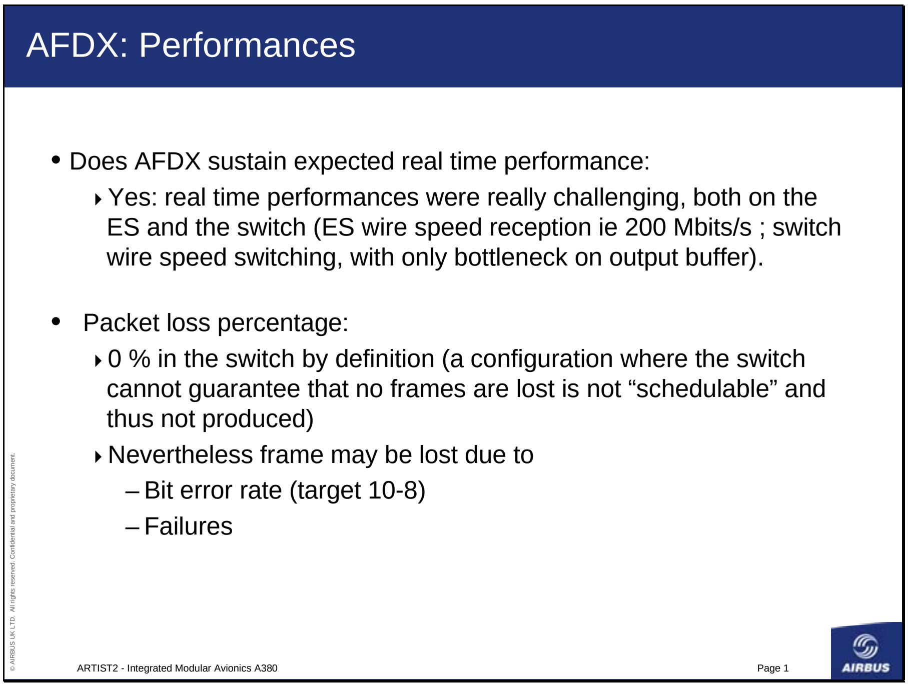
- Avionics communications are based on multicast:
  - one transmitter
  - one or several receivers
- Asynchrony individual clocks
- NO reconfiguration capability in the AFDX network

Alt = 10 000 ft	UDP SRC / UDP DEST	IP SRC / IP DEST	MAC SRC / MAC DEST

ARTIST2 - Integrated Modular Avionics A380 Page 2

AIRBUS

The aim is to enforce determinism; no reconfiguration capability in ADFX network; multicast; asynchrony of individual clocks for the different sub-system. Systems are essentially asynchronous at entire aircraft level.

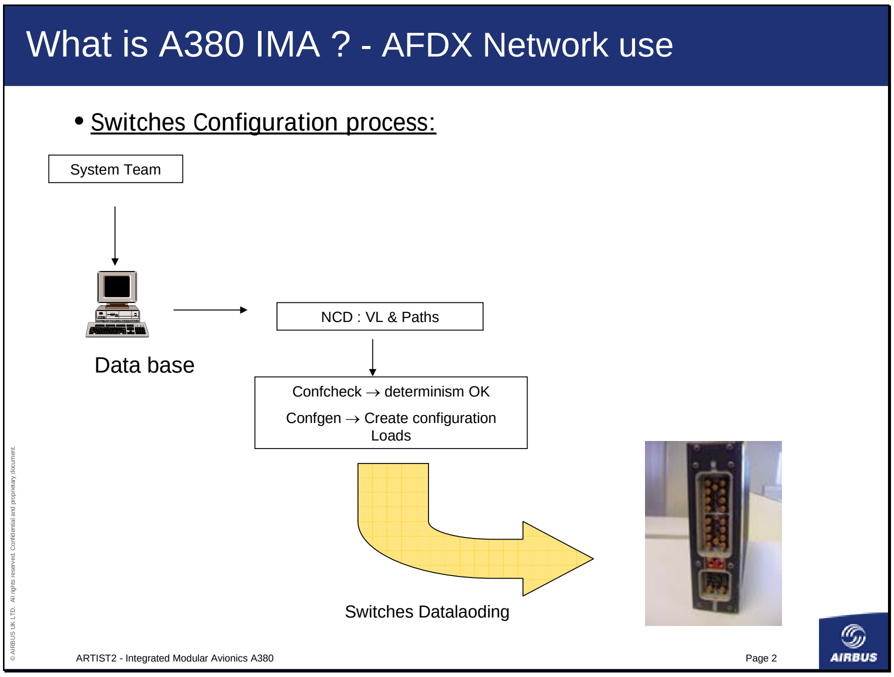


AFDX: Performances

- Does AFDX sustain expected real time performance:
  - Yes: real time performances were really challenging, both on the ES and the switch (ES wire speed reception ie 200 Mbits/s ; switch wire speed switching, with only bottleneck on output buffer).
- Packet loss percentage:
  - 0 % in the switch by definition (a configuration where the switch cannot guarantee that no frames are lost is not “schedulable” and thus not produced)
  - Nevertheless frame may be lost due to
    - Bit error rate (target 10-8)
    - Failures

ARTIST2 - Integrated Modular Avionics A380 Page 1

AIRBUS



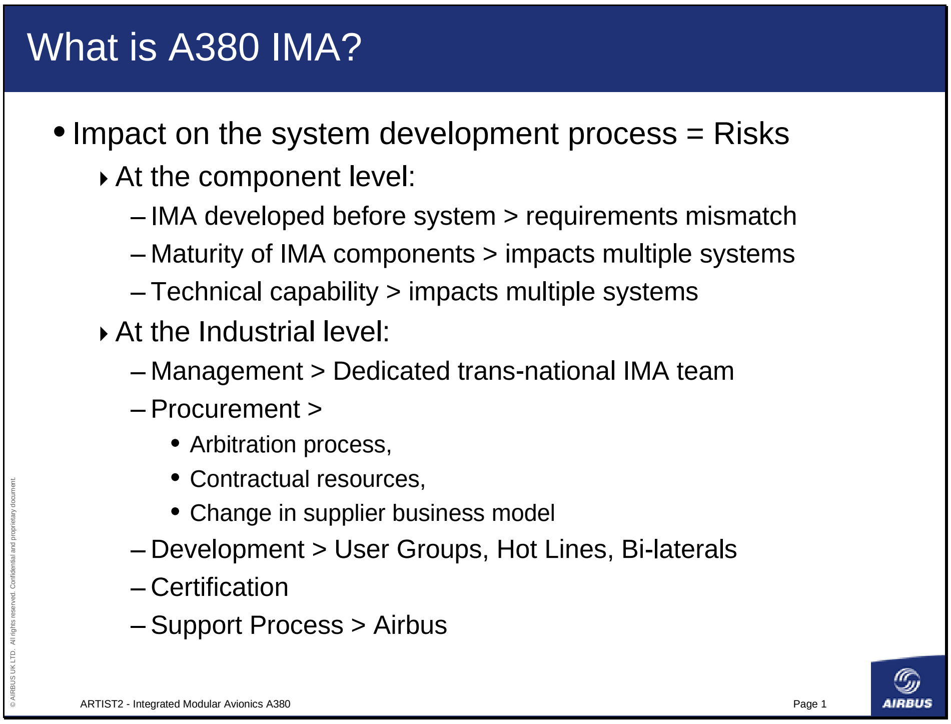
What is A380 IMA ? - AFDX Network use

- Switches Configuration process:

System Team
Data base
NCD : VL & Paths
Confcheck → determinism OK
Confgen → Create configuration Loads
Switches Datalaoding

Switch configuration process: the important point is that this configuration is performed at design time; flexibility supports multiple configurations for the aircraft.

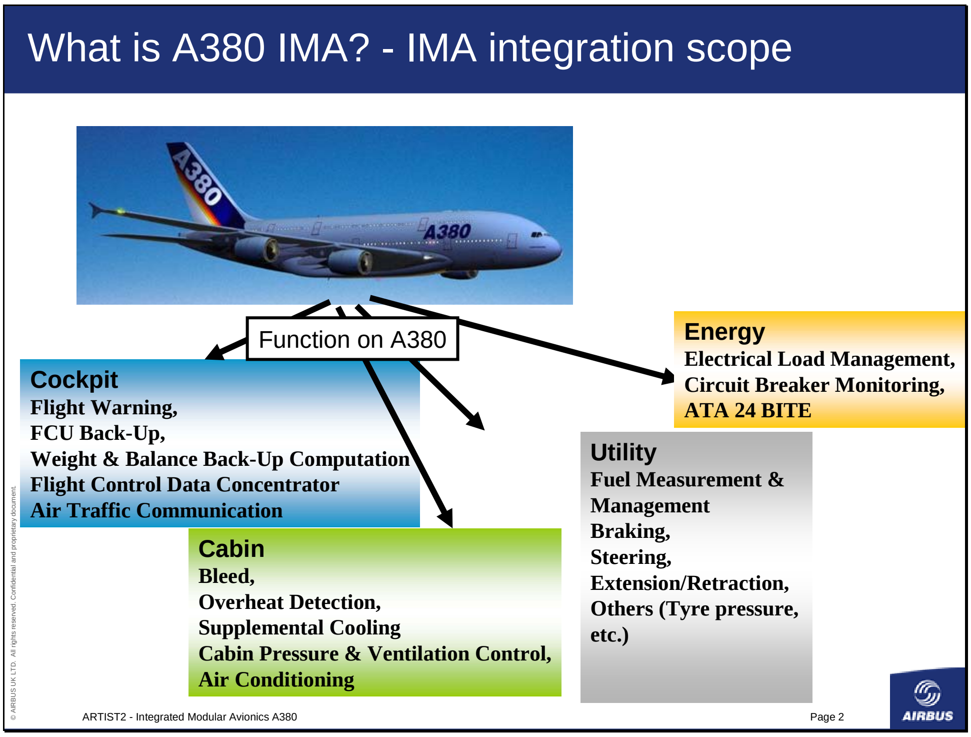
Impact of IMA



What is A380 IMA?

- Impact on the system development process = Risks
  - At the component level:
    - IMA developed before system > requirements mismatch
    - Maturity of IMA components > impacts multiple systems
    - Technical capability > impacts multiple systems
  - At the Industrial level:
    - Management > Dedicated trans-national IMA team
    - Procurement >
      - Arbitration process,
      - Contractual resources,
      - Change in supplier business model
    - Development > User Groups, Hot Lines, Bi-laterals
    - Certification
    - Support Process > Airbus

Altogether big change in the supplier business model; some suppliers provide only software; the way to pay them is different. New qualification approach was necessary to maximize IMA concept benefit & reuse HW qualification (Incremental qualification approach). A review of the many solutions proposed by the suppliers is performed before choosing the subsystems and components. Joint launch team to define the avionics solution.



What is A380 IMA? - IMA integration scope

Function on A380

Cockpit
Flight Warning,
FCU Back-Up,
Weight & Balance Back-Up Computation
Flight Control Data Concentrator
Air Traffic Communication

Cabin
Bleed,
Overheat Detection,
Supplemental Cooling
Cabin Pressure & Ventilation Control,
Air Conditioning

Energy
Electrical Load Management,
Circuit Breaker Monitoring,
ATA 24 BITE

Utility
Fuel Measurement & Management
Braking,
Steering,
Extension/Retraction,
Others (Tyre pressure, etc.)

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 2

Some systems are not in, mainly for performance issues; also it was the 1st time and the number of systems based on this technology had to be minimized.

Modules have been developed at:

- Airbus (EYY)
- Thales
- Diehl

ARINC 643 OS and API

Suppliers had to re-develop their applications in this way. Push independence between HW and SW; application SW independent from HW:

- No direct access to I/O
- Internal process control services
- Health Monitoring services

Multi-threading is used but (probably) not preemption. Partitions support segmentation; multi-threading is used within the partition. Scheduling between partitions is static.

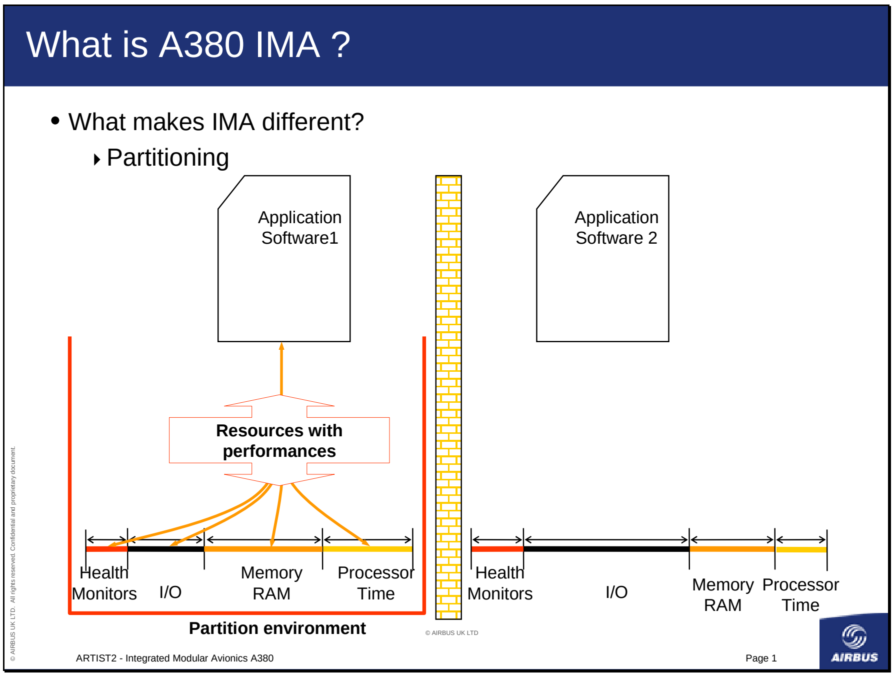
This allows fighting obsolescence of HW components. Sometimes these components will be upgraded. SW will be kept even if HW is upgraded.

Paritioning

Functions can be allocated to one or more partitions.

Each system must be able to be validated in isolation on the module. Faults must be contained. The performance of each system must be ***unaffected*** by any other

Q: is “unaffected” really possible? Do you mean that it continue working, but may affect indirectly other functions through the controlled plant.



What is A380 IMA ?

- What makes IMA different?
  - Partitioning

Application Software1

Application Software 2

Resources with performances

Health Monitors

I/O

Memory RAM

Processor Time

Partition environment

Health Monitors

I/O

Memory RAM

Processor Time

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

© AIRBUS UK LTD

ARTIST2 - Integrated Modular Avionics A380

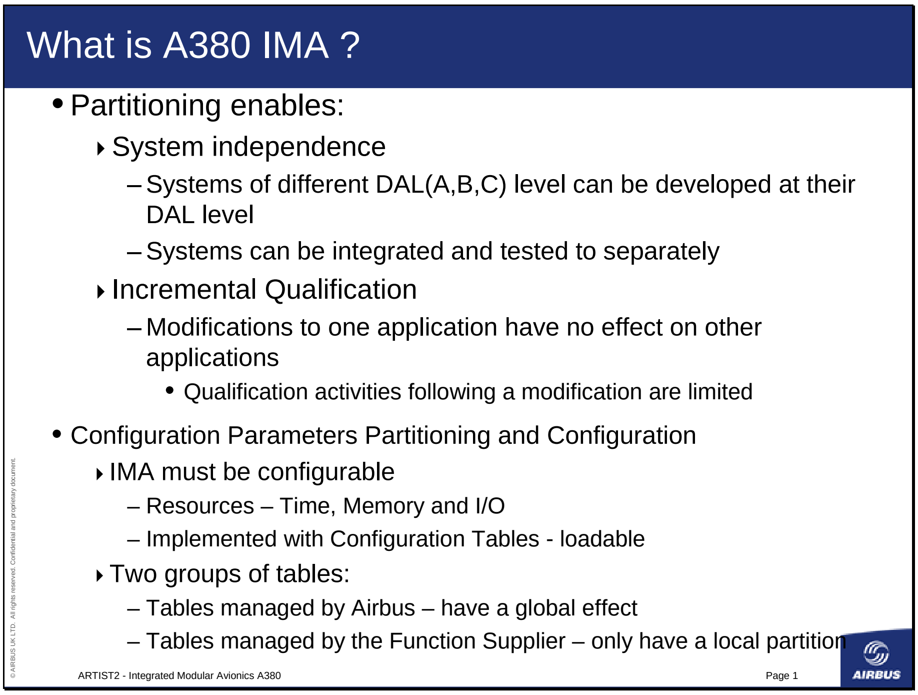
Page 1

AIRBUS

Regarding timing: time slots are allocated to partitions. Memory is protected via MMU to ensure that it remains safe.

Segregation of I/O; the different modules read the current value of exchanged data at their own pace. All exchanges are through the network using the virtual links.

What partition enables is shown below; incremental qualification is important and is enabled by partitioning.

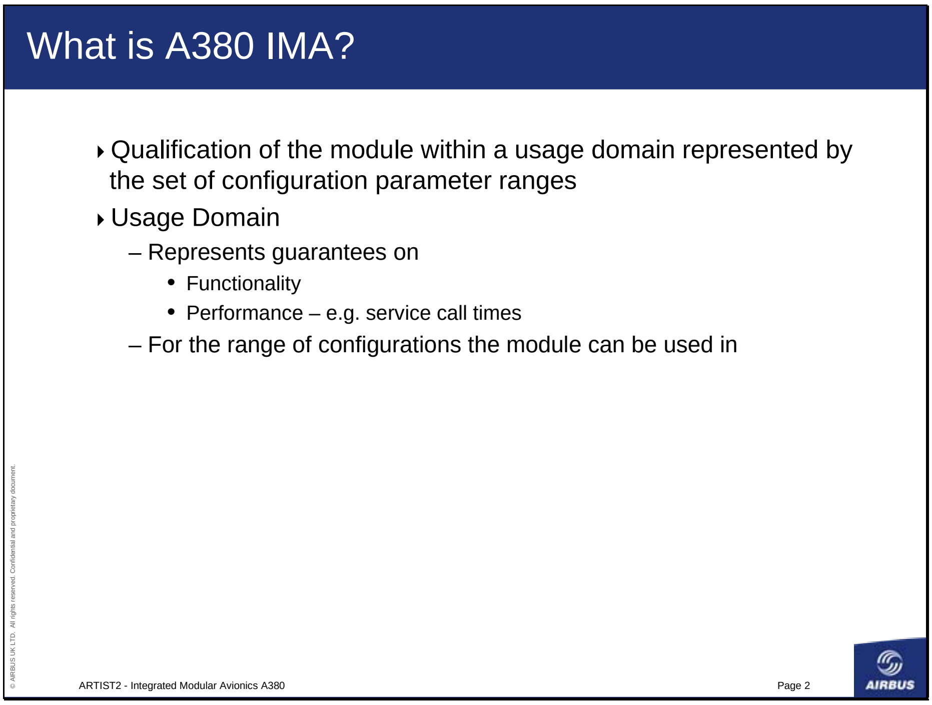


What is A380 IMA ?

- Partitioning enables:
  - System independence
    - Systems of different DAL(A,B,C) level can be developed at their DAL level
    - Systems can be integrated and tested to separately
  - Incremental Qualification
    - Modifications to one application have no effect on other applications
      - Qualification activities following a modification are limited
- Configuration Parameters Partitioning and Configuration
  - IMA must be configurable
    - Resources – Time, Memory and I/O
    - Implemented with Configuration Tables - loadable
  - Two groups of tables:
    - Tables managed by Airbus – have a global effect
    - Tables managed by the Function Supplier – only have a local partition

ARTIST2 - Integrated Modular Avionics A380 Page 1

AIRBUS

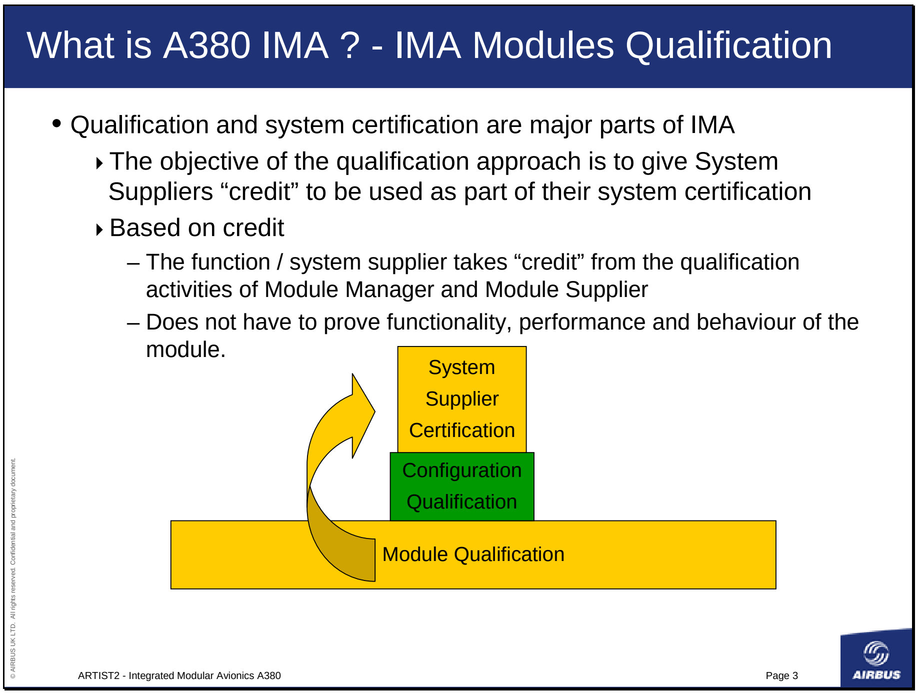


What is A380 IMA?

- Qualification of the module within a usage domain represented by the set of configuration parameter ranges
- Usage Domain
  - Represents guarantees on
    - Functionality
    - Performance – e.g. service call times
  - For the range of configurations the module can be used in

ARTIST2 - Integrated Modular Avionics A380 Page 2

AIRBUS



What is A380 IMA ? - IMA Modules Qualification

- Qualification and system certification are major parts of IMA
  - The objective of the qualification approach is to give System Suppliers "credit" to be used as part of their system certification
  - Based on credit
    - The function / system supplier takes "credit" from the qualification activities of Module Manager and Module Supplier
    - Does not have to prove functionality, performance and behaviour of the module.

Q: How to maintain determinism while not using inter-partition communication (which is guaranteed deterministic)? ***A: in fact different subsystems are designed from the beginning to tolerate asynchronous communications, thus nondeterminism in communications is no additional harm.***

Room of improvement to reduce the cost of testing

IMA induces new roles

IMA manager, played by Airbus; includes providing configuration tables (with their qualification), controls the use of modules resources, performs confidence testing and qualification of module configuration.

Integration

Integration tests are performed on HBOSS (Host Based OS Simulation) and then on TBOSS (Target Based OS Simulation).

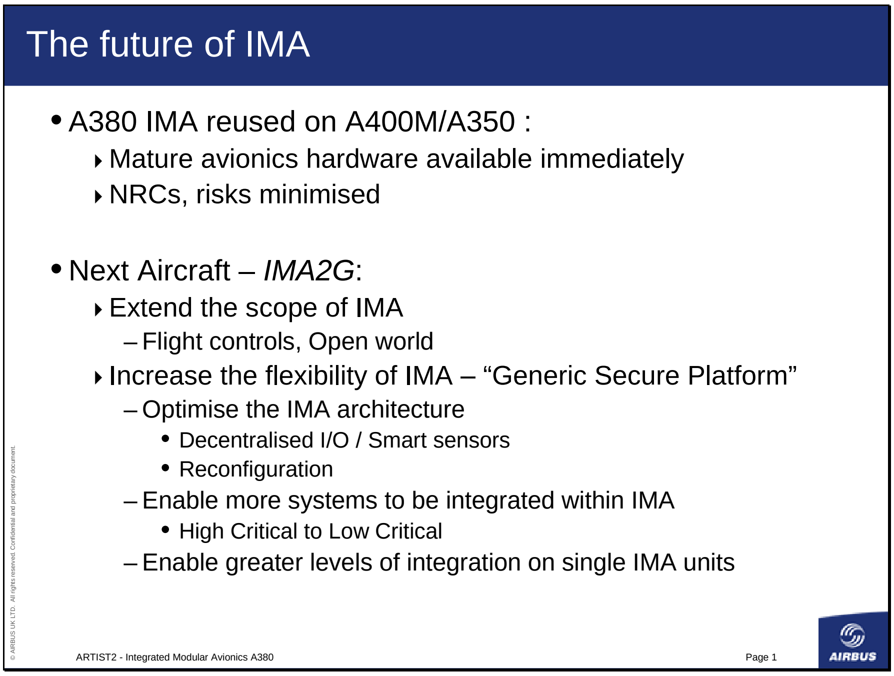
The future of IMA

See figure below; it is considered very beneficial since allows reusing mature avionics; risks minimised.

There is a need to extend the scope of this technology, so a new generation is planed, including flight control, also functions related to open world (non safety critical, e.g., infotainment). We wish to enable more systems to be integrated within IMA (from highly critical to low critical), up to the entire system on board.

Security protection for the open part of A380 may possibly be extended to other more core systems, using same techniques. Move to decentralised architecture and smart sensors. Allow for more flexible reconfiguration, e.g., for maintenance purposes.

Enable grater level of integration on single IMA units: more applications on a single module.

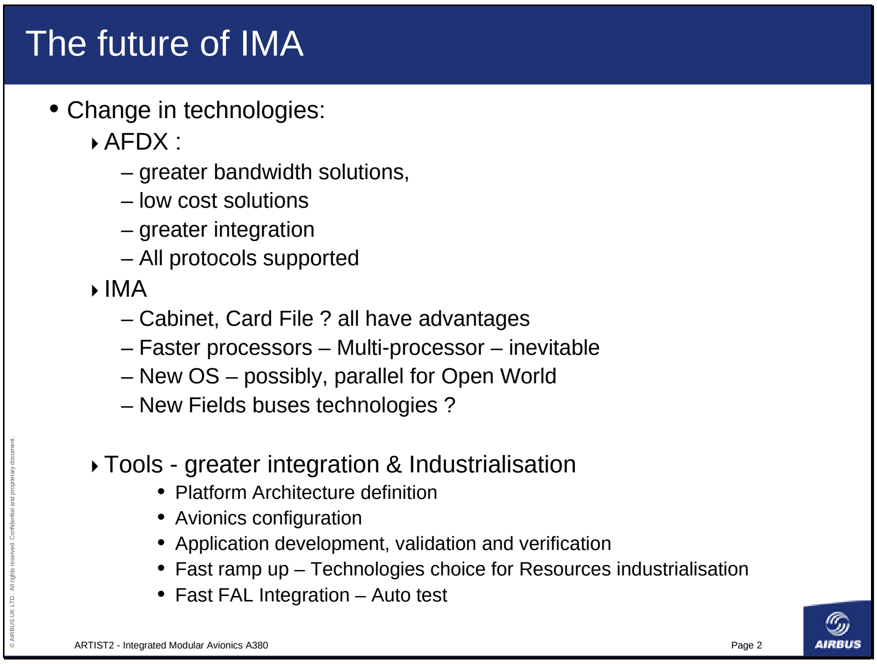


The future of IMA

- A380 IMA reused on A400M/A350 :
  - Mature avionics hardware available immediately
  - NRCs, risks minimised
- Next Aircraft – *IMA2G*:
  - Extend the scope of IMA
    - Flight controls, Open world
  - Increase the flexibility of IMA – “Generic Secure Platform”
    - Optimise the IMA architecture
      - Decentralised I/O / Smart sensors
      - Reconfiguration
    - Enable more systems to be integrated within IMA
      - High Critical to Low Critical
    - Enable greater levels of integration on single IMA units

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 1 AIRBUS



The future of IMA

- Change in technologies:
  - AFDX :
    - greater bandwidth solutions,
    - low cost solutions
    - greater integration
    - All protocols supported
  - IMA
    - Cabinet, Card File ? all have advantages
    - Faster processors – Multi-processor – inevitable
    - New OS – possibly, parallel for Open World
    - New Fields buses technologies ?
  - Tools - greater integration & Industrialisation
    - Platform Architecture definition
    - Avionics configuration
    - Application development, validation and verification
    - Fast ramp up – Technologies choice for Resources industrialisation
    - Fast FAL Integration – Auto test

© AIRBUS UK LTD. All rights reserved. Confidential and proprietary document.

ARTIST2 - Integrated Modular Avionics A380 Page 2 AIRBUS

Needed changes of technologies, see figure above:

- AFDX: grater bandwidth, all protocols supported (including video, voice…); will be kept because of performance and wide use.
- IMA: new OS?
- New field buses technologies, including TTA. For what is TTA best suited? Studies ongoing on Flexray (today CAN is the baseline).
- Toolset: greater integration and industrialisation of the entire toolset; full system virtual validation. Need for a platform offering full auto tests.

Discussion

Q: is it possible that the supplier has larger responsibility in how to achieve qualification of subsystems? ***A: the toolset should help supporting this****.*

Q: in what way current field bus technology is not adequate? ***A: For some systems like flight control, CAN is not adequate and we would like something else. Now, one additional technology is more costs. Not decided yet on this point.***

Q: question on tools; how do you have a global model of your system? ***A: Depends on system designer, not a single method is used. Some use Simulink...***
Q: How do you verify AFDX interaction? AFDX is not deterministic. ***A: it is deterministic and verified by design using "Confcheck". This Tool verifies that the performances requested by all the Virtual link of the networks will be satisfied****.*

Note that the different modules are synchronous. Just their clocks are not synchronised and therefore the resulting communication is not synchronous, regardless of the particular communication mode used. In addition, Ethernet communication is used, which uses asynchronous protocol. The system is deterministic only if the speeds is fast enough. Information on this is found in ARINC documents.

Q: is this not the reason for looking for TT architectures and buses? ***A: if enough time available, no need for a deterministic system managed by 1 clock to achieve this****.*

Q: this may not survive if you increase the usage pressure, there is some clock drift, and you increase the load, you may get problems. ***A: if enough time is available, no need for a deterministic system managed by 1 clock to achieve this.***

Q: future of reconfiguration capabilities, are you interested in this research direction? ***A: may be interesting but considered risky; will progress cautiously in this direction. Will depend on complexity and real A/C Benefit.***

Q: what about changes in SW maintenance? ***A: not the right person to answer. Having a clean platform with standardised API and configuration means are important elements.***

Thierry Cornilleau, Dassault-Aviation: Lessons learned by Dassault-Aviation from military and civil IMA applications

Design division of Dassault Aviation

MDPU –IMA in military aircrafts

Modular data processing unit, IMA in military aircraft. Started in 1998. For use in Rafale and Mirage 2000/9. Was developed with Thales. Aims at reducing cost, allow for third party developments. MDPU provides HW and SW resources that are generic.

MDPU reuses ASAAC concepts. This started in the US in 1980's, for Apache helicopter. Phase II of ASAAC started 1998. MDPU is based on available techno in 1999, e.g., SCI was selected for communications.

MDPU restricted to non critical functions or critical interruptible functions, not flight control.

MDPU generic HW architecture: it is a cabinet with modules inside and power supply, based on SCI backplane topology for communications. Air cooling is used. For the SW architecture, POSIX is used as an OS. There is an API to be interfaced with applications. There are also system applications such as configuration and synchronization (a global clock is used). Aim is to decouple application from HW architecture.

Inside an MDPU module, there is also a generic architecture with SCI backplane and an interface to the avionics global bus. This has a cost but allows incremental upgrades of this architecture. Module packaging is performed in a way that makes them easily replaceable by pilots or technicians not experts of these machines.

MDPU benefits:

- reduced weight, volume, power consumption, due to the sharing by several application of a same MDPU
- 2nd level of maintenance removed, thanks to LRM concept (Line replaceable module)
- room for upgrades
- open architecture and standardised interfaces, allows coping with obsolescence
- same resources can be used for different variants of aircrafts

EASY – IMA in Falcon business jets

Based on an avionics from Honeywell Primus Epic. Architecture built with Modular Avionics Units (MAU). Cabinet with up to 20 modules in it and power supply. Covered: auto-pilot, flight management, and more. The processors in modules include Pentium.

OS is common to several modules, uses a Honeywell OS close to ARINC 653. Core functions on top of the OS: fault history, file system, configuration, inter-module communications, and built-in test. Communication is through the backplane to ensure deterministic latency.

Communications rely on a dedicated cPCI backplane for intra-cabinet, on a redundant serial bus between MAUs, and on Ethernet for maintenance purpose.

Technology selected in 2000, first certification in 2004.

Key points

These key points give hints about how to handle more easily an IMA. They are not really new for a system designer but they remain true in an IMA context.

Industrial work share

Industrial work share evolved. Moving from federated architecture (where black-boxes were assembled) to modular architecture where diverse subsystems and components are integrated:

- platform supplier

- function supplier
- application suppliers
- system integrator

All this requires clarification of roles and interfaces.

Specifications

A top-down approach is used, with explicit non-functional specs. Observability requirements are formulated regarding the interfaces; interfaces must be clearly exposed. There is a provision for reconfiguration capabilities at run time ("mode changes"). Define a clear policy to handle this at certification time. Early validation is important.

COTS are reused from non-aircraft markets. Mostly used in previously experimented usage domains. Otherwise develop own solution.

Application development and integration

HW/SW decoupling is important. Offer a platform with deterministic behavior. Ensure that applications behave the same regardless from their localization.

Spatial and temporal partitioning is used. Platform is integrated first, before integrating the applications.

Try to predict early the needed size for the HW platform.

Next challenges

Increase of 3rd party development:

- Open interfaces & standards
- Work based on tool supported contracts; plug & play
- Data security concepts inside IMA: MILS (Multiple Independent Levels of Security)
- Application reuse, particularly when a new supplier is brought into the process
- Enhance complexity control: configuration capabilities (cold and warm reconfigurations of an IMA subject to certification). Early verification & validation, virtual design space exploration, predictable sizing.
- Support incremental certification.
- Assessment of new technologies:
  - multi-core
  - IT technologies in embedded world,
  - asynchronous networks for deterministic applications (TT buses are costly)
- commonalities between the different transport industries (automotive, train, aerospace); avoid reinventing the wheel

Conclusion

Successful concept used in civil and military projects. Open architectures allow manufacturers to subcontracts suppliers while keeping control of the system.

Discussion

Q: have you tools to specify blueprints? ***A: tables for configuration are mostly made by hand. AADL is a 1st progress.***

Q: are you going to use CORBA or other type of contract. ***A: interested in component concept, with notion of service required and offered.***

Q: How to face dynamically reconfigurable systems? ***A: for the moment, lots of tests at integration; but reconfiguration is interesting since you use fewer resources. There is in-flight reconfiguration after fault.***

Q: we have lots of experience in CORBA component models using CCM. ***A: If applied, CCM will only be used with static configuration.***

Q: what are the standards of interest? ***A: ARINC 653 is the basis. It is well designed. This gives the plug but not necessarily the play****.*

Q: using contracts in standards: how? ***A: this is an idea and need to work on this. AADL would be the good basis to standardize contracts, both functional and non-functional.***

Q: how to make computing time estimations? ***A: well, we expect technologies from the university to get as soon as possible in design process WCET and CPU loads****.*

Q: (Absint) there are solutions for WCET estimations with cache. You don't have to wait for solutions from the university... (Note: This works only if the application binary is available, but very often estimations are required more early in the development process)

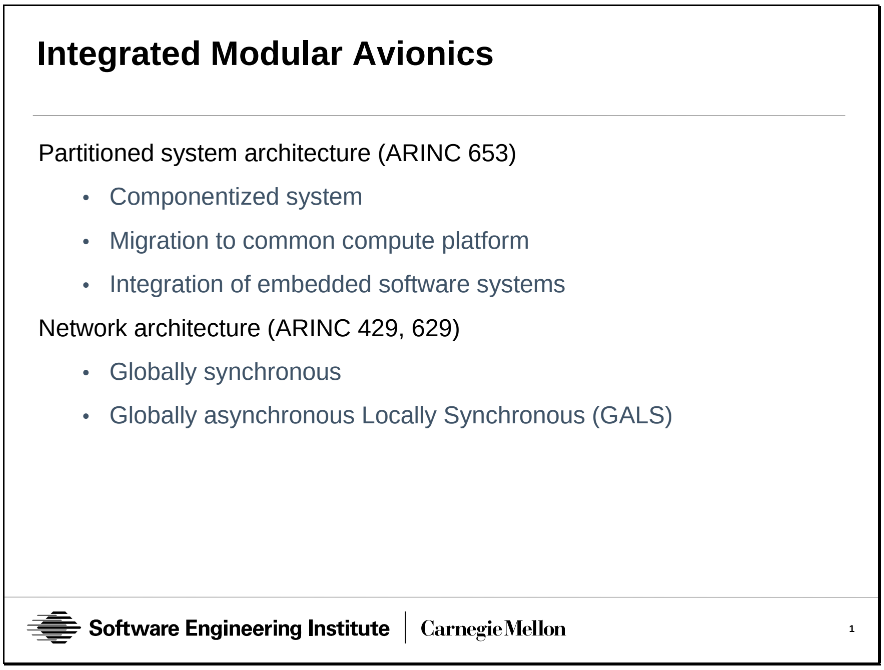
Q: how do you manage coordination between partitions and I/O. ***A: For high performance, we have asynchronous mechanisms. For determinism, on civil IMA, all communications go systematically through the bus.***

Q: is the number of modules going to further increase, or reduce? ***A: modules are more powerful but you want more applications, but sharing applications among modules reduces the number of modules used****.* ***No idea of how the overall load will increase – depends on the aircraft.***

Peter Feiler, SAE AADL Committee: IMA: The Good, The Bad, and The Ugly

SW engineering institute at CMU in Pittsburgh. Technical leader for AADL standard.

The good



Integrated Modular Avionics

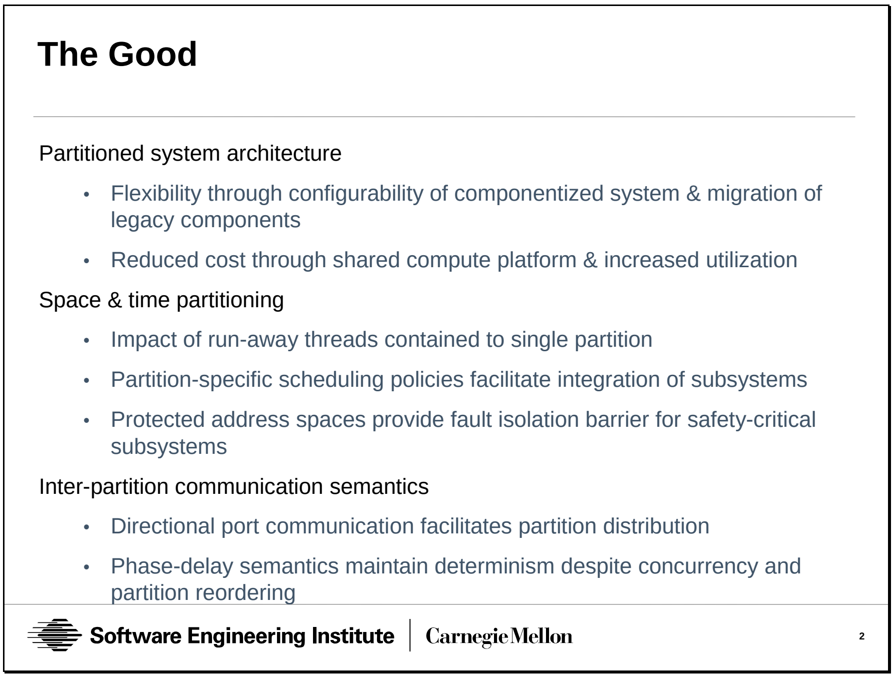
Partitioned system architecture (ARINC 653)

- Componentized system
- Migration to common compute platform
- Integration of embedded software systems

Network architecture (ARINC 429, 629)

- Globally synchronous
- Globally asynchronous Locally Synchronous (GALS)

Software Engineering Institute | Carnegie Mellon 1



The Good

Partitioned system architecture

- Flexibility through configurability of componentized system & migration of legacy components
- Reduced cost through shared compute platform & increased utilization

Space & time partitioning

- Impact of run-away threads contained to single partition
- Partition-specific scheduling policies facilitate integration of subsystems
- Protected address spaces provide fault isolation barrier for safety-critical subsystems

Inter-partition communication semantics

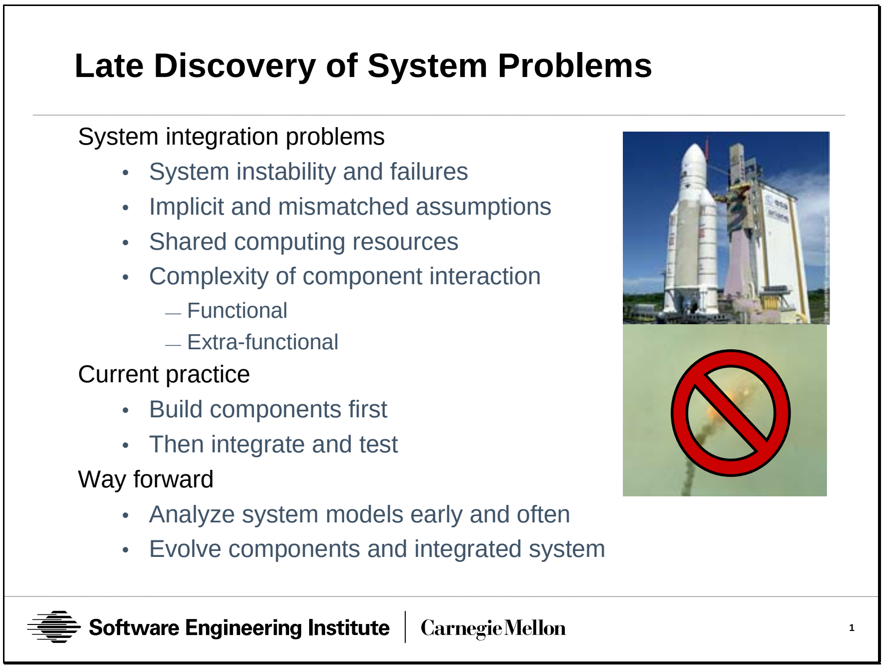
- Directional port communication facilitates partition distribution
- Phase-delay semantics maintain determinism despite concurrency and partition reordering

Software Engineering Institute | Carnegie Mellon 2

Spae&time partitioning is important; impact of run-away threads contained to a single partition. Partition themselves are statically scheduled; but inside partitions you can use dynamic scheduling or other policy, with preemption.

Inter-partition communication guarantees determinism despite concurrency.

The bad



Late Discovery of System Problems

System integration problems

- System instability and failures
- Implicit and mismatched assumptions
- Shared computing resources
- Complexity of component interaction
  - Functional
  - Extra-functional

Current practice

- Build components first
- Then integrate and test

Way forward

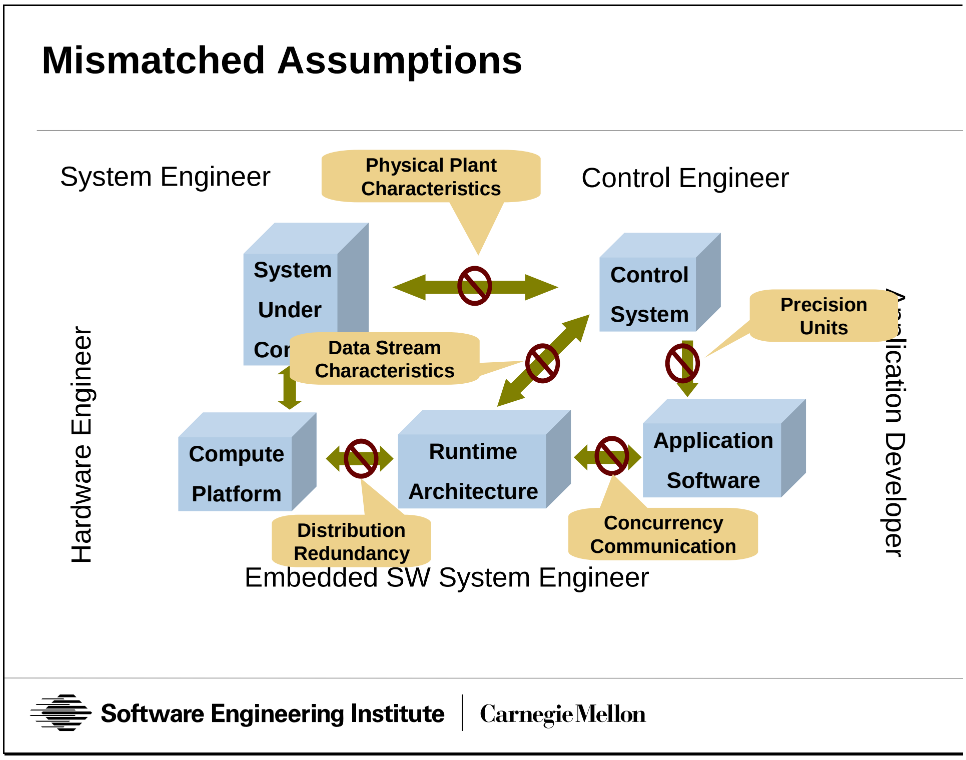
- Analyze system models early and often
- Evolve components and integrated system

Software Engineering Institute | CarnegieMellon 1

Despite fault tolerance built into systems problems are discovered during integration and operation. Examples are System instability and failures, implicit and mismatched assumptions, complexity of component interaction (functional and non-functional).

Way forward: evolve components and integrate them, not just integrate existing components.

Partition assumptions



Mismatched Assumptions

System Engineer

Physical Plant Characteristics

Control Engineer

System Under Cor

Control System

Precision Units

Data Stream Characteristics

Hardware Engineer

Application Developer

Compute Platform

Runtime Architecture

Application Software

Distribution Redundancy

Concurrency Communication

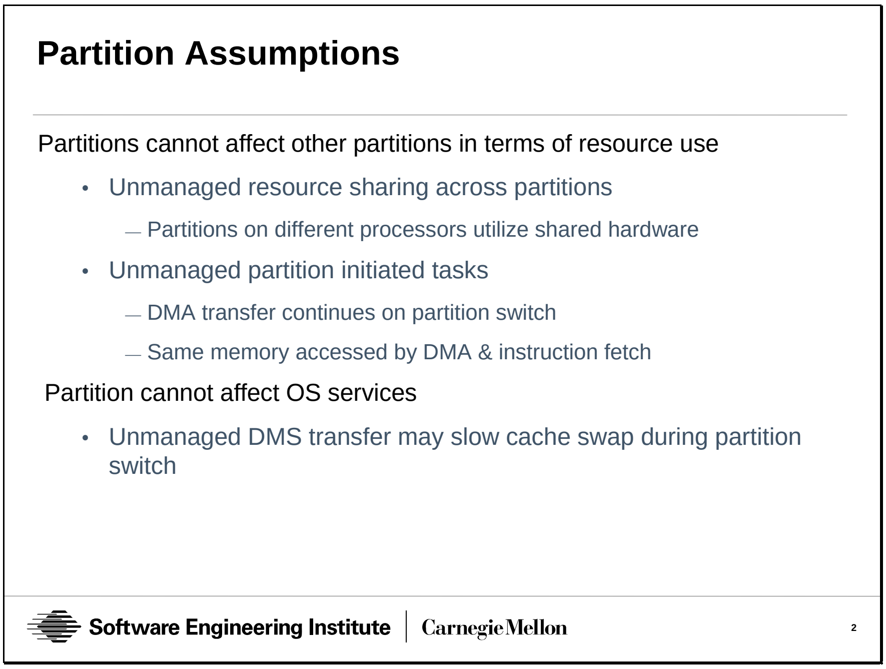
Embedded SW System Engineer

Software Engineering Institute | Carnegie Mellon

These problems are due to a shift in focus on system integration from a system engineering perspective to a second focus on embedded software system integration.

In this context additional engineering roles interact with each other. These interactions are based on assumptions that are mismatched and not validated. For example control engineers make assumptions about the physical plant in their control algorithm parameters. The translation of these algorithms into code leads to choices such as use of 16 bit arithmetic, which places bounds on domain values represented by the variables, which may be violated by the physical system (Ariane 5).

Similarly, assumptions about sequential execution of separate tasks may result in unexpected behavior when tasks are executed concurrently, e.g., preemptive scheduling of tasks with shared variable communication. Similarly, their mapping onto the computing platform may result in violation of redundancy assumptions. Finally, assumptions about data streams being processed by control algorithms, such as latency jitter, age, and missing stream elements, may not be upheld by the software runtime system and cause controller instability.



Partition Assumptions

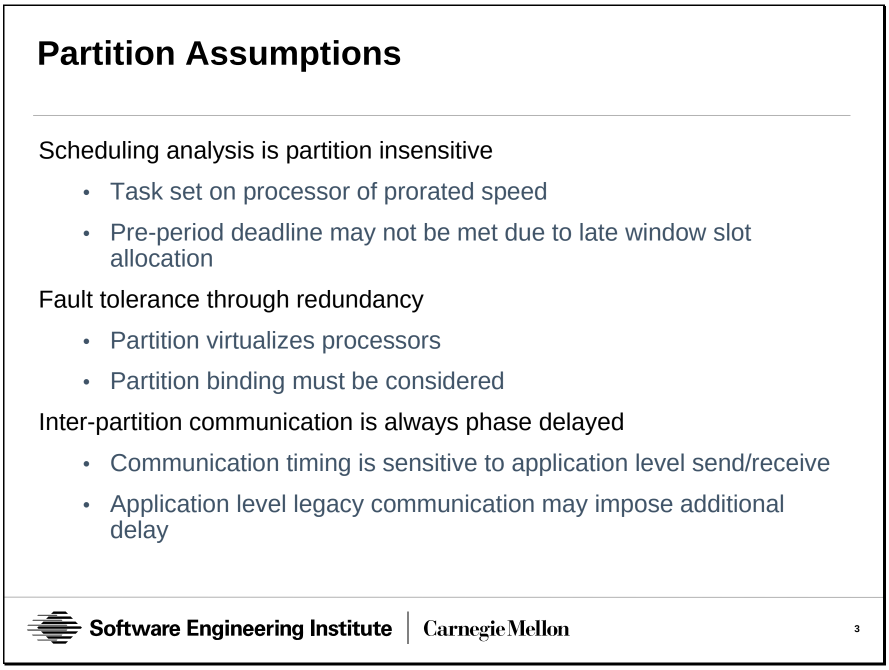
Partitions cannot affect other partitions in terms of resource use

- Unmanaged resource sharing across partitions
  - Partitions on different processors utilize shared hardware
- Unmanaged partition initiated tasks
  - DMA transfer continues on partition switch
  - Same memory accessed by DMA & instruction fetch

Partition cannot affect OS services

- Unmanaged DMS transfer may slow cache swap during partition switch

Software Engineering Institute | Carnegie Mellon 2



Partition Assumptions

Scheduling analysis is partition insensitive

- Task set on processor of prorated speed
- Pre-period deadline may not be met due to late window slot allocation

Fault tolerance through redundancy

- Partition virtualizes processors
- Partition binding must be considered

Inter-partition communication is always phase delayed

- Communication timing is sensitive to application level send/receive
- Application level legacy communication may impose additional delay

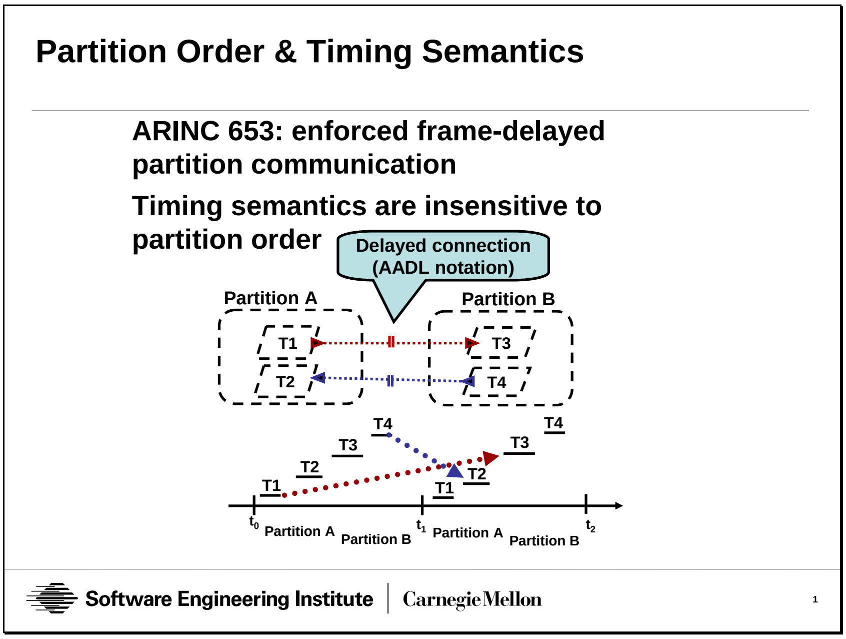
Software Engineering Institute | Carnegie Mellon 3

Partition assumptions:

Partitions are intended to provide predictability through spade & time partitioning. However, assumptions about the use of partitions to guarantee no side effects of multiple partitions sharing processors and other compute platform resources are not always valid. The above slides give some examples encountered in actual systems.

Such assumptions must be documented as they are not avoidable. Make this explicit and include the problem in the analysis, do not hide it.

There is a second class of system wide problems that center around data stream processing – as found in control systems – and assumptions about their characteristics. These are the ugly aspects of partitioned architectures as they propagate mismatched assumptions, i.e., faults, throughout the system.



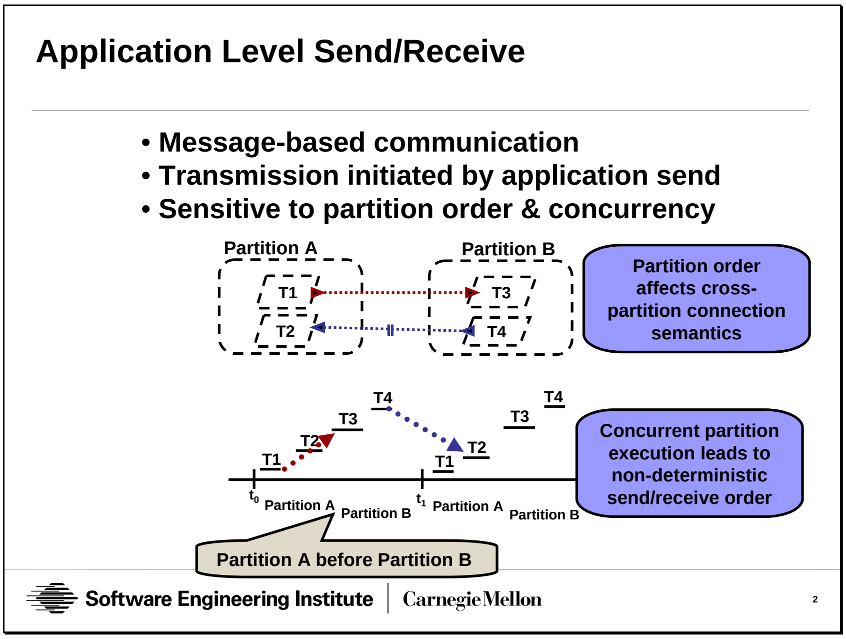
Partition Order & Timing Semantics

ARINC 653: enforced frame-delayed partition communication

Timing semantics are insensitive to partition order

Delayed connection (AADL notation)

Software Engineering Institute | Carnegie Mellon 1



Application Level Send/Receive

- Message-based communication
- Transmission initiated by application send
- Sensitive to partition order & concurrency

Partition order affects cross-partition connection semantics

Concurrent partition execution leads to non-deterministic send/receive order

Partition A before Partition B

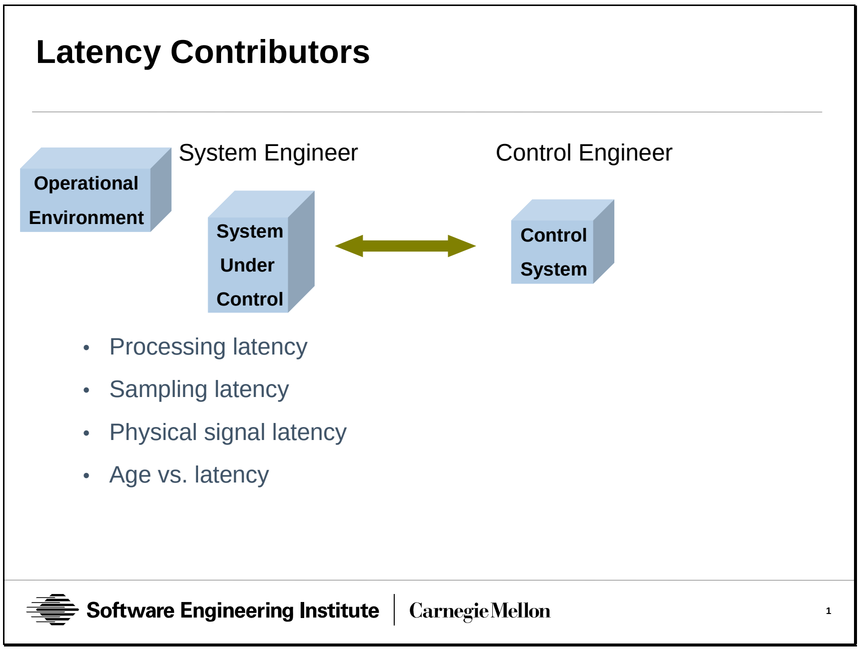
Software Engineering Institute | Carnegie Mellon 2

The ugly

Impact of latency jitter

Latency and jitter can affect the stability of the entire system. Variations of actual write & read times due to preemption and concurrency. Introduce SW noise in the control system.

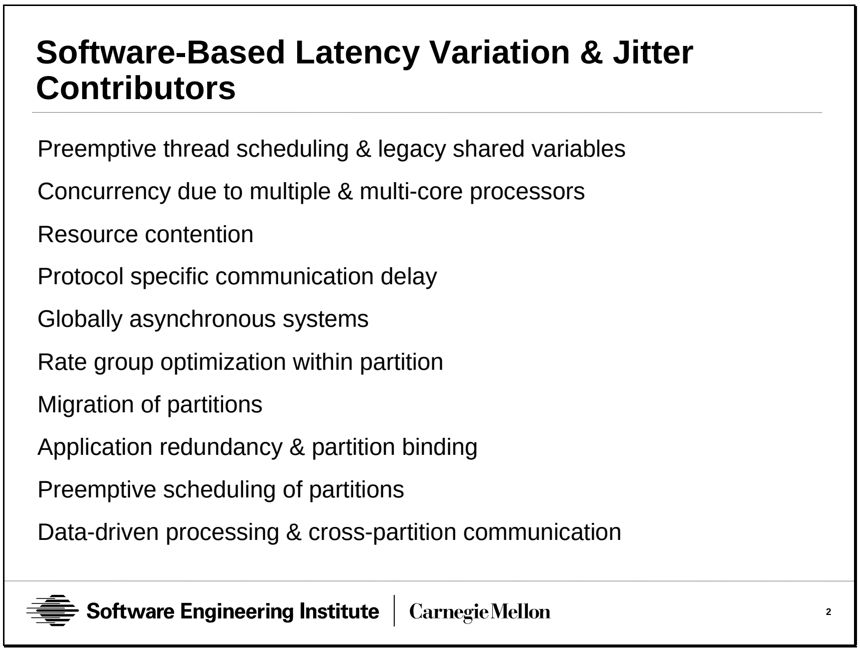
Latency is a topic where both SW and control engineers meet and must understand each other, see figures below. Physical plants introduce latency. Lots of artifacts cause additional jitter, even loss/duplication of data.



Latency Contributors

- Processing latency
- Sampling latency
- Physical signal latency
- Age vs. latency

Software Engineering Institute | Carnegie Mellon 1



Software-Based Latency Variation & Jitter Contributors

Preemptive thread scheduling & legacy shared variables

Concurrency due to multiple & multi-core processors

Resource contention

Protocol specific communication delay

Globally asynchronous systems

Rate group optimization within partition

Migration of partitions

Application redundancy & partition binding

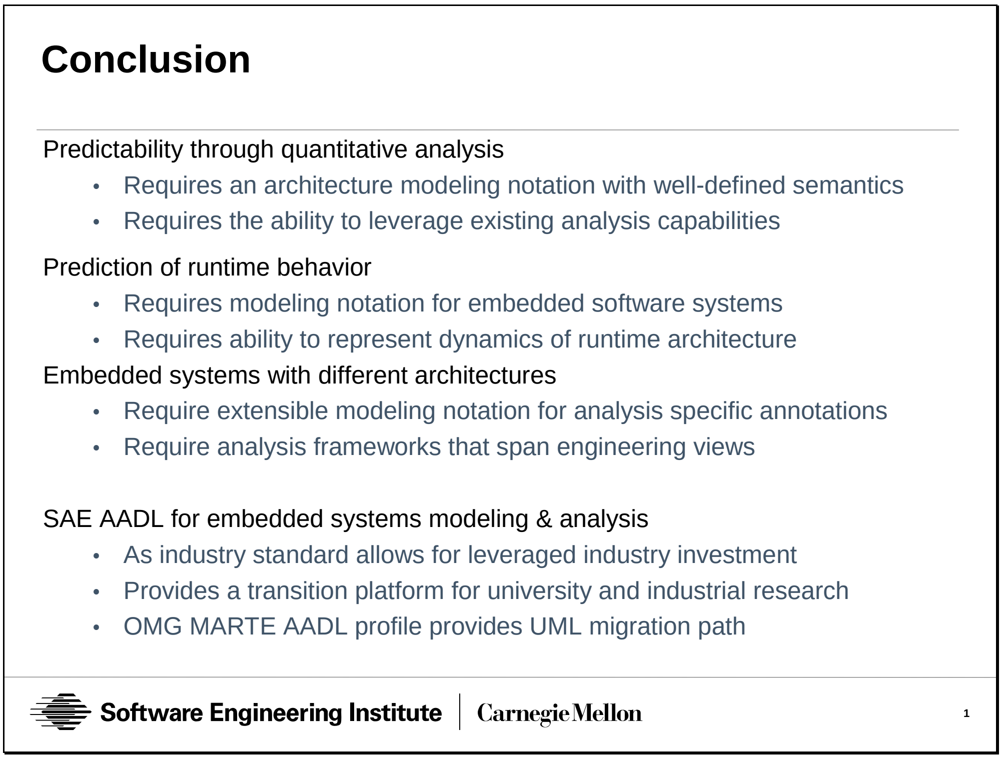
Preemptive scheduling of partitions

Data-driven processing & cross-partition communication

Software Engineering Institute | Carnegie Mellon 2

From the perspective of a control engineer there are three major contributors to latency. Furthermore, there is a distinction between latency and age of data. Control algorithms are tuned based on end-to-end latency analysis from a control engineer’s perspective.

When implemented as a embedded software system, a number of contributors to latency, latency jitter, and age are purely due to the fact that we have a software-based implementation and are often not fully taken into account in the analysis. This results in the perception that control systems receive data from sensors that are noisier and unless taken into account in a robust control system design will lead to instability behavior due to the dynamics of the executing software.



Conclusion

Predictability through quantitative analysis

- Requires an architecture modeling notation with well-defined semantics
- Requires the ability to leverage existing analysis capabilities

Prediction of runtime behavior

- Requires modeling notation for embedded software systems
- Requires ability to represent dynamics of runtime architecture

Embedded systems with different architectures

- Require extensible modeling notation for analysis specific annotations
- Require analysis frameworks that span engineering views

SAE AADL for embedded systems modeling & analysis

- As industry standard allows for leveraged industry investment
- Provides a transition platform for university and industrial research
- OMG MARTE AADL profile provides UML migration path

Software Engineering Institute | Carnegie Mellon 1

Conclusion

In this presentation, we focused on performance issues. Other non-functional properties such as security and safety-criticality can also be impacted by the choices being made in the runtime architecture of embedded software systems.

SAE AADL has been designed as an architecture modeling language specifically for embedded systems, and as such provides execution semantics for tasks and communication that allows for analysis of non-functional properties. In particular, it has provided an abstraction of flows with mid-frame and phase-delayed communication semantics as well as sampled and message-based processing. This abstraction allows models to capture the perspective of control engineers and communicate it to embedded software system engineers.

For the UML community OMG MARTE with the AADL profile is an important ongoing development that allows developers to leverage their existing UML tool investments and integrate embedded software system engineering with system engineering through SysML.

Discussion

Calls for solid design techniques based on architecture models of not just the conceptual system or a platform independent architecture, but also the compute platform and physical environment at the appropriate level of abstraction. Industry has recognized the shortcomings of UML as a notation to model the runtime architecture of embedded software system. The challenge is to model the "what" and abstract away from the "how". This provides an analytic framework for understanding the intent of the embedded application, and through separation of concerns address the appropriate implementation *choices, which in the long run can be addressed through a generation approach from a platform specific model. SAE AADL and OMG MARTE are addressing this need.*

Q: what experience with AADL in industrial domain.

A: A number of pilot project have been carried out by companies in the avionics, aerospace, and medical domain. The success is not so much in the modeling itself but in the analysis of system architectures. The Mars Rover is a nice example, where priority inversion was detected through analysis and corrected remotely through appropriate runtime support for this system condition.

Paul Caspi, Verimag: Some issues about IMA in safety critical applications

Was involved in consulting activities in embedded control systems and certification activities: Hermes space-shuttle, RER-A emergency braking, driver-less Lyon subway, Certifer certification agency. Was faced with the IMA question in many of these cases.

Where does this viewpoint come from?

Computer technology is poorly reliable:

- cars recalled for computing bugs
- Ariane V
- electricity and telephone crashes

Two questions:

- Is it wise to use this technology in safety critical systems?
- Why, despite of everything, this technology appears as no so bad after all?

Why simple federated architectures were so successful?

Some partial answers:

- these industries have developed very solid methods
- very robust computing architecture HW/SW
    - each computer has a periodic behavior triggered by a quasi periodic clock which triggers a single loop software; no dynamic memory, no unbounded loop;
    - WCET relatively easy; jitter minimized
    - little use of OS (a single interrupt taking place when the computer is idle); these programs use to work on bare machines, with no OS at all
- computers periodically sample the physical world but also the other computers
    - non blocking *communication by sampling*
    - can be found in lots of safety-critical systems
    - later modified by so-called Time-Triggered Architecture
- Segregation between critical and less critical tasks

  - allows extending FMEA/FTA methods from HW to computers and SW
  - criticality inherited backward from outputs to tasks

Two accidentss that could have been avoided by simple federated architectures

ARIANE V

Overflow causes failure

A conjunction of several development flaws, including (there are others):

- segregation: the task responsible for flaw was non critical; but was packed in the same computer as critical ones
- since the function was not critical, exceptions needed not be caught

Priority inversion in the Mars Pathfinder

Priority inversion is known to take place when multi-tasking (threading) is used in conjunction with synchronization (semaphores).

However, multi-tasking is mandatory in several cases:

- multi-periodic systems
- mix time- and event-triggered systems

Synchronization was considered needed for communication between tasks. *Wrong! Other solutions exist!*

Multi-tasking raises scheduling problems: efficient policies and tests exist for periodic and event-triggered systems

Multi-tasking raises communication problems:

- preemption can corrupt data
- critical sections can be a solution
- scheduling tests can take this into account

What kind of guarantee should we require for IMA?

- an important certification principle is: *non regression*; keep same safety as before
- IMA has thus to give evidence that
  - no side effect can result from violating the segregation principle
  - no side effect can result from violating the single thread principle
  - more… [not easy to draw a complete list of risks]

Thus the use of a new technology like IMA should be thoroughly justified and its introduction should be progressive and careful.

Discussion

Q: Airbus has applied IMA on flight warning. The ARIANE case is rather an issue of reuse of SW. Try to avoid by design the issues that were mentioned. When developing modules at level A, we enforce segregation by design. Main issue in the future will be SW reuse. This might become a source problem. When performing reuse, keep within the same domain, reusing in this way the domain *assumptions.* ***A: agree that there are other reasons for ARIANE V case than those mentioned. But still was a good illustration of violating segregation. No semaphore, no critical sections were used for communication in A340, by chance (or by consciousness!!).***

Q: the ARIANE case is 11 years ago… The important fact is that that in the mean time there has been lots of growth in SW development techniques. Need for incremental validation techniques. Otherwise the integration time grows exponentially; not very reassuring; a good path to bad situations.

Q: do you mean that level B,C should also go through severe qualification procedures? ***A: when I am aware of side effect, then I know how criticality propagates and can qualify what is B,C properly. In the case we discussed, it should have been inherited as an A. Of course this is not a good technique since then everything would become level A. Instead, show that non-critical pieces of SW cannot harm critical ones.***

John Rushby, SRI: Compositional Assurance for IMA [presented by Albert Benveniste, in absence of John Rushby]

Selected slides enclosed

Just-In-Time Certification

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Certification

- Provides assurance that deploying a given system does not pose an unacceptable risk of adverse consequences
- Certification methods should be effective (i.e., they work) and credible (i.e., they work for the reason we think they do)
- Current methods have been effective, but are they credible?
- Current methods of assurance explicitly depend on
  - Standards and regulations
  - Rigorous examination of the whole, finished system

  And implicitly on
  - Conservative practices
  - Safety culture
- All of these are changing

Overview

- Scientific certification
- Compositional certification
- Just-in-time certification

A Recent Incident

- Fuel emergency on Airbus A340-642, G-VATL, on 8 February 2005 (AAIB SPECIAL Bulletin S1/2005)
- Toward the end of a flight from Hong Kong to London: two engines shut down, crew discovered they were critically low on fuel, declared an emergency, landed at Amsterdam
- Two Fuel Control Monitoring Computers (FCMCs) on this type of airplane; they cross-compare and the "healthiest" one drives the outputs to the data bus
- Both FCMCs had fault indications, and one of them was unable to drive the data bus
- Unfortunately, this one was judged the healthiest and was given control of the bus even though it could not exercise it
- Further backup systems were not invoked because the FCMCs indicated they were not both failed

Implicit and Explicit Factors

- See also ATSB incident report for in-flight upset of Boeing 777, 9M-MRG (Malaysian Airlines, near Perth Australia)
- Maybe effectiveness of current certification methods depends on implicit factors such as safety culture, conservatism
- Current business models are leading to a loss of these
  - ◦ Outsourcing, COTS, complacency, innovation
- Surely, a credible certification regime should be effective on the basis of its explicit practices
- All assurance is based on **arguments** that purport to justify certain **claims**, based on documented **evidence**
- There are two approaches to assurance: standards-based, and goal-based

The Standards-Based Approach to Software Certification

- E.g., airborne s/w (DO-178B), security (Common Criteria)
- Applicant follows a prescribed method (or processes)
  - ◦ Delivers prescribed outputs
    - ⋆ e.g., documented requirements, designs, analyses, tests and outcomes, traceability among these
- Standard usually defines only the evidence to be produced
- The claims and arguments are implicit
- Hence, hard to tell whether given evidence meets the intent
- Works well in fields that are stable or change slowly
  - ◦ Can institutionalize lessons learned, best practice
    - ⋆ e.g. evolution of DO-178 from A to B to C
- But less suitable with novel problems, solutions, methods

The Goal-Based Approach to Software Certification

- E.g., air traffic management (CAP670 SW01), UK aircraft
- Applicant develops an assurance case
  - Whose outline form may be specified by standards or regulation (e.g., MOD DefStan 00-56)
  - Makes an explicit set of goals or claims
  - Provides supporting evidence for the claims
  - And arguments that link the evidence to the claims
    - ⋆ Make clear the underlying assumptions and judgments
    - ⋆ Should allow different viewpoints and levels of detail
- The case is evaluated by independent assessors
  - Explicit claims, evidence, argument

Multiple Forms of Evidence

- More evidence is required at higher Levels/EALs/SILs
- What's the argument that these deliver increased assurance?
- Generally an implicit appeal to diversity
  - And belief that diverse methods fail independently
  - Not true in n-version software, should be viewed with suspicion here too
- Need to know the arguments supported by each item of evidence, and how they compose
- Want to distinguish rational multi-legged cases from nervous demands for more and more and . . .
  - Bayesian Belief Networks (BBNs) can formalize these

A Science of Certification

- Certification is ultimately a judgment
- But the judgment should be based on rational argument supported by adequate explicit and credible evidence
- A Science of Certification would be about ways to develop that argument and evidence
- Favor goal-based over standards-based approaches
  - At the very least, expose and examine the claims, arguments and assumptions implicit in standards
- Be wary of demands for more and more evidence, with implicit appeal to diversity and independence
  - Instead favor explicit multi-legged cases
- Use formal ("machinable") design descriptions
  - Can then use automated analysis methods

Systems and Components

- The FAA certifies airplanes, engines and propellers
- Components are certified only as part of an airplane or engine
- That's because it's the interactions that matter and it's not known how to certify these compositionally
- So no alternative to looking at the whole system
- But modern engineering and business practices use massive subcontracting and component-based development that provide little visibility into subsystem designs
- Strong case for "pre-certification" of components

  Business case: Component vendors want it (cf. IMA)

  Certification case: simple extensions to current approach are too onerous or lack credibility (cf. DO-297)

Compositional Analysis

- Computer scientists have ways to do compositional verification of programs—e.g., prove
  - Program A guarantees P if environment ensures Q
  - Program B guarantees Q if environment ensures P

  Conclude that $A || B$ guarantees P and Q
- Assumes programs interact only through explicit computational mechanisms (e.g., shared variables)
- Software and systems can interact through other mechanisms
  - Computational context: shared resources
  - Noncomputational mechanisms: the controlled plant
- So compositional certification is harder than verification

Unintended Interaction Through Shared Resources

- This must not happen
- Need an integration framework (i.e., an architecture) that guarantees composability and compositionality

  Composability: properties of a component are preserved when it is used within a larger system

  Compositionality: properties of a system can be derived from those of its components
- This is what partitioning is about
- Or separation in a MILS security context

Composability

Partitioning ensures **composability** of components

- Properties of a collection of interacting components are preserved when they are placed (suitably) in the environment provided by a collection of partitioning mechanisms
- Hence partitioning does not get in the way
- And the combination is itself composable
- Hence components cannot interfere with each other nor with the partitioning mechanisms

Additivity

Partitioning mechanisms compose with each other **additively**

- e.g., **partitioning(kernel)** + **partitioning(network)** provides **partitioning(kernel + network)**
- There is an asymmetry: partitioning network stacks and file systems and so on run as clients of the partitioning kernel

Partitioning (composability and additivity) make the world safe for compositional reasoning

Unintended Interaction Through The Plant

- The notion of interface must be expanded to include assumptions about the noncomputational environment (i.e., the plant)
    - Cf. Ariane V failure (due to differences from Ariane IV)
- Compositional reasoning must extend to take the plant into account (i.e., composition of hybrid systems)
- Control engineers do this, computer scientists are less familiar with it
    - Assumption generation is attractive
- Must also consider response to failures
    - Avoid domino effect
    - Control number of cases (otherwise exponential)

Compositional Certification

- This is a big research challenge
- It demands clarification of the difference between verification and certification, and the role of partitioning
- And explication of what constitutes an interface to a certified component
    - e.g., the notion of interface automata
    - The certification data is in terms of the interface only
    - You cannot look inside when analysing compositions
- Compositional certification should extend to incremental certification, reuse, and modification
- It's also the big challenge for regulatory agencies
    - A completely different way of doing business

Late(r) Binding

- More and more functionality is being determined later than the time at which certification is performed
- E.g., kernel configuration determined at load time
  - 15 KSLOC in certified kernel
  - 50 KSLOC of XML for configuration
- SOA and self-assembly
- AI planning
- Runtime adaptation and learning
- How can these be certified?

Monitoring and Synthesis

- Certification rests on consideration of reachable states
- Scientific certification uses formal methods to calculate and analyze these at design time
- Instead, we could use these methods to construct monitors that check behavior at runtime
  - `www.runtime-verification.org`
- Or to synthesize controllers to generate safe behavior
  - Ramage and Wonham: controller synthesis

Runtime Assurance

- Instead of design-time analysis of implementation
- Use run-time monitoring or synthesis of behavior from models
  - Typically with a receding horizon (bounded lookahead)
  - Fewer possibilities to examine, known current state
- Each component makes its model available to others, pursues its own goals while ensuring that possible moves by others cannot trap it into following a bad path, or cause violation of safety
  - Analyzed as a game: guarantee a winning strategy
- Instead of using model checking and other formal methods for analysis, we use them for monitoring and synthesis

Just-In-Time Certification

- Some of the verification and certification activity is moved from design-time to run-time
- We trust automated verification methods for analysis, so why not trust them for monitoring and synthesis?
  - Certification examines the models, trusts the synthesis
- Will need to consider time-constrained synthesis
  - Anytime algorithms
  - Seek improvements on safe default
- Some analysis methods can deliver a certificate (e.g., a proof), used for synthesis that would truly be just-in-time certification!

A Research Agenda

- A Science of Certification
  - Or the science for certification
- Specification and verification of integration frameworks
  - Partitioning, separation, buses, kernels
- High-performance automated verification for strong properties of model-based designs
  - Mostly infinite state and hybrid systems

  And automation of related processes (test generation, FTA)
- Compositional certification
  - Composition of hybrid systems
- Tool qualification
  - Evidence management
- Just-in-time certification and runtime synthesis

Gert Döhmen, Airbus Germany: Embedded System Development for Distributed Networked Computing Platforms

GD is with Airbus Germany, development for IMA for A380. Has some background on how this was done for the A380. Now he is leading EU-SPEEDS project for embedded systems development.

The SPEEDS project

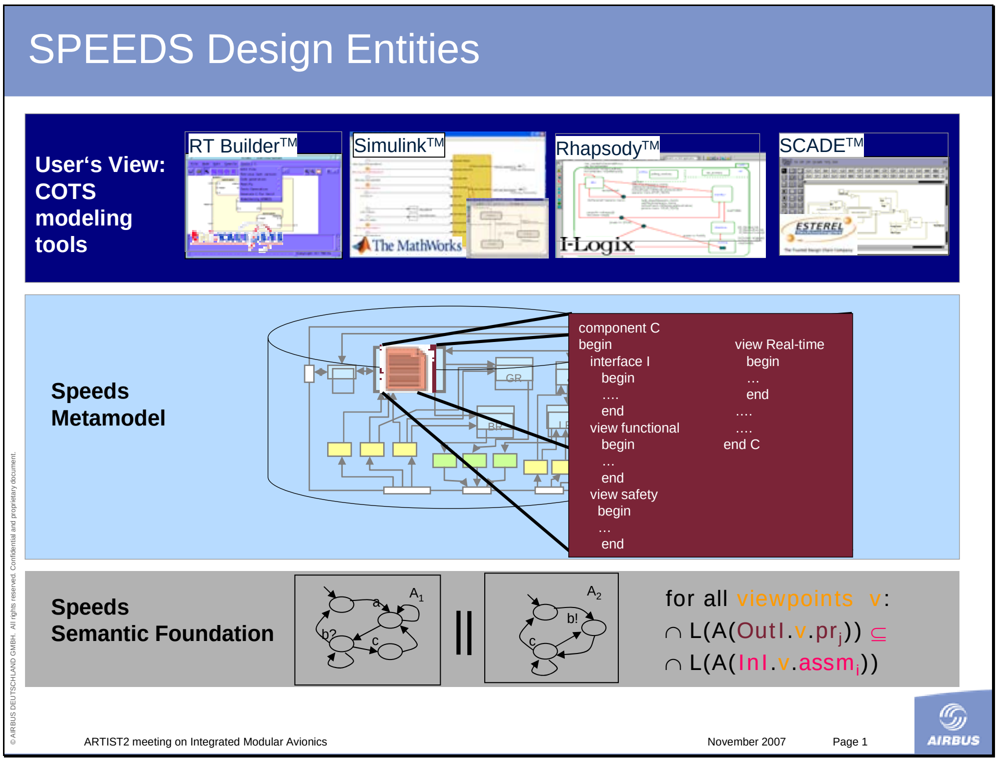
- “Fool-proof” representation of systems using HRC (Heterogeneous Rich Components)
- Formal analyses to verify compatibility, consistency of systems
- Process control/monitoring techniques to evaluate progress, maturity
- SPEEDS BUS to allow transfer of data between tools

HRC:

- multi-viewpoint components
- assumptions on environment
- compatible with AUTOSAR, SysML profile; complements MARTE by its emphasis on composition, not just timing

Contracts

- assumption/guarantee
- horizontal/vertical
- multiple viewpoints



SPEEDS Design Entities

User‘s View: COTS modeling tools

Speeds Metamodel

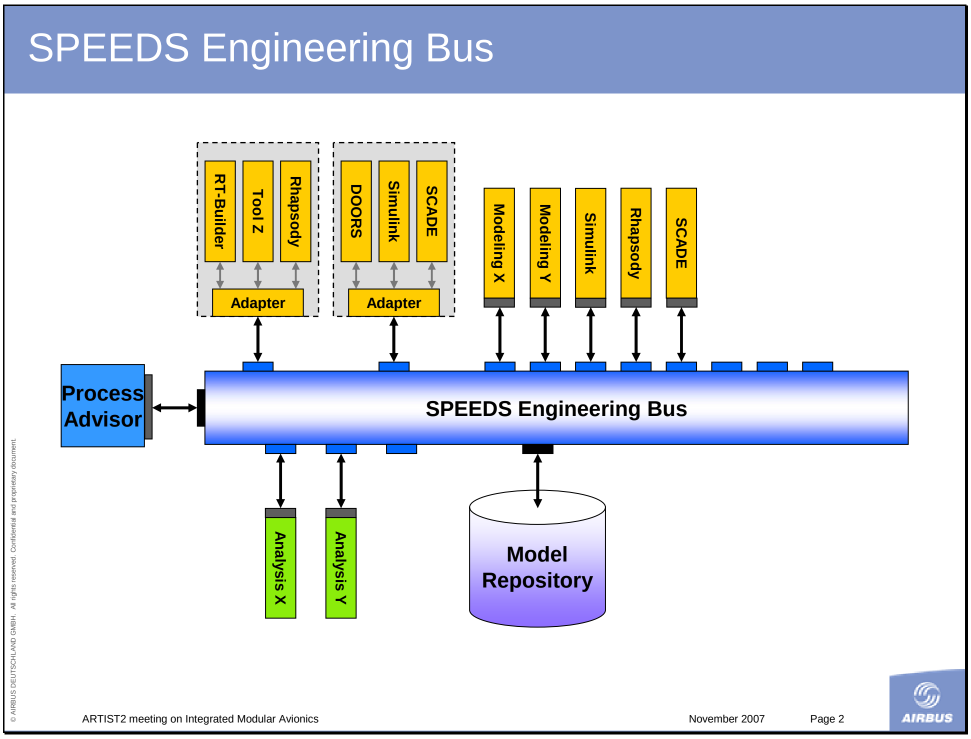
```
component C
begin
  interface I
    begin
    ….
    end
  view functional
    begin
    …
    end
  view safety
   begin
   …
    end
                 view Real-time
                   begin
                   …
                   end
                 ….
                 ….
               end C
```

Speeds Semantic Foundation

for all viewpoints v:
$\cap$ L(A(OutI.v.pr$_j$)) $\subseteq$
$\cap$ L(A(InI.v.assm$_i$))

ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 1

© AIRBUS DEUTSCHLAND GMBH. All rights reserved. Confidential and proprietary document.

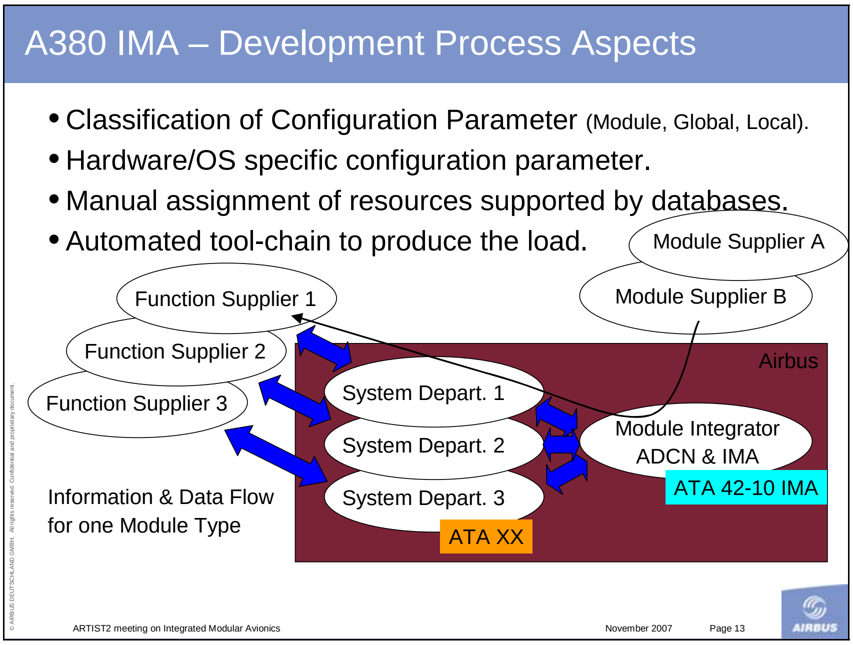


SPEEDS Engineering Bus

© AIRBUS DEUTSCHLAND GMBH. All rights reserved. Confidential and proprietary document.

ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 2

Distributed networked computing platform



A380 IMA – Development Process Aspects

- Classification of Configuration Parameter (Module, Global, Local).
- Hardware/OS specific configuration parameter.
- Manual assignment of resources supported by databases.
- Automated tool-chain to produce the load.

© AIRBUS DEUTSCHLAND GMBH. All rights reserved. Confidential and proprietary document.

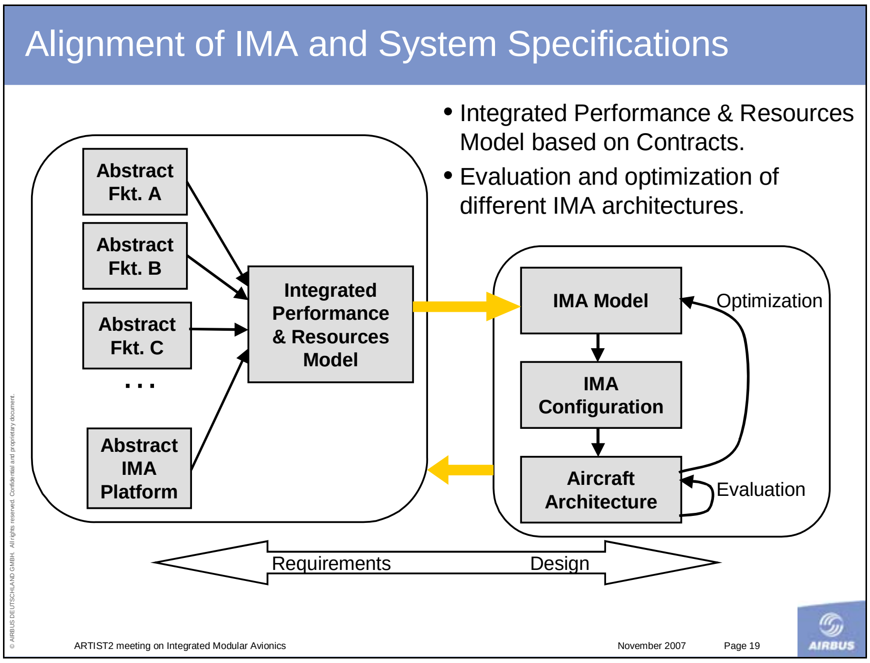
ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 13

High parallel IMA development process requests formal requirement and design specifications in order to cope with rising complexity of distributed systems. Model based design-space exploration techniques will be needed to analyze and optimize future IMA architectures.

Specification architecture & validation is a fairly complex process with information exchange and alignment between IMA development and the development of different functions on IMA. There is a lot of room for improvement here.

Using SPEEDS for IMA development

Rich Component Models (RCM) with contracts (assumptions/promises) supporting functional & nonfunctional aspects will specify requirements/constrains on IMA platform and different functions. Those abstract models cover aspects of System Requirements Documents. Linking those models provides Integrated Performance & Resources Model.

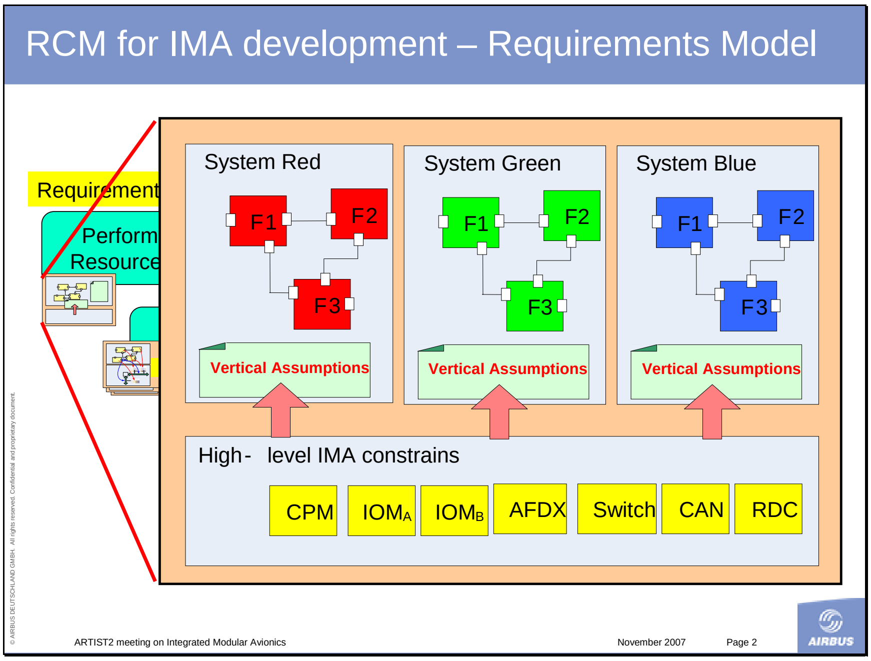


Alignment of IMA and System Specifications

- Integrated Performance & Resources Model based on Contracts.
- Evaluation and optimization of different IMA architectures.

Abstract Fkt. A
Abstract Fkt. B
Abstract Fkt. C
. . .
Abstract IMA Platform
Integrated Performance & Resources Model
IMA Model
IMA Configuration
Aircraft Architecture
Optimization
Evaluation
Requirements
Design

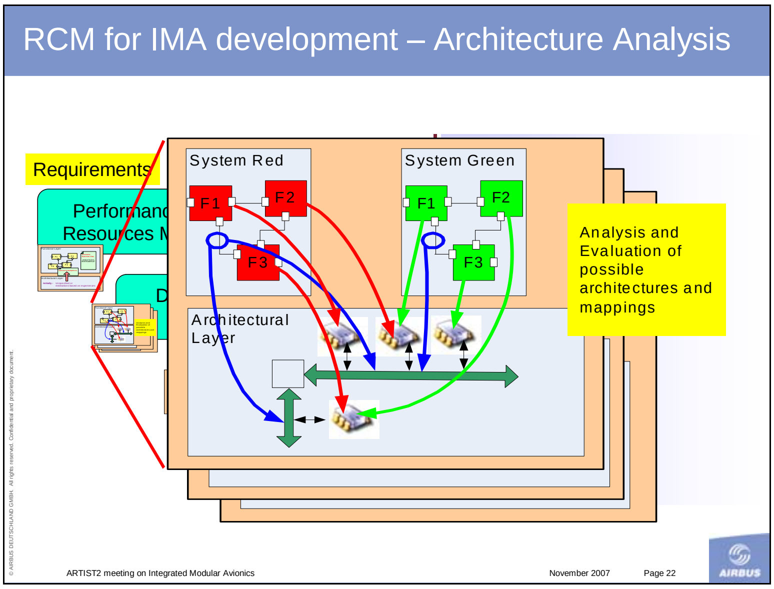
ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 19



RCM for IMA development – Requirements Model

Requirement
Perform
Resource
System Red
F1
F2
F3
Vertical Assumptions
System Green
F1
F2
F3
Vertical Assumptions
System Blue
F1
F2
F3
Vertical Assumptions
High- level IMA constrains
CPM
IOMA
IOMB
AFDX
Switch
CAN
RDC

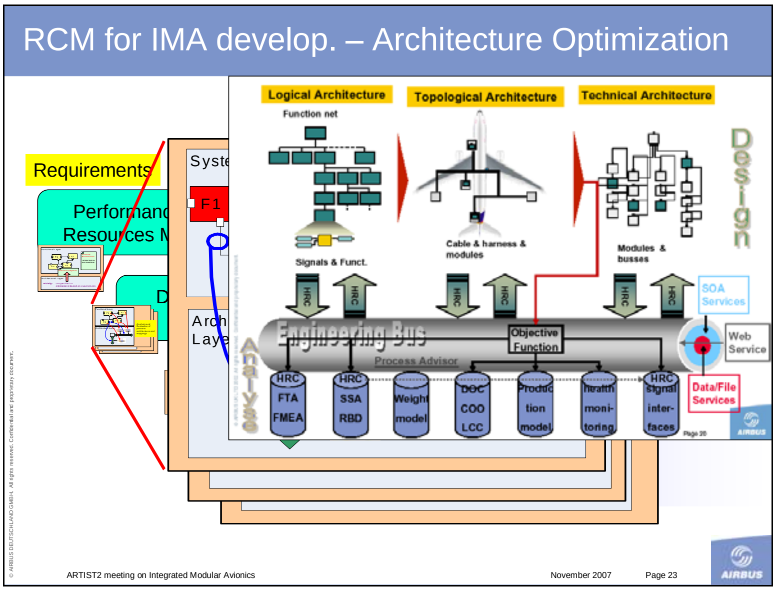
ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 2



RCM for IMA development – Architecture Analysis

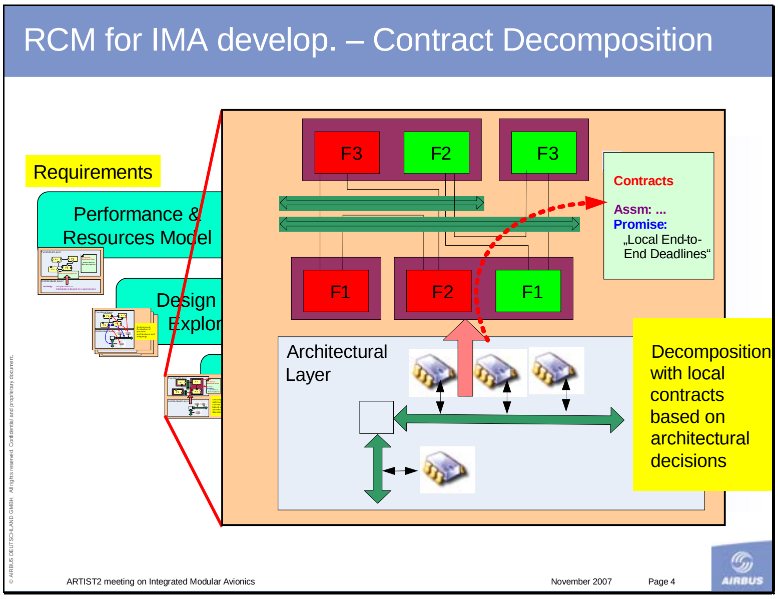
Analysis and Evaluation of possible architectures and mappings

ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 22



RCM for IMA develop. – Architecture Optimization

ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 23



RCM for IMA develop. – Contract Decomposition

Contracts

Assm: ...

Promise: „Local End-to-End Deadlines“

Decomposition with local contracts based on architectural decisions

ARTIST2 meeting on Integrated Modular Avionics November 2007 Page 4

From the Integrated Performance & Resources Model several architectural choices can be derived (Design Space Exploration). Evaluate and compare different IMA solutions wrt. development and life-cycle aspects; allows choosing actual IMA architecture and exploring it (not fixed in advance); finally, based on chosen architecture, decomposed contracts for the components are then submitted to the suppliers; finally, contracts serve, symmetrically, for the integration phase.

Documents exchanged with the suppliers will be supplemented with formal notations (models).

Discussion

Q: question on HRC, what it is, how you express contracts. ***A: HRC is a meta-model able to express contracts with their behavioral aspects, both functional and non-functional. It comes equipped with a strong mathematical semantics covering composition.***

Q: how do you address circularity in the assumption/guarantee reasoning, horizontal or vertical? ***A: doing iterations will solve this.***

Q: link planed btw TOPCASED and SPEEDS? ***A: yes, of course.***

Q: how close is the SPEEDS bus to the MODEL BUS concept of TOPCASED? Seems very similar in their objectives.

Roman Obermaisser, TU Vienna: Supporting Heterogeneous Applications in the DECOS Integrated Architecture

TU-Vienna. Presents results of DECOS project on integrated systems architectures. Airbus is also part of this project.

Federated and integrated architectures

Provide each application subsystem with it sown dedicated computer system

- natural separation of application subsystems
- service optimization
- fault isolation

Integrated architectures support multiple application subsystems within a single distributed computer system

- reduced HW cost
- dependability
- flexibility

Shift to integrated architectures: IMA, AUTOSAR, DECOS

Challenges:

- system complexity increases
- application complexity

- interferences between application subsystems (e.g., leading to invalidation of prior services)

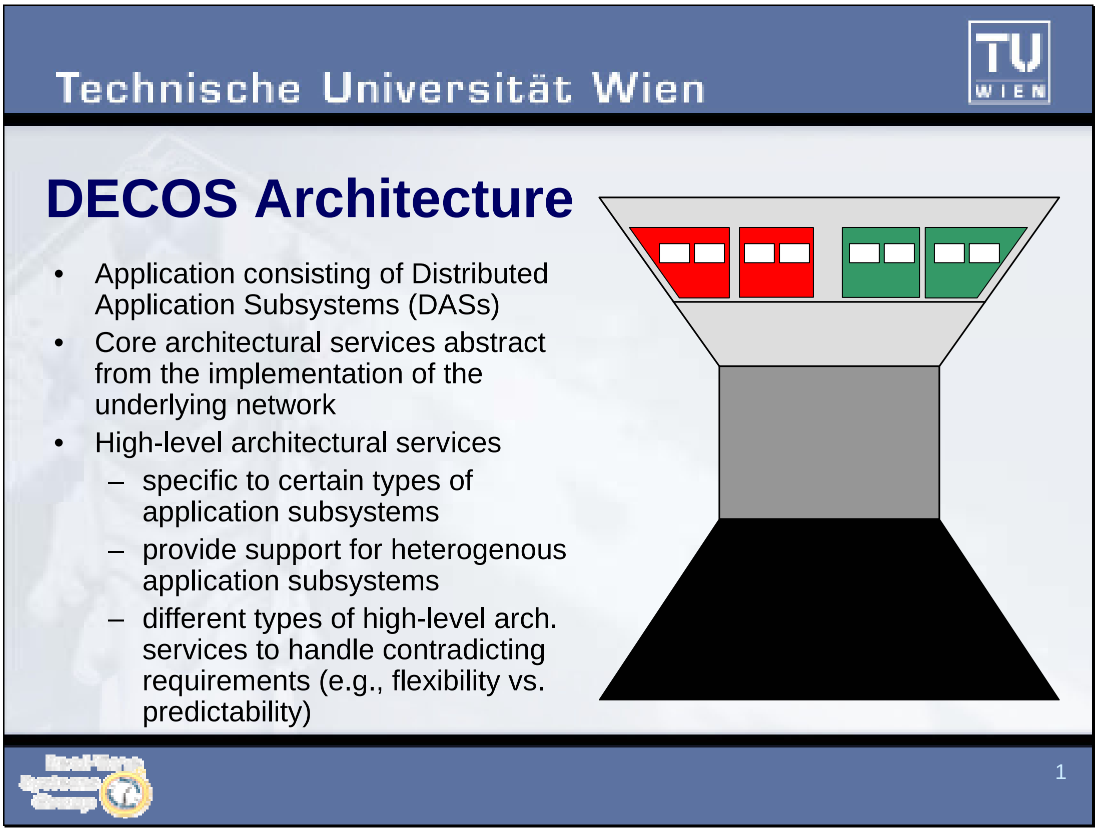
Answer is *temporal Composability.* Important: temporal correctness is not refuted by system integration.

Challenge (2)

- heterogeneous application subsystems (safety-critical and non-safety critical; different programming models, different platforms)
- divergent requirements concerning the services of the underlying platform (functionality, timing, QoS)

DECOS architecture

Time-Triggered architectures are widely accepted as communication infrastructures for safety-critical applications (aerospace, currently introduced in automobile); temporal predictability and fault-tolerance.



Technische Universität Wien

DECOS Architecture

- Application consisting of Distributed Application Subsystems (DASs)
- Core architectural services abstract from the implementation of the underlying network
- High-level architectural services
  - specific to certain types of application subsystems
  - provide support for heterogenous application subsystems
  - different types of high-level arch. services to handle contradicting requirements (e.g., flexibility vs. predictability)

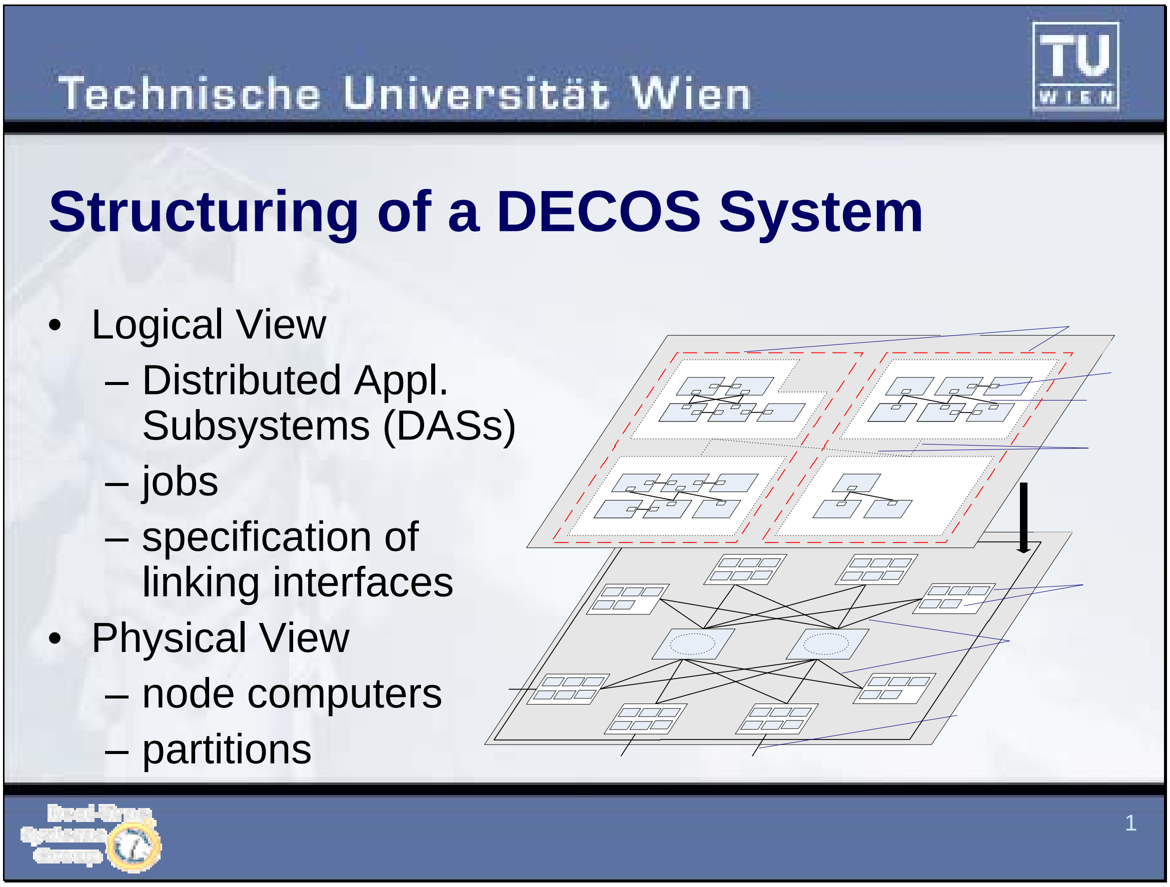
1

The layered nature allows supporting contradictory features for different uses.

A special OS has been developed that is fairly close to ARINC 653. Close OS can be used as well in combination with DECOS architecture.

Q: no sharing of resource? ***A: yes there is. But it is carefully handled to guarantee partitioning.***

Structuring of a DECOS system is shown in the figure below:



Technische Universität Wien

Structuring of a DECOS System

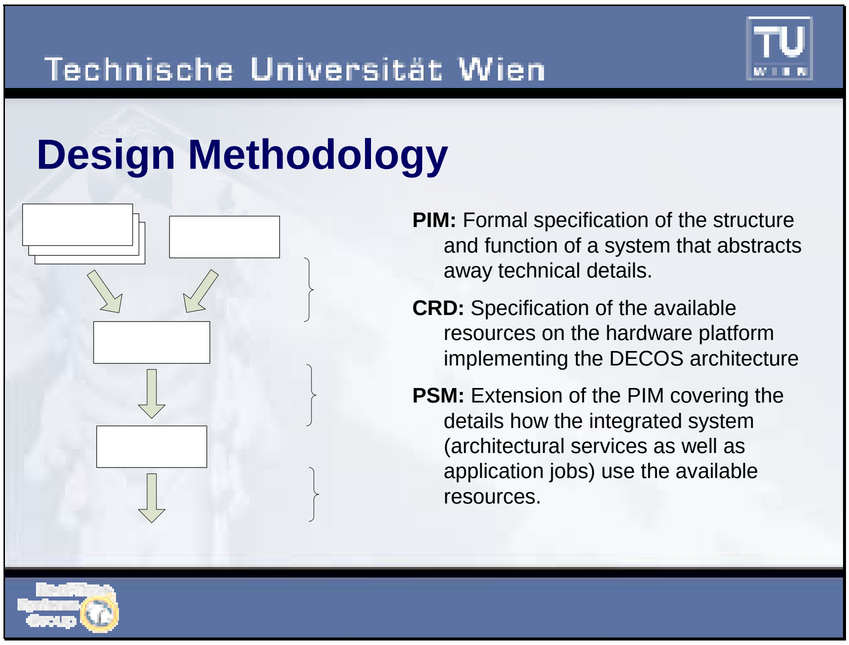
- Logical View
  - Distributed Appl. Subsystems (DASs)
  - jobs
  - specification of linking interfaces
- Physical View
  - node computers
  - partitions

1

Encapsulation:

- partitioning for computational and communication resources
- in value and time-domain

Model based development process



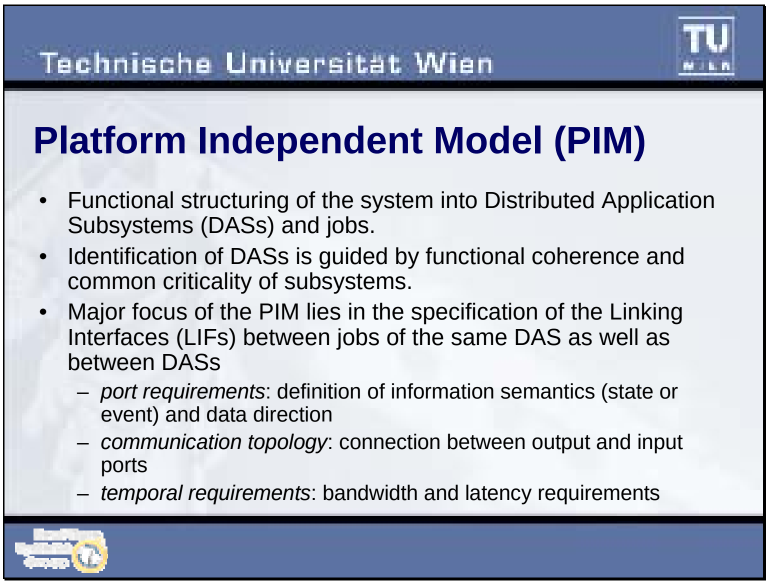
Technische Universität Wien

Design Methodology

PIM: Formal specification of the structure and function of a system that abstracts away technical details.

CRD: Specification of the available resources on the hardware platform implementing the DECOS architecture

PSM: Extension of the PIM covering the details how the integrated system (architectural services as well as application jobs) use the available resources.



Technische Universität Wien

Platform Independent Model (PIM)

- Functional structuring of the system into Distributed Application Subsystems (DASs) and jobs.
- Identification of DASs is guided by functional coherence and common criticality of subsystems.
- Major focus of the PIM lies in the specification of the Linking Interfaces (LIFs) between jobs of the same DAS as well as between DASs
  - *port requirements*: definition of information semantics (state or event) and data direction
  - *communication topology*: connection between output and input ports
  - *temporal requirements*: bandwidth and latency requirements

Platform-Independent
DAS Representation

Platform Independent
DAS Representation
(PIM)

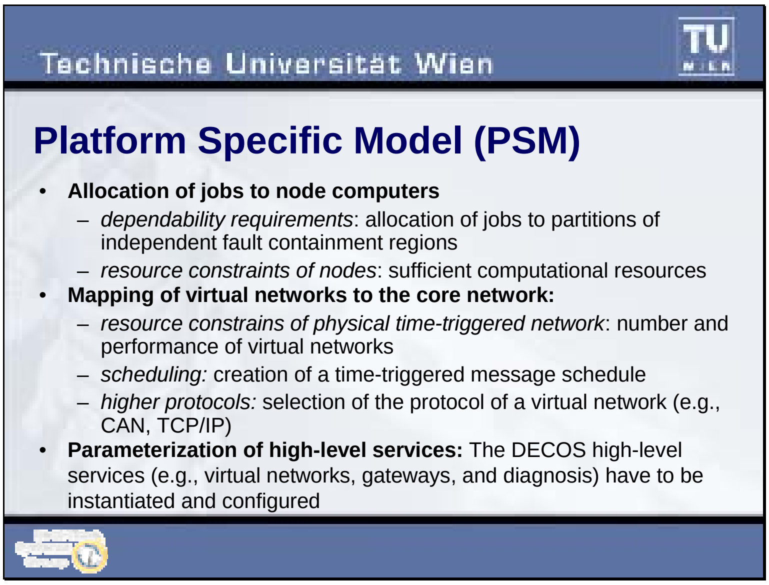
Platform Independent
DAS Representation

Resource Specification
(CRD)

Virtual
Integration

Platform-Specific
System Representation
(PSM)

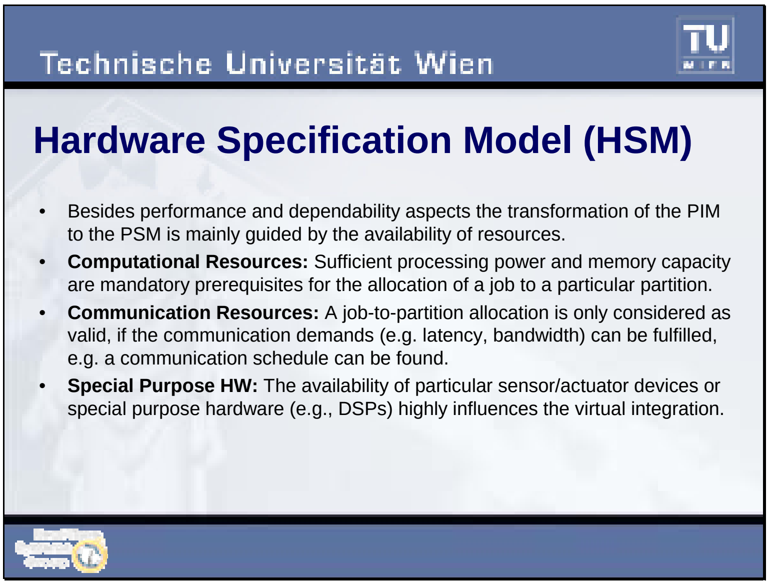
Code
Generation



Technische Universität Wien

Platform Specific Model (PSM)

- **Allocation of jobs to node computers**
  - *dependability requirements*: allocation of jobs to partitions of independent fault containment regions
  - *resource constraints of nodes*: sufficient computational resources
- **Mapping of virtual networks to the core network:**
  - *resource constrains of physical time-triggered network*: number and performance of virtual networks
  - *scheduling:* creation of a time-triggered message schedule
  - *higher protocols:* selection of the protocol of a virtual network (e.g., CAN, TCP/IP)
- **Parameterization of high-level services:** The DECOS high-level services (e.g., virtual networks, gateways, and diagnosis) have to be instantiated and configured



Technische Universität Wien

Hardware Specification Model (HSM)

- Besides performance and dependability aspects the transformation of the PIM to the PSM is mainly guided by the availability of resources.
- **Computational Resources:** Sufficient processing power and memory capacity are mandatory prerequisites for the allocation of a job to a particular partition.
- **Communication Resources:** A job-to-partition allocation is only considered as valid, if the communication demands (e.g. latency, bandwidth) can be fulfilled, e.g. a communication schedule can be found.
- **Special Purpose HW:** The availability of particular sensor/actuator devices or special purpose hardware (e.g., DSPs) highly influences the virtual integration.

Notes: scheduling is a central activity to ensure and drive the mapping; TDMA multiplexing. This methodology puts emphasis on timing whatever the nature of the application is.

Note: the PIM model is not executable per se. But it can be modeled, e.g., using SCADE. Other programming models could be used as well.

Q: why not reusing existing architecture formalisms or languages? ***A: DECOS started earlier and has considered specific issues.***

Architeectural services

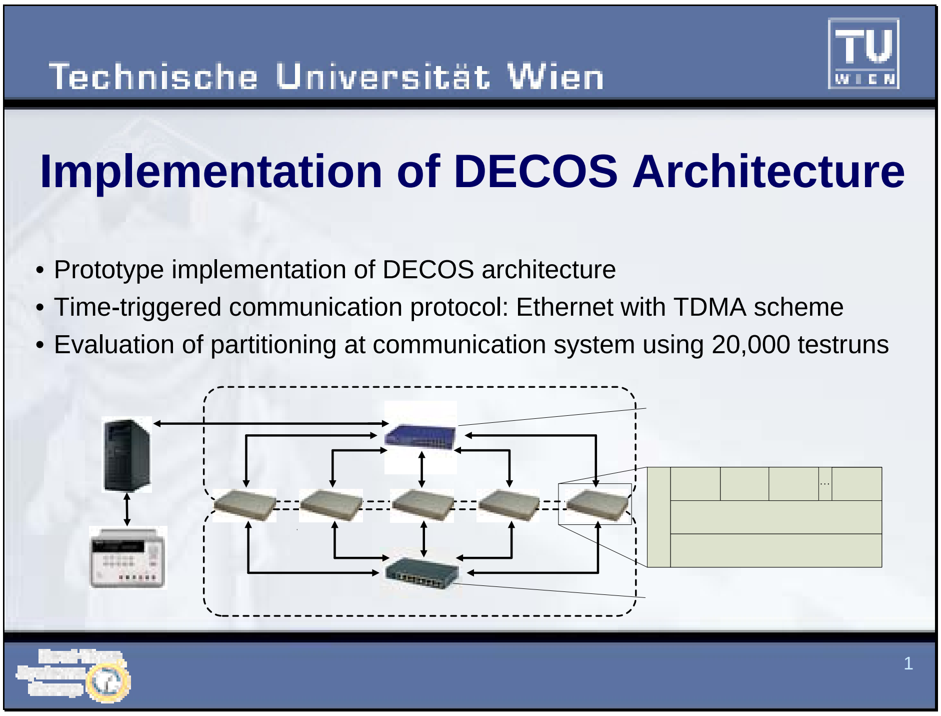
- Virtual networks: overlay network on top of TT physical network, e.g., to offer E.T. services. Partitioning of communication resources is performed using TDMA to share the different levels of criticality. Realised:
  - Virtual CAN
  - TT virtual network

  - Virtual TCP-IP
  - Virtual network with CORBA

  Looks like an expensive way of emulating cheap networks. May seem foolish but may greatly simplify the overall architecture when safety critical is dominant.

- Diagnosis is based on maintenance-oriented fault models; internal borderline, non-destructive external. Jobs are used as units of update for SW faults. Based on out-of-norm assertions, i.e., behaviors that cannot be classified as correct/faulty but are suspicious. Analysis is reinforced by performing correlation; correlation is facilitated by time stamping provided by TTA. Heterogeneous applications can be supported for out-of-norm behavior; for event-triggered applications, more difficult since timing information is uncertain. This analysis is built on timed automata framework. These models are generated automatically; in the future more intuitive ways of generating them is planed.

Implementation and sresults



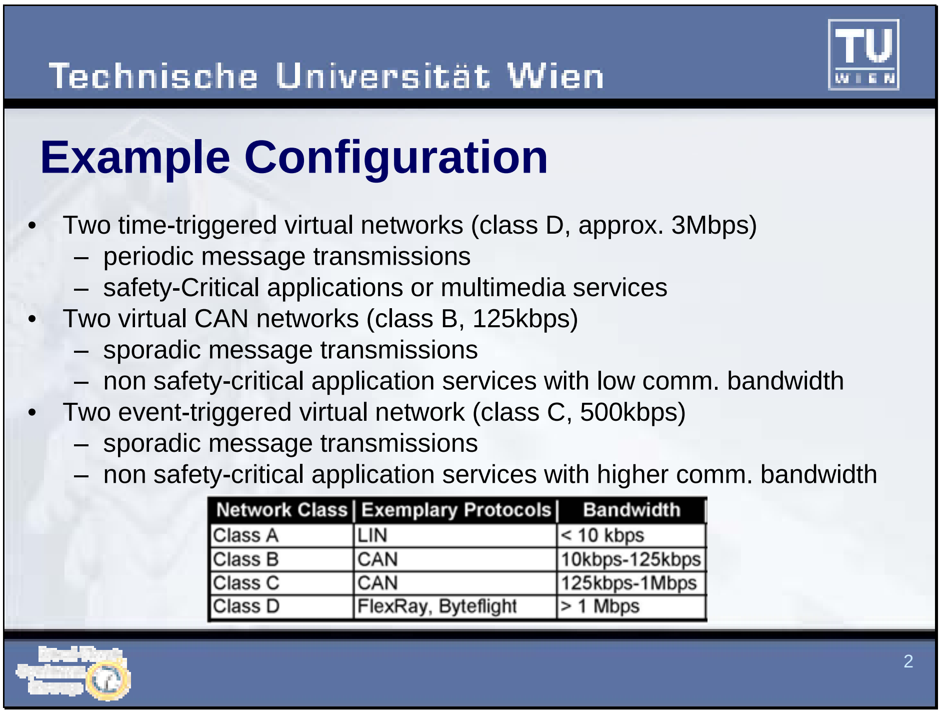
Technische Universität Wien

TU WIEN

Implementation of DECOS Architecture

- Prototype implementation of DECOS architecture
- Time-triggered communication protocol: Ethernet with TDMA scheme
- Evaluation of partitioning at communication system using 20,000 testruns

1



Technische Universität Wien

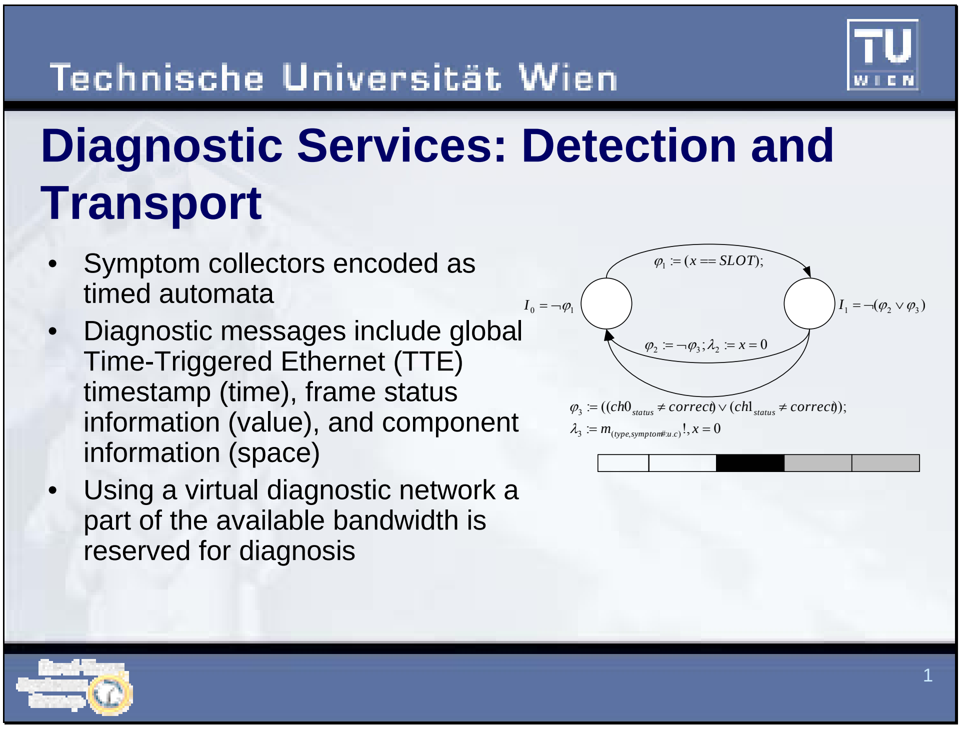
Example Configuration

- Two time-triggered virtual networks (class D, approx. 3Mbps)
  - periodic message transmissions
  - safety-Critical applications or multimedia services
- Two virtual CAN networks (class B, 125kbps)
  - sporadic message transmissions
  - non safety-critical application services with low comm. bandwidth
- Two event-triggered virtual network (class C, 500kbps)
  - sporadic message transmissions
  - non safety-critical application services with higher comm. bandwidth

Network Class	Exemplary Protocols	Bandwidth
Class A	LIN	< 10 kbps
Class B	CAN	10kbps-125kbps
Class C	CAN	125kbps-1Mbps
Class D	FlexRay, Byteflight	> 1 Mbps

2

Sporadic and periodic messages have been generated for this implementation. Corresponding synthesized benchmarks have been obtained, for latency and bandwidth.



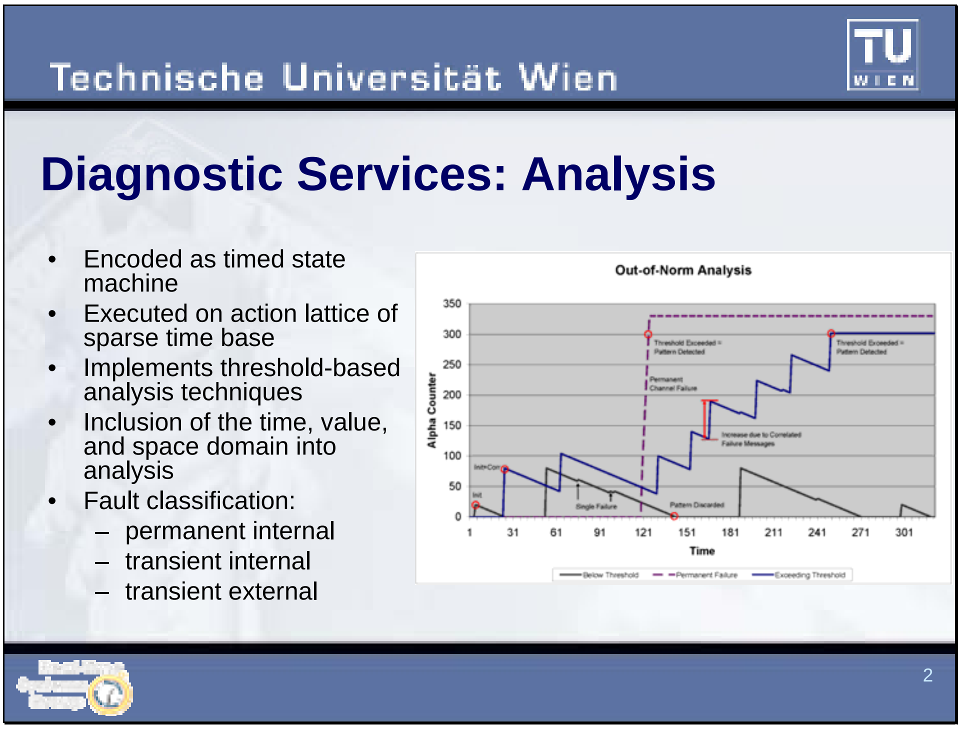
Technische Universität Wien

Diagnostic Services: Detection and Transport

- Symptom collectors encoded as timed automata
- Diagnostic messages include global Time-Triggered Ethernet (TTE) timestamp (time), frame status information (value), and component information (space)
- Using a virtual diagnostic network a part of the available bandwidth is reserved for diagnosis

$\varphi_1 := (x == SLOT);$
$I_0 = \neg\varphi_1$
$I_1 = \neg(\varphi_2 \vee \varphi_3)$
$\varphi_2 := \neg\varphi_3; \lambda_2 := x = 0$
$\varphi_3 := ((ch0_{status} \neq correct) \vee (ch1_{status} \neq correct));$
$\lambda_3 := m_{(type,symptom\#:u.c)}!, x = 0$

1



Technische Universität Wien

Diagnostic Services: Analysis

- Encoded as timed state machine
- Executed on action lattice of sparse time base
- Implements threshold-based analysis techniques
- Inclusion of the time, value, and space domain into analysis
- Fault classification:
  - permanent internal
  - transient internal
  - transient external

Out-of-Norm Analysis
Alpha Counter
350
300
250
200
150
100
50
0
1
31
61
91
121
151
181
211
241
271
301
Time
Threshold Exceeded = Pattern Detected
Threshold Exceeded = Pattern Detected
Permanent Channel Failure
Increase due to Correlated Failure Messages
Init+Corr
Init
Single Failure
Pattern Discarded
Below Threshold
Permanent Failure
Exceeding Threshold

2

Discussion

Q: which property do you guarantee by construction, and which temporal property you address, can they be invalidated by additional jobs? ***A: very robust to introduction of new jobs. Take the strong properties of DECOS architectures to facilitate proofs. These assumptions simplify proofs, they do not replace them.***

The results of DECOS project could be taken as an input to the evolution of IMA. The use of TTN might be an important input to IMA.

Q: what is the typical size of application tested? ***A: there were in part synthetic benchmarks. But there were also demonstrators in avionics and automotive and industrial control systems.***

Q: do you plan to standardize something? ***A: some partners are involved in it for side activities (e.g., AUTOSAR). There is a meta-model covering part of the architecture.***

Q: are you moving to real applications? ***A: the project will end with this year. There will be a three real-world examples using this technology. Then will come exploitation, after the project, with industrial partners.***

Kevin Driscoll, Honeywell: Honeywell requirements for IMA

KD is one of the pioneers of IMA and a key architect in the 1st airplane using IMA.

Federated vs Integrated architecture

What is architecture versus design?

Design has 3 orthogonal aspects

- behavioral: how the system does its functions
- logical: processors, ALU, memory, logic functions, bit storage
- physical: boxes, chips, wires, sensors, actuators, aircraft…

Federated architecture

Advantages

- Function has guaranteed access to processor
- deterministic and guaranteed access to I/O, controller latency & jitter
- critical function separation; low criticality functions can't corrupt critical functions (loosely coupled separate LRUs)

Drawbacks

- all functions sharing a processor must share highest level of criticality; encourages many processors
- inefficient in use of resources, multiple power supplies; single processors are oversized w.r.t 1 function
- proliferation of part types
- increased weight, power, wiring

Why integrate

Goal: an architecture that provides all the benefits of a federated architecture and solves the problems

Requirements beyond designer's experience

- 10^6 or 10^10 is much larger than what a designer experiences over a lifetime (10^4 hours) ; these probabilities are rather abstract; little intuitive support

How systems fail

Assumed importance order:

1. exhaustion of resources
2. single point of failure
3. chain or domino effect

In reality, the order is inverted!! 3, 2, 1.

Determinism

- Determinism is the characteristic of a system which allows the correct prediction of its future behavior given its current state and knowledge of future changes to its environment (inputs)

Dangers of mixed criticality

What could a non-critical function do to a critical one?

- erroneously write data into wrong areas
- steal time / interrupt processor
- crash the processor
  - halt and smoke

    - thermal slow down, missed deadline, jitter
    - thermally increased errors
- corrupt I/O
    - falsely send output data appearing to come from critical function
    - corrupt data before critical function uses it

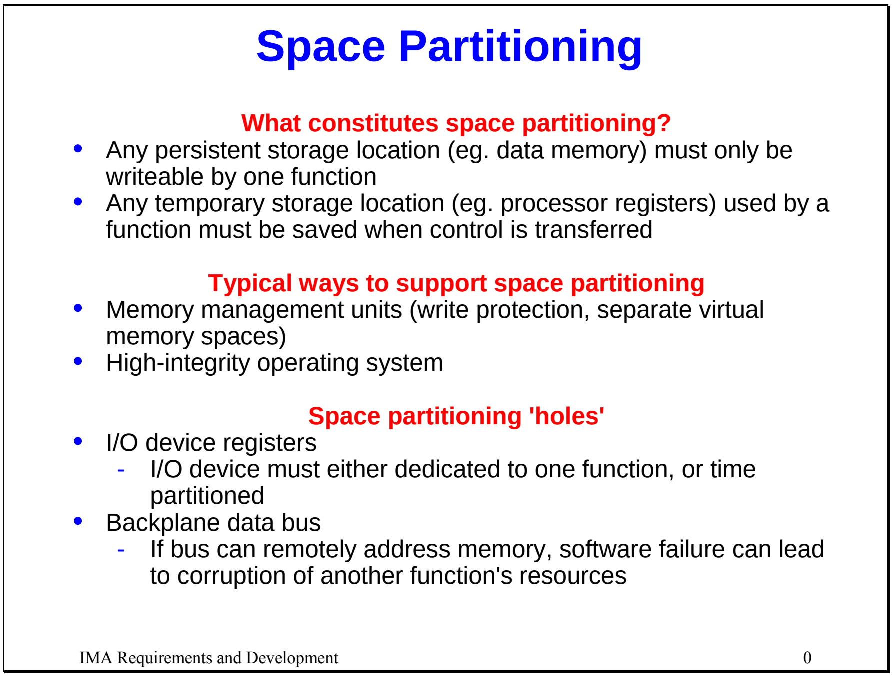
In addition, radiations are becoming an increasing problem.

IMA requirements

IMA partitioning requirements

- integrating multiple functions into shared HW opens the potential that failure in a function propagates to another function
- ARINC 651 requires *robust partitioning*
    - can support mixed levels of criticality on shared HW (lower certification & maintenance costs)
    - fault containment
- partitioning is required in 2 dimensions (time and space)

Space and time partitioning is illustrated on the following two figures.



Space Partitioning

What constitutes space partitioning?

- Any persistent storage location (eg. data memory) must only be writeable by one function
- Any temporary storage location (eg. processor registers) used by a function must be saved when control is transferred

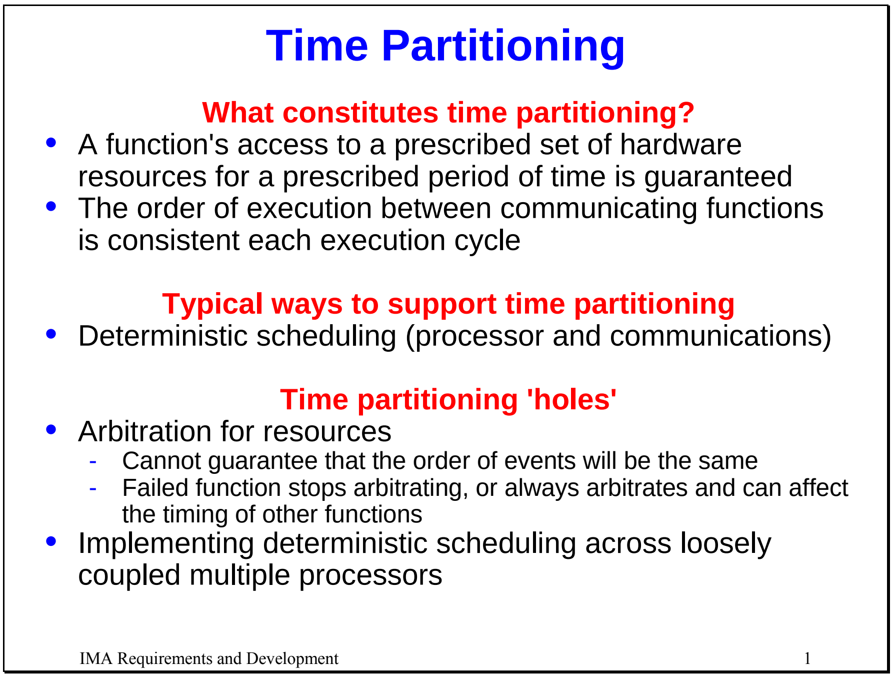
Typical ways to support space partitioning

- Memory management units (write protection, separate virtual memory spaces)
- High-integrity operating system

Space partitioning 'holes'

- I/O device registers
    - I/O device must either dedicated to one function, or time partitioned
- Backplane data bus
    - If bus can remotely address memory, software failure can lead to corruption of another function's resources

IMA Requirements and Development 0



Time Partitioning

What constitutes time partitioning?

- A function's access to a prescribed set of hardware resources for a prescribed period of time is guaranteed
- The order of execution between communicating functions is consistent each execution cycle

Typical ways to support time partitioning

- Deterministic scheduling (processor and communications)

Time partitioning 'holes'

- Arbitration for resources
  - Cannot guarantee that the order of events will be the same
  - Failed function stops arbitrating, or always arbitrates and can affect the timing of other functions
- Implementing deterministic scheduling across loosely coupled multiple processors

IMA Requirements and Development 1

Summary of IMA requirements

Integrated Modular Avionics Requirements

Robust Partitioning
- No failure in a function can cause another function to fail

Reliability
- Aggressive no-maintenance policy requiring high dispatch probability forces high-reliability on all components

Fault Tolerance
- No single failure can cause loss of critical or essential avionics functions
- Dispatch with failed components to meet dispatch probability requirements

Flexibility
- Must support many module types of differing capability
- Must allow for addition/modification of cabinet functions **without** forcing recertification of unmodified functions

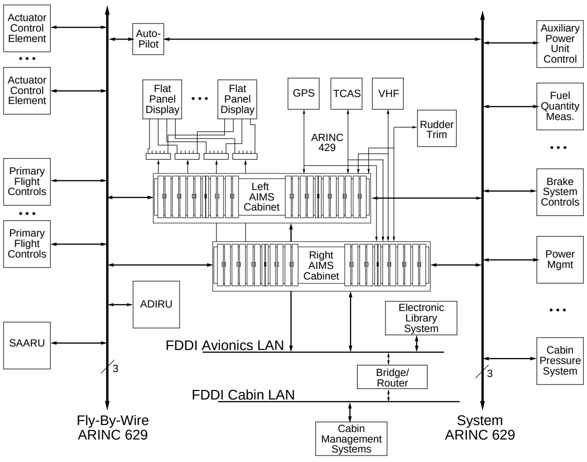
Easy to Debug and Certify
- Control of hardware/software integration
- Predictable behavior over all possible operating conditions

IMA Requirements and Development 0

An important point is to support incremental certification.

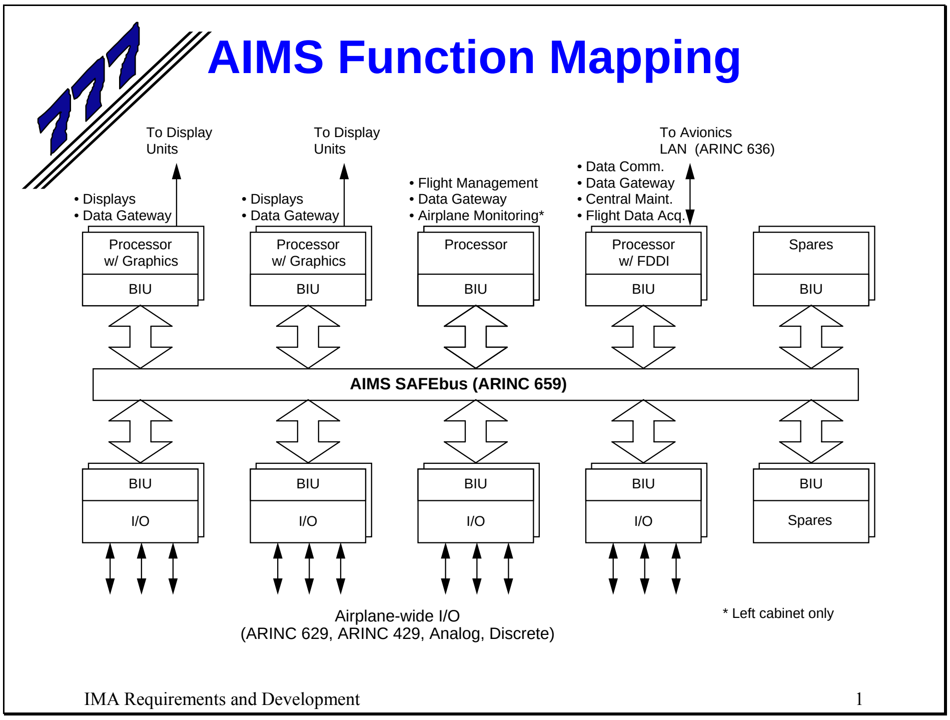
Boeing 777 avionics architecture: a 1st step in IMA

Boeing 777 Avionics Architecture



- A hybrid between a fully integrated and fully federated architecture
- A number of formerly federated functions integrated into AIMS
- Remaining units are assigned to one of three federated subsystems
  - Fly-by-wire (triple ARINC 629 bus connection)
  - System (triple ARINC 629 bus connection)
  - OLAN (ARINC 636, FDDI ring connection)
- AIMS acts as a gateway among these federated subsystems and other I/O
- Units replicated for safety and deferred maintenance

IMA Requirements and Development 0



AIMS Function Mapping

777

To Display Units

To Display Units

To Avionics LAN (ARINC 636)

- Displays
- Data Gateway

- Displays
- Data Gateway

- Flight Management
- Data Gateway
- Airplane Monitoring*

- Data Comm.
- Data Gateway
- Central Maint.
- Flight Data Acq.

Processor w/ Graphics | BIU

Processor w/ Graphics | BIU

Processor | BIU

Processor w/ FDDI | BIU

Spares | BIU

AIMS SAFEbus (ARINC 659)

BIU | I/O

BIU | I/O

BIU | I/O

BIU | I/O

BIU | Spares

Airplane-wide I/O
(ARINC 629, ARINC 429, Analog, Discrete)

* Left cabinet only

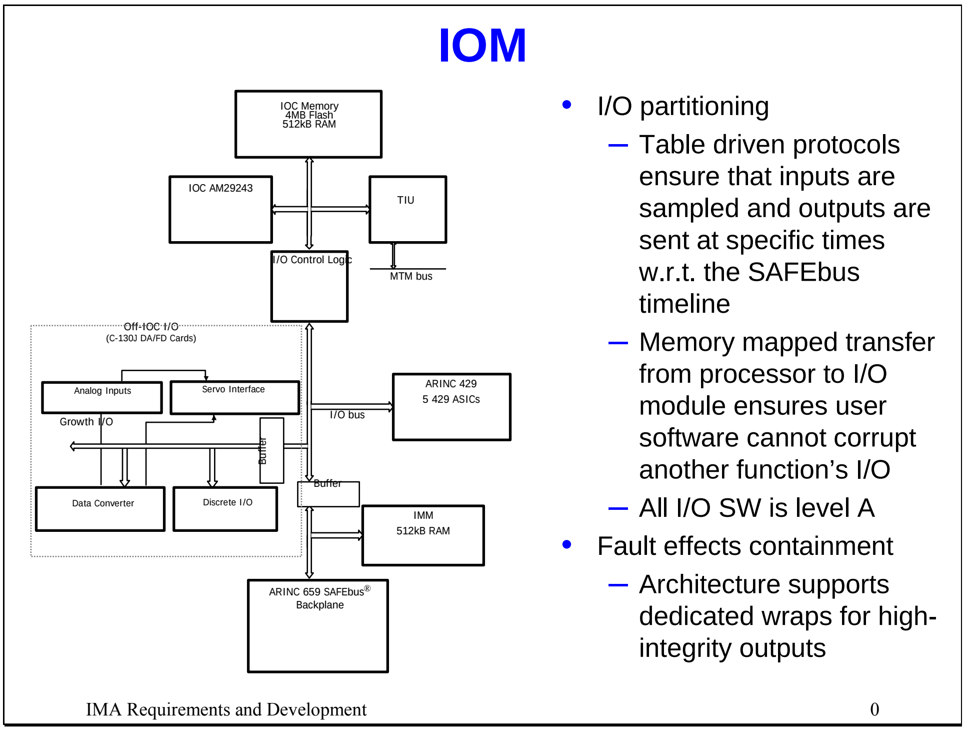
IMA Requirements and Development 1

Q: how was the choice made of what to integrate and what not to? ***A: a matter of experience, not really formalized…***

Key concepts

- SAFEbus backplane; the bus is the time master for all processors; protocol time, not physical time, is used globally
- dual lock-step processor modules
- all modules synchronize to safebus
- memory management
- OS
- I/O

Progress vs. effort in using synchrony vs. asynchrony. Asynchrony looks less costly at the beginning, but this will changes in the long range. When clocks are OK the rest and integration gets much quicker.



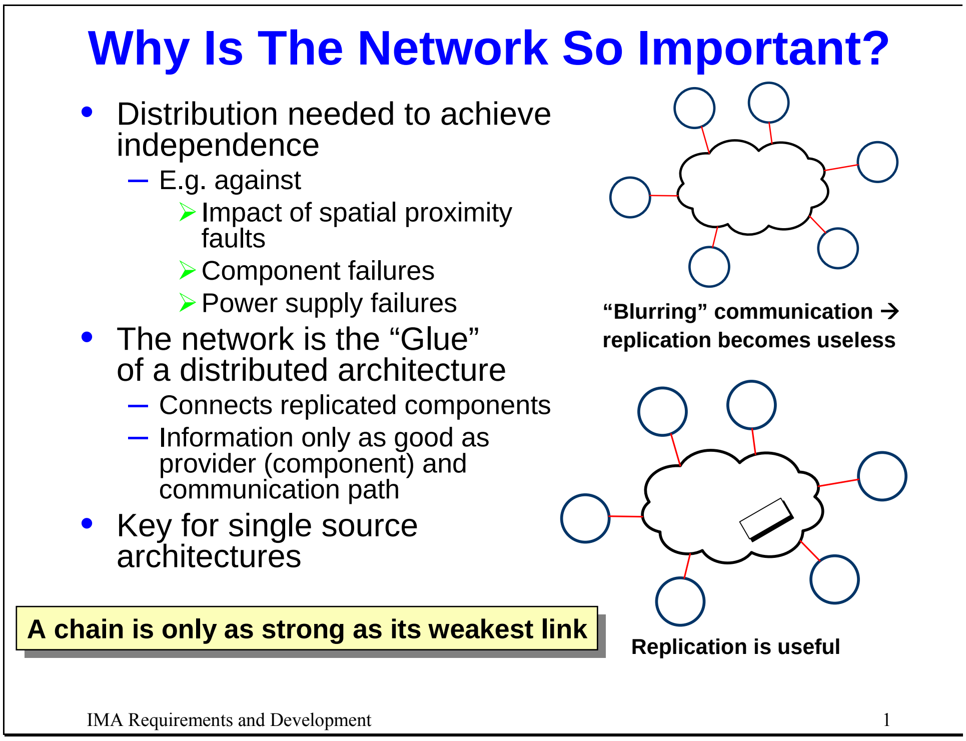
IOM

- I/O partitioning
  - Table driven protocols ensure that inputs are sampled and outputs are sent at specific times w.r.t. the SAFEbus timeline
  - Memory mapped transfer from processor to I/O module ensures user software cannot corrupt another function's I/O
  - All I/O SW is level A
- Fault effects containment
  - Architecture supports dedicated wraps for high-integrity outputs

IMA Requirements and Development 0

Network and bus: SAFEbus

The importance of network and SAFEbus is explained in the following figures.



Why Is The Network So Important?

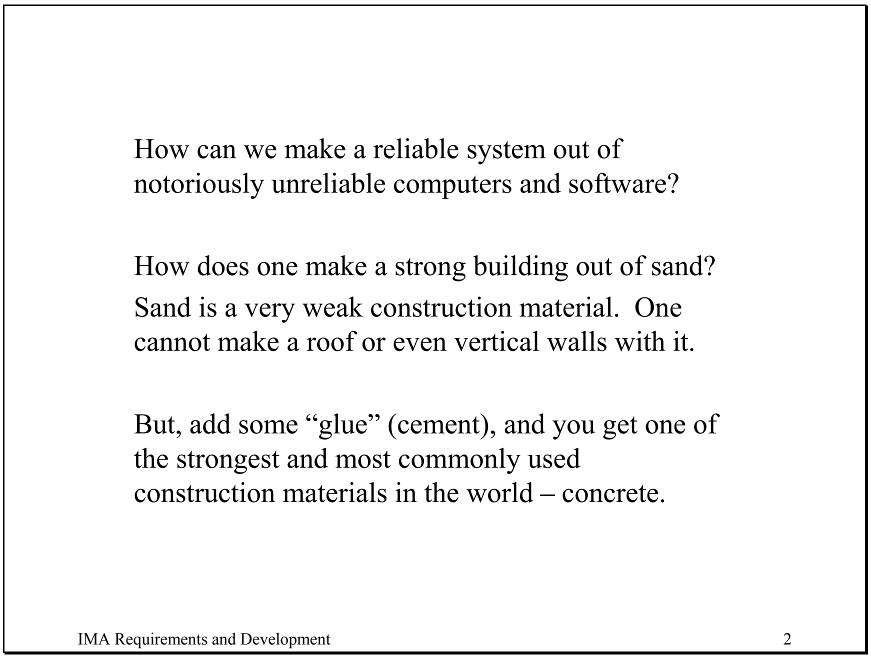
- Distribution needed to achieve independence
  - E.g. against
    - ➢ Impact of spatial proximity faults
    - ➢ Component failures
    - ➢ Power supply failures
- The network is the "Glue" of a distributed architecture
  - Connects replicated components
  - Information only as good as provider (component) and communication path
- Key for single source architectures

A chain is only as strong as its weakest link

"Blurring" communication → replication becomes useless

Replication is useful

IMA Requirements and Development 1

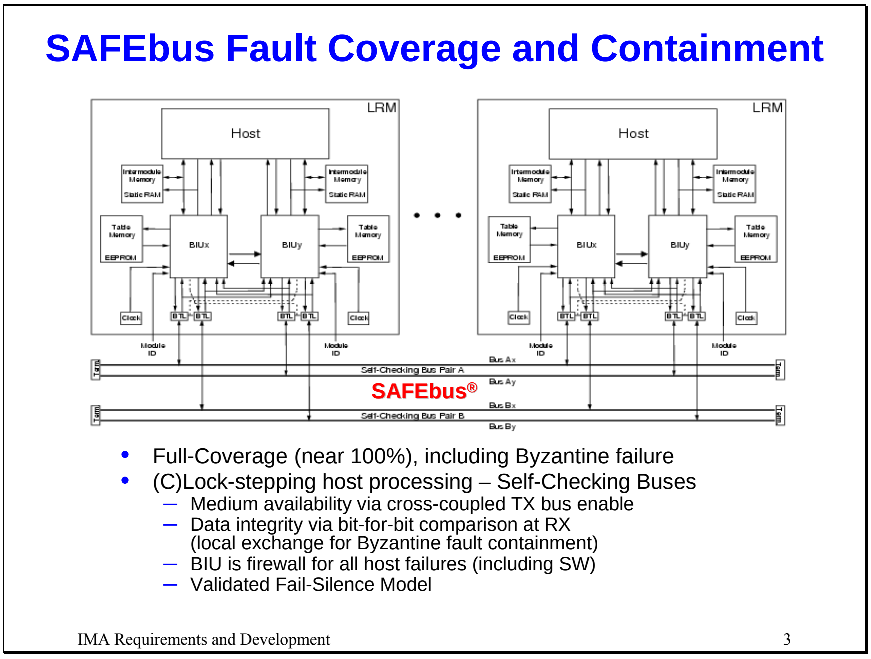


How can we make a reliable system out of notoriously unreliable computers and software?

How does one make a strong building out of sand? Sand is a very weak construction material. One cannot make a roof or even vertical walls with it.

But, add some “glue” (cement), and you get one of the strongest and most commonly used construction materials in the world – concrete.

IMA Requirements and Development 2

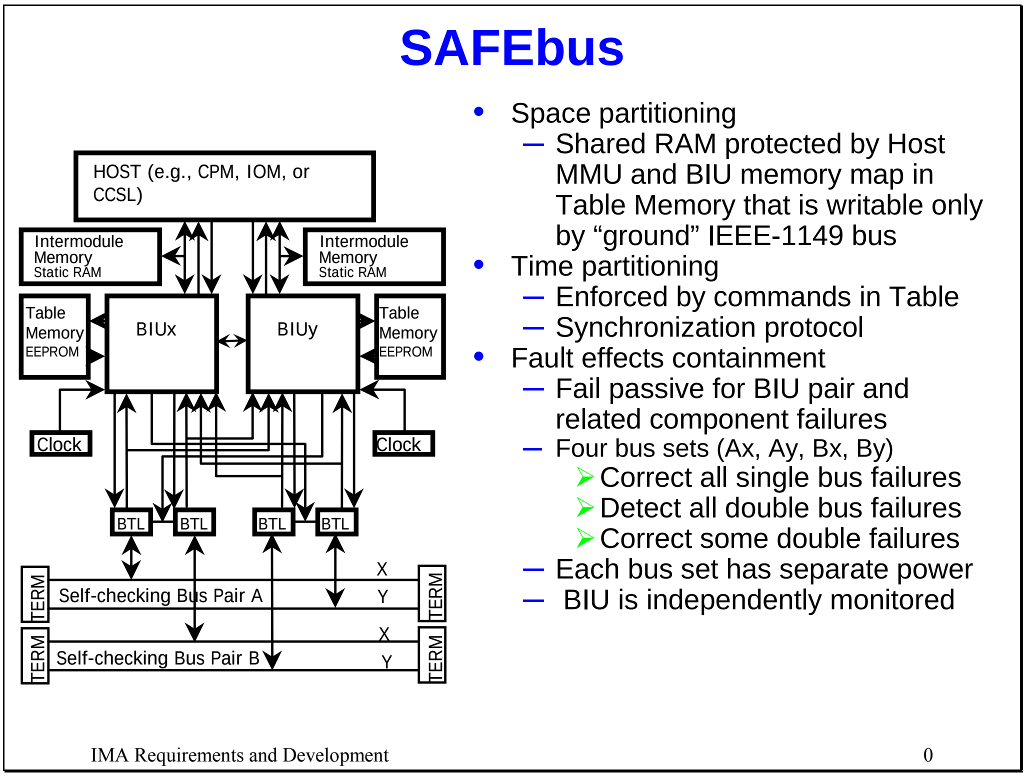


SAFEbus Fault Coverage and Containment

- Full-Coverage (near 100%), including Byzantine failure
- (C)Lock-stepping host processing – Self-Checking Buses
  - Medium availability via cross-coupled TX bus enable
  - Data integrity via bit-for-bit comparison at RX (local exchange for Byzantine fault containment)
  - BIU is firewall for all host failures (including SW)
  - Validated Fail-Silence Model

IMA Requirements and Development 3

SAFEbus implements space and time partitioning



SAFEbus

- Space partitioning
  - Shared RAM protected by Host MMU and BIU memory map in Table Memory that is writable only by “ground” IEEE-1149 bus
- Time partitioning
  - Enforced by commands in Table
  - Synchronization protocol
- Fault effects containment
  - Fail passive for BIU pair and related component failures
  - Four bus sets (Ax, Ay, Bx, By)
    - Correct all single bus failures
    - Detect all double bus failures
    - Correct some double failures
  - Each bus set has separate power
  - BIU is independently monitored

IMA Requirements and Development 0

No cache is used.

What does dual redundancy provide? (dual redundancy is the cheapest safety mechanism)

- availability: readiness for correct service; if miss-compare pick one of them; not safe
- integrity: if miss-compare reject both; this is safe

Lock-step processor architecture

- space partitioning
- time partitioning: no other interrupt other than by the SAFEbus and Fatal (processor does not return from Fatal interrupt).
- fault effect containment

Memory management

- functional dataflow is the model of inter-partition communication

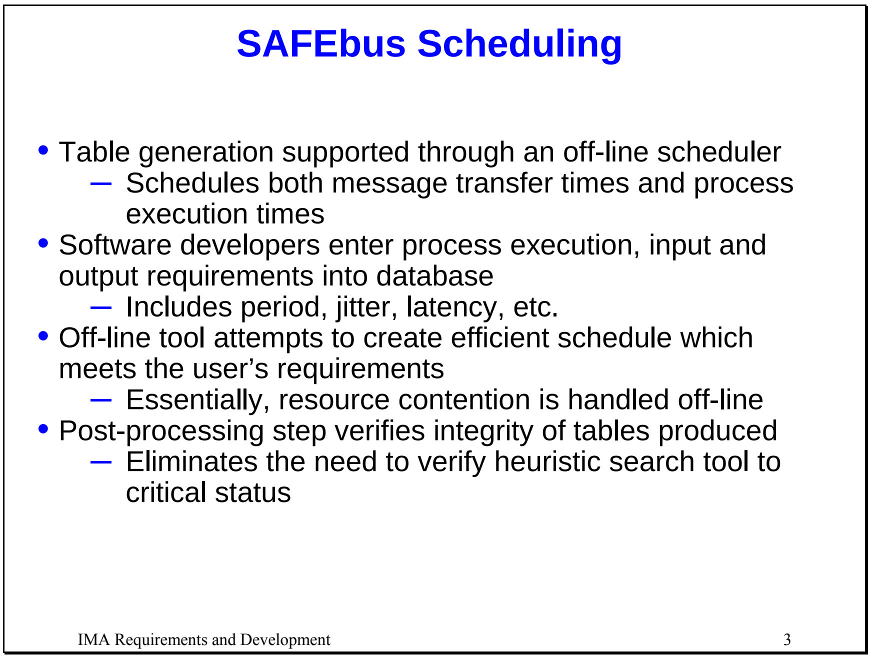
System scheduling enforces space (MMU and page tables) and time partitioning (response to SAFEbus time interrupts). There is multi-threading and preemption but statically defined at design time and checked by a tool. Fault effect containment by detecting deadline failures, responding to partition attempts to write to illegal locations, rapid restarting after power failure…

System architecture should guarantee that faults are easily distinguished regarding their origin (computing platform or application & plant)

Processor fault recovery, transparent re-try (however, bounded), restart/shutdown partition or CPM; fault recovery is dependent upon SW executing during fault occurrence

IOM. I/O partitioning, see SLIDE 25 (include or copy) All I/.O SW are level A.

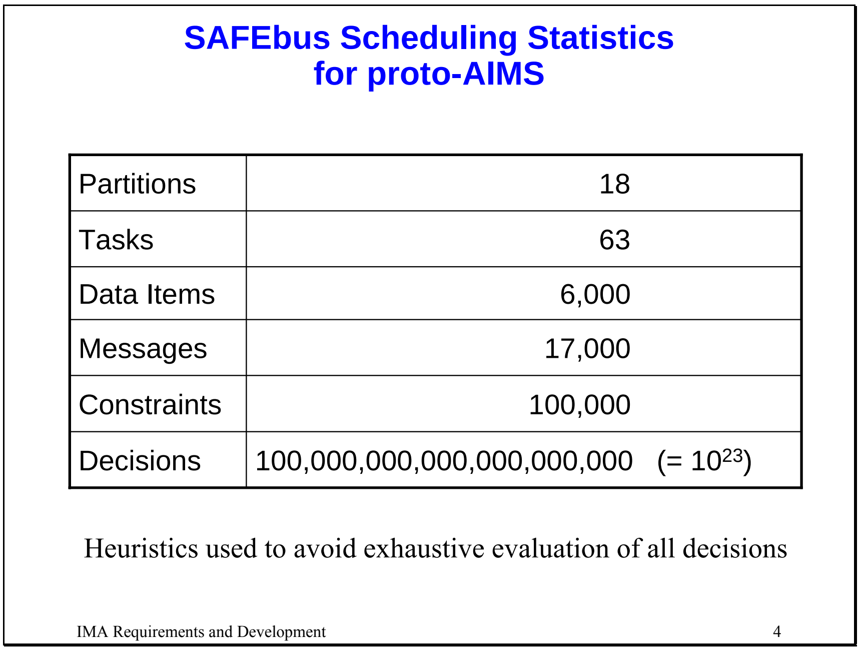
SAFEbus scheduling is an important effort, as explained in the following figures.



SAFEbus Scheduling

- Table generation supported through an off-line scheduler
  - Schedules both message transfer times and process execution times
- Software developers enter process execution, input and output requirements into database
  - Includes period, jitter, latency, etc.
- Off-line tool attempts to create efficient schedule which meets the user's requirements
  - Essentially, resource contention is handled off-line
- Post-processing step verifies integrity of tables produced
  - Eliminates the need to verify heuristic search tool to critical status

IMA Requirements and Development 3

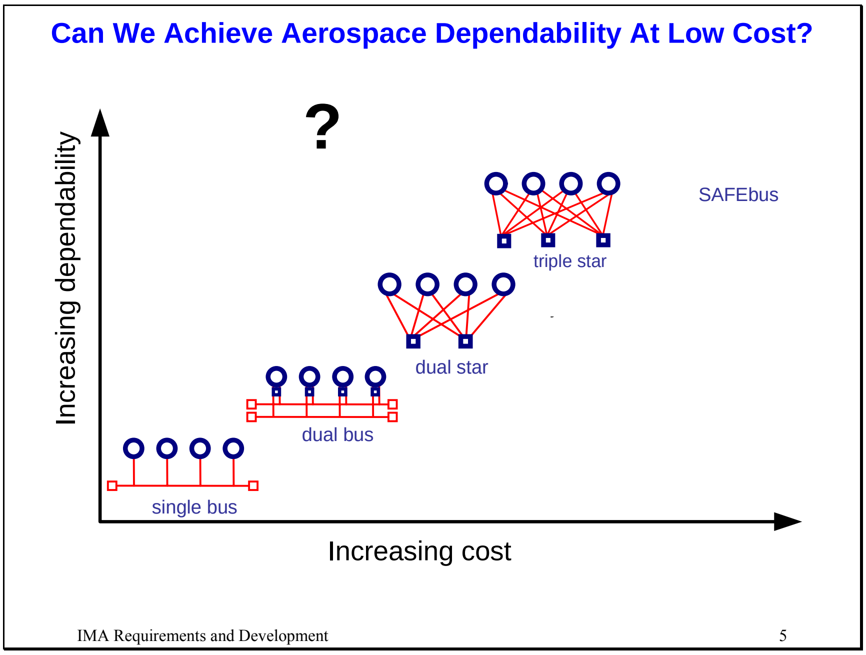


SAFEbus Scheduling Statistics for proto-AIMS

Partitions	18
Tasks	63
Data Items	6,000
Messages	17,000
Constraints	100,000
Decisions	100,000,000,000,000,000,000 (= 10^{23})

Heuristics used to avoid exhaustive evaluation of all decisions

IMA Requirements and Development 4

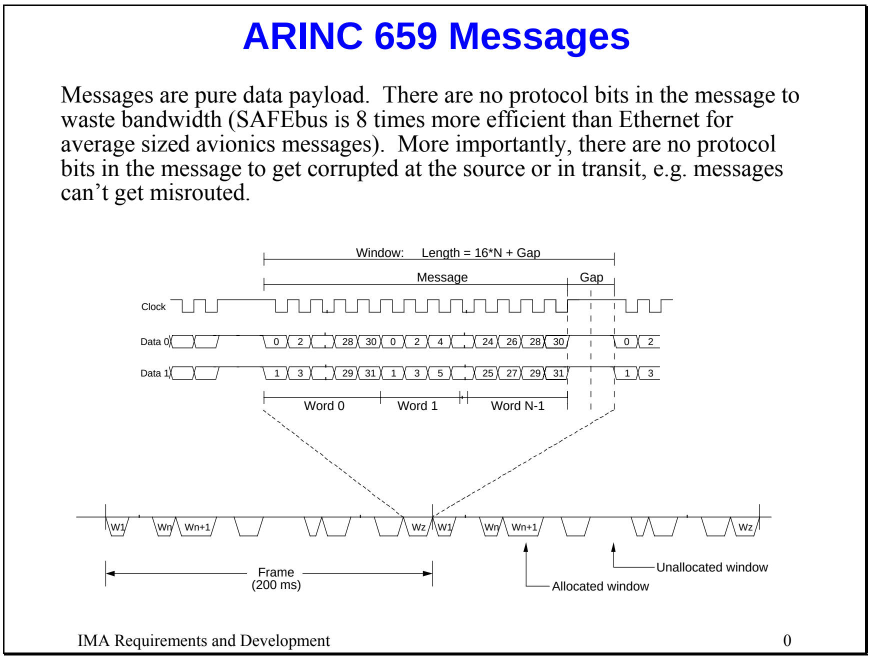


Can We Achieve Aerospace Dependability At Low Cost?

IMA Requirements and Development 5

ARINC 659

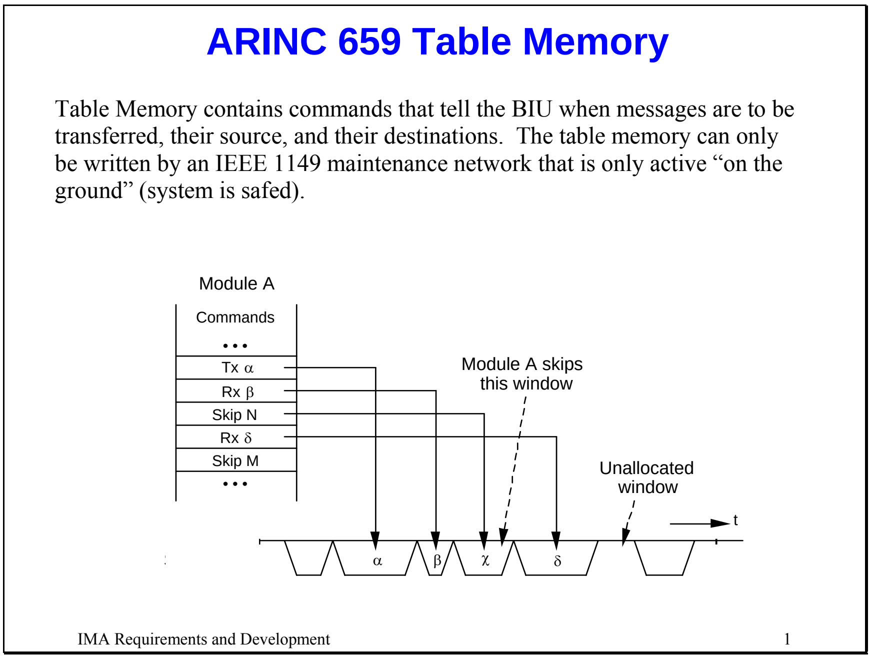
ARINC 659 is illustrated and explained in the figures below.



ARINC 659 Messages

Messages are pure data payload. There are no protocol bits in the message to waste bandwidth (SAFEbus is 8 times more efficient than Ethernet for average sized avionics messages). More importantly, there are no protocol bits in the message to get corrupted at the source or in transit, e.g. messages can't get misrouted.

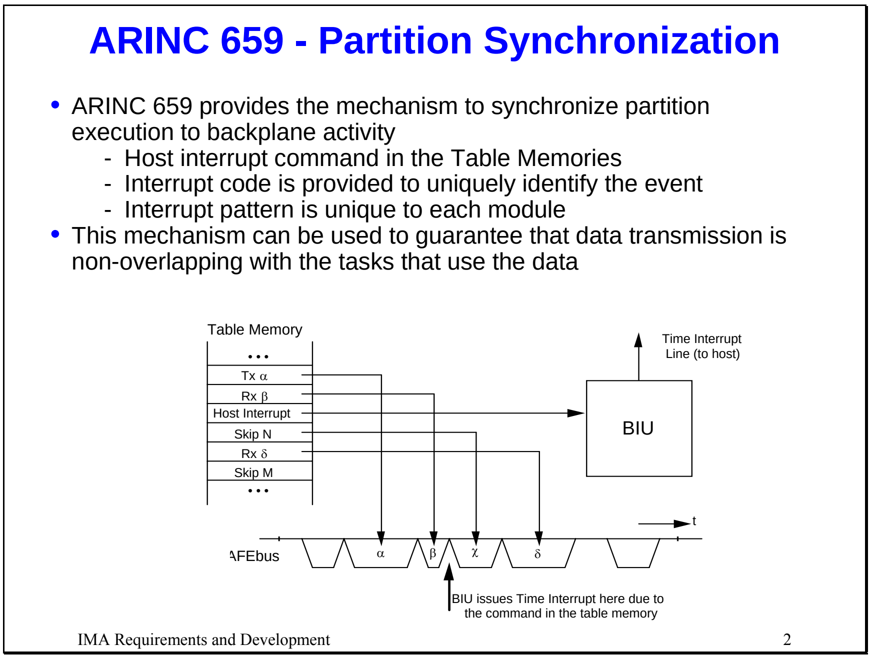
IMA Requirements and Development 0



ARINC 659 Table Memory

Table Memory contains commands that tell the BIU when messages are to be transferred, their source, and their destinations. The table memory can only be written by an IEEE 1149 maintenance network that is only active "on the ground" (system is safed).

IMA Requirements and Development 1



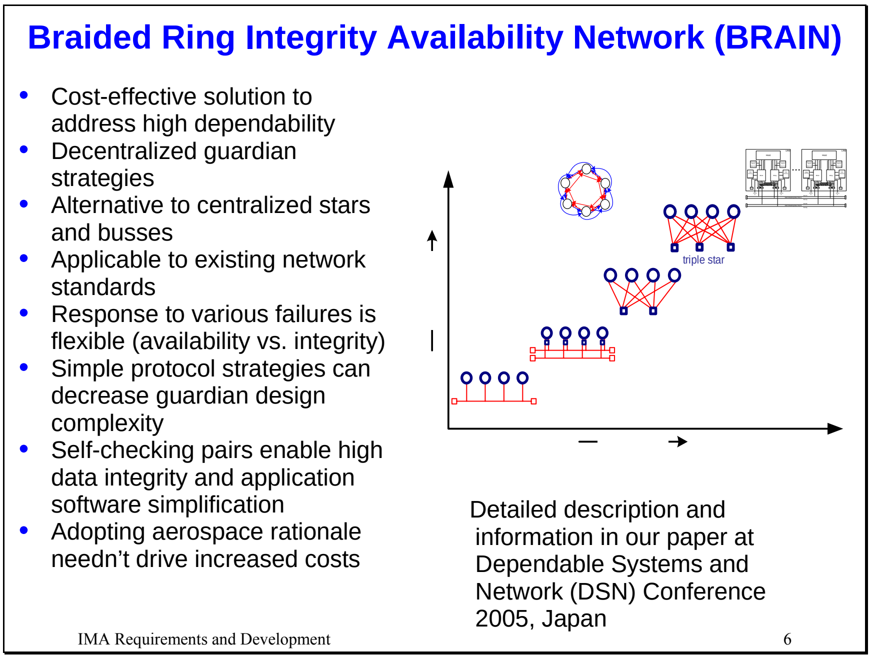
ARINC 659 - Partition Synchronization

- ARINC 659 provides the mechanism to synchronize partition execution to backplane activity
  - Host interrupt command in the Table Memories
  - Interrupt code is provided to uniquely identify the event
  - Interrupt pattern is unique to each module
- This mechanism can be used to guarantee that data transmission is non-overlapping with the tasks that use the data

Table Memory
• • •
Tx α
Rx β
Host Interrupt
Skip N
Rx δ
Skip M
• • •
Time Interrupt Line (to host)
BIU
t
AFEbus
α
β
χ
δ
BIU issues Time Interrupt here due to the command in the table memory

IMA Requirements and Development 2

A future project



Braided Ring Integrity Availability Network (BRAIN)

- Cost-effective solution to address high dependability
- Decentralized guardian strategies
- Alternative to centralized stars and busses
- Applicable to existing network standards
- Response to various failures is flexible (availability vs. integrity)
- Simple protocol strategies can decrease guardian design complexity
- Self-checking pairs enable high data integrity and application software simplification
- Adopting aerospace rationale needn't drive increased costs

Detailed description and information in our paper at Dependable Systems and Network (DSN) Conference 2005, Japan

IMA Requirements and Development 6

Discussion

Q: how can you afford having this customized architecture considering costs? ***A: where do you think the cost is? COTS processors are the cost. Now, lock-step, not clock-step is used (not oscillator time, just a global logical tick no more precise than is needed to ensure bit-identical outputs).***

Different architecture for lower end aircrafts; partitioning and redundancy at coarser grain.

Alex Wilson, Wind River: The evolving ARINC 653 standard and its application to IMA

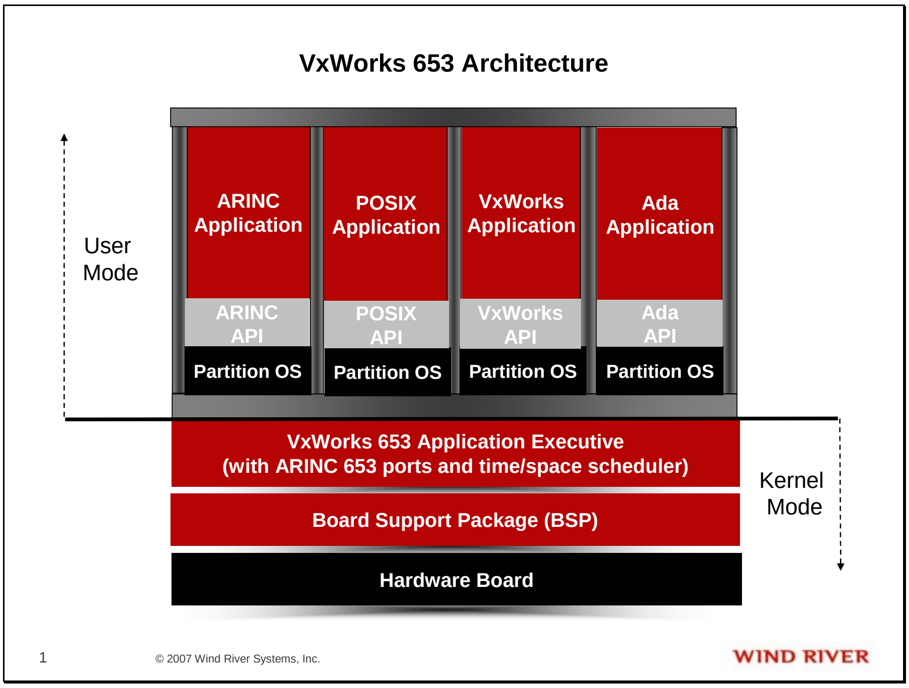
Alex Wilson obtained a BSc(Eng) in Electrical Engineering from Imperial College, London in 1986. Prior to Wind River, Alex worked at British Aerospace on Automated Test Equipment for various Inertial Navigation Systems using VME and RTOS technology. He then worked as a Field Applications Engineer (FAE) for Motorola Computer Group working with 68k and PowerPC VME boards and 3rd party Real Time Operating Systems. He joined Wind River in the UK as an FAE in 1996 supporting VxWorks and Tornado. In 2002 he became European Business Development Manager for Wind River focusing on the Aerospace and Defense market. As Senior Program Manager for Wind River, he is responsible for A&D programs and Opportunities in the EMEA region (Europe, Middle East and Africa)

ARINC 653 specification

Is a specification for an application executive for Integrated Modular Avionics. 51 routines: time and space partitioning, health monitoring, comm. via ports,

VxWorks 653 platform

Provides certification evidence (not certified OS). Designed for performance: implements a two-level OS model, scales from a single partition to a max of 255 partitions.



VxWorks 653 Architecture

User Mode

ARINC Application | POSIX Application | VxWorks Application | Ada Application

ARINC API | POSIX API | VxWorks API | Ada API

Partition OS | Partition OS | Partition OS | Partition OS

VxWorks 653 Application Executive
(with ARINC 653 ports and time/space scheduler)

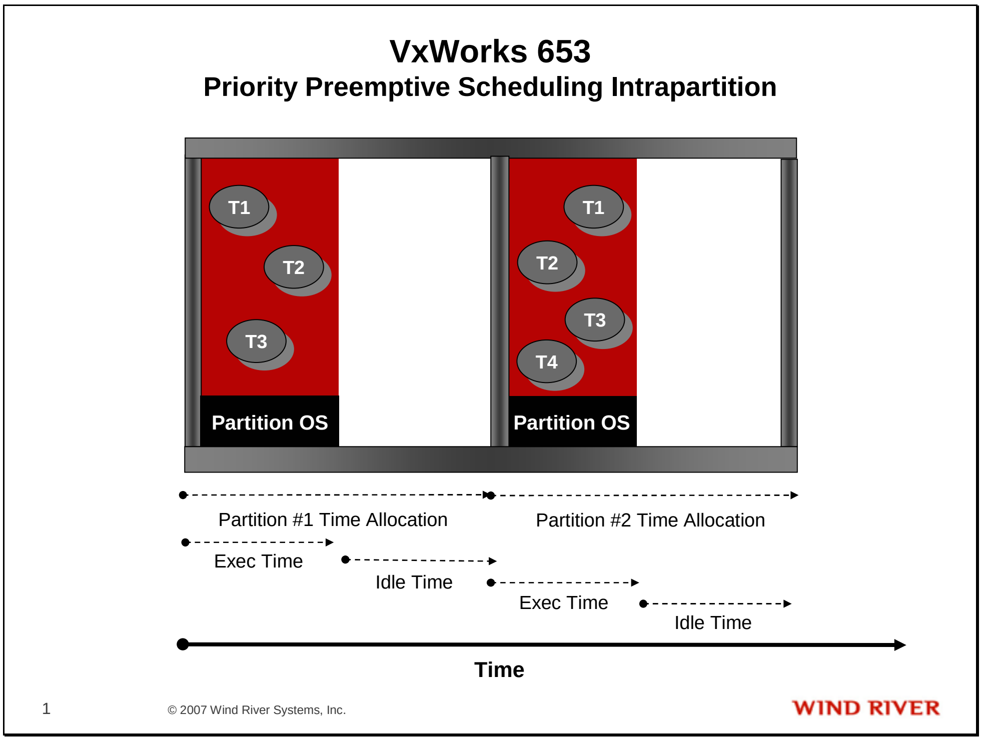
Board Support Package (BSP)

Hardware Board

Kernel Mode

1

Provides a time and space allocation for each partition. Inside partitions priority preemptive scheduling.



VxWorks 653
Priority Preemptive Scheduling Intrapartition

Partition #1 Time Allocation

Partition #2 Time Allocation

Exec Time

Idle Time

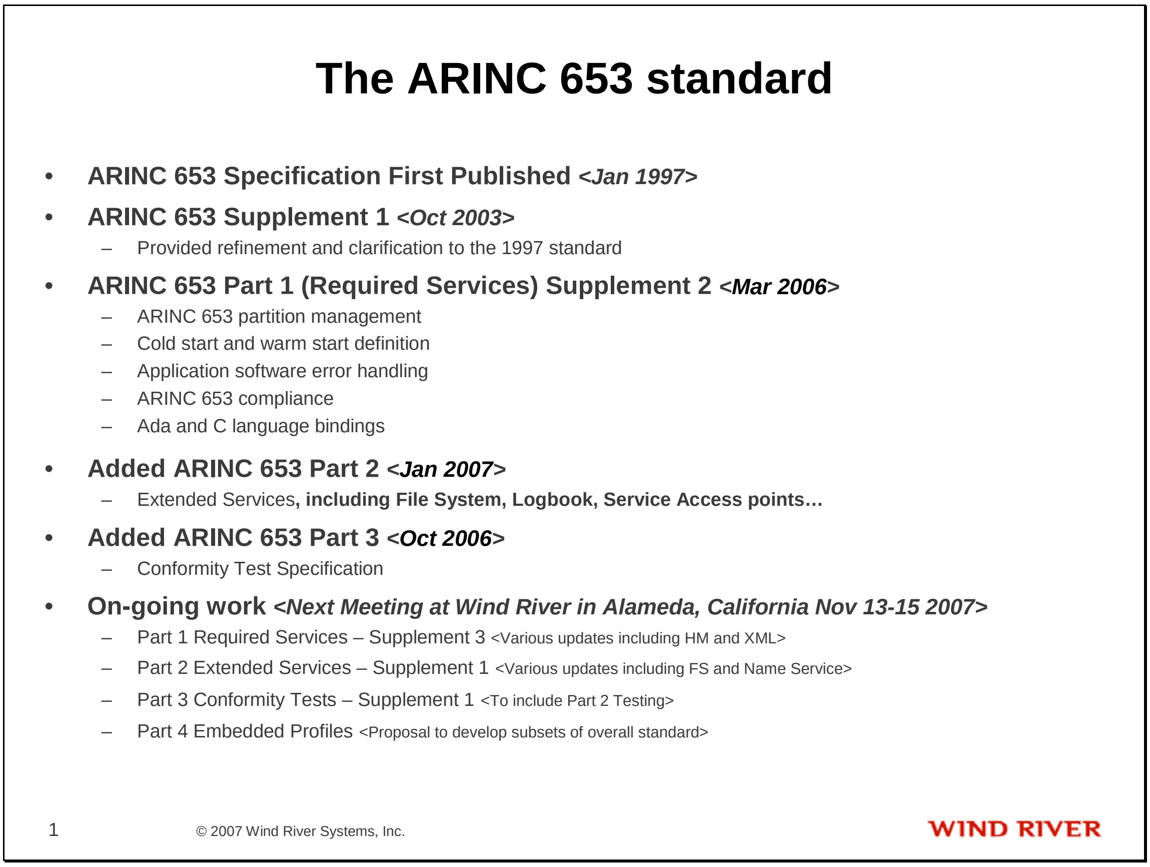
Exec Time

Idle Time

Time

1

The ARINC 653 standard is summarized in the figure below



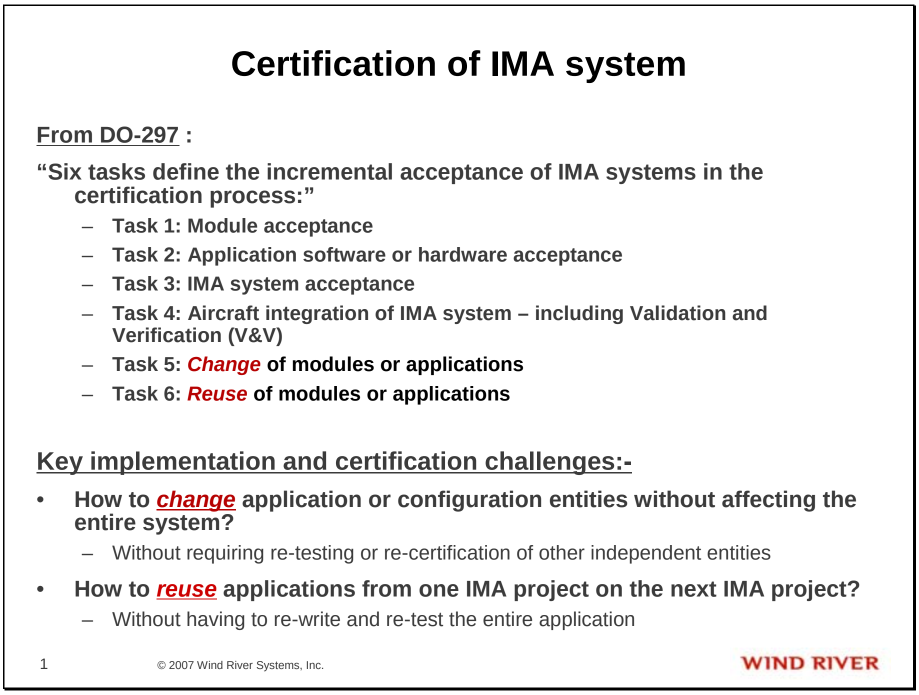
The ARINC 653 standard

- **ARINC 653 Specification First Published** ***<Jan 1997>***
- **ARINC 653 Supplement 1** ***<Oct 2003>***
  - Provided refinement and clarification to the 1997 standard
- **ARINC 653 Part 1 (Required Services) Supplement 2** ***<Mar 2006>***
  - ARINC 653 partition management
  - Cold start and warm start definition
  - Application software error handling
  - ARINC 653 compliance
  - Ada and C language bindings
- **Added ARINC 653 Part 2** ***<Jan 2007>***
  - Extended Services**, including File System, Logbook, Service Access points…**
- **Added ARINC 653 Part 3** ***<Oct 2006>***
  - Conformity Test Specification
- **On-going work** ***<Next Meeting at Wind River in Alameda, California Nov 13-15 2007>***
  - Part 1 Required Services – Supplement 3 <Various updates including HM and XML>
  - Part 2 Extended Services – Supplement 1 <Various updates including FS and Name Service>
  - Part 3 Conformity Tests – Supplement 1 <To include Part 2 Testing>
  - Part 4 Embedded Profiles <Proposal to develop subsets of overall standard>

1 © 2007 Wind River Systems, Inc. WIND RIVER

RTCA DO 297 / EUROCAE ED 124 – Guidance and certification considerations

> “Provides guidance for IMA developers, integrators, applicants, and those involved in the approval and continue airworthiness of IMA systems. It provides specific guidance for the assurance of IMA systems as differentiated from traditional federated avionics”



Certification of IMA system

From DO-297 :

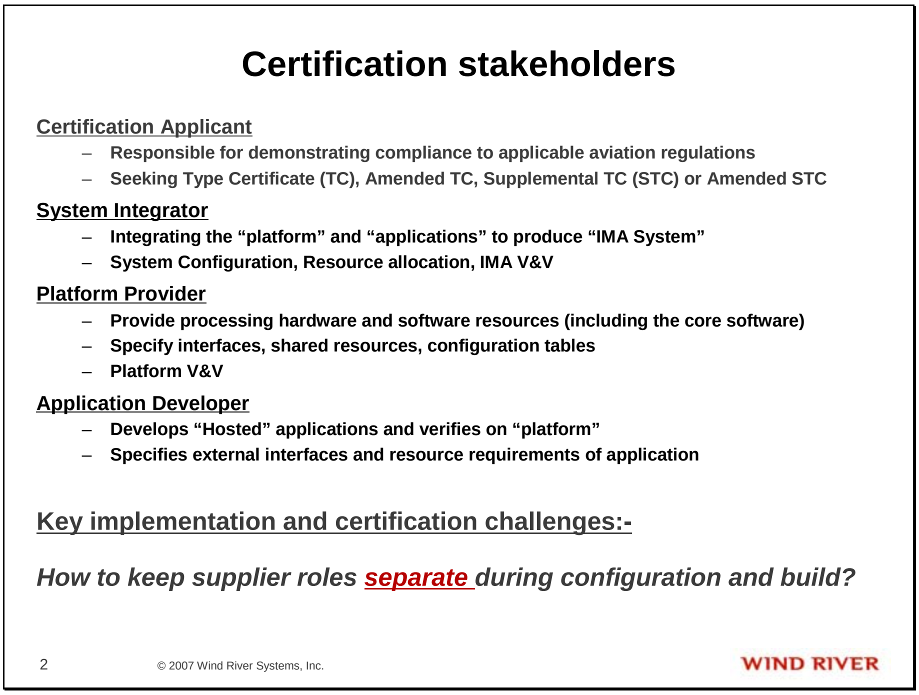
"Six tasks define the incremental acceptance of IMA systems in the certification process:"

- Task 1: Module acceptance
- Task 2: Application software or hardware acceptance
- Task 3: IMA system acceptance
- Task 4: Aircraft integration of IMA system – including Validation and Verification (V&V)
- Task 5: *Change* of modules or applications
- Task 6: *Reuse* of modules or applications

Key implementation and certification challenges:-

- How to ***change*** application or configuration entities without affecting the entire system?
  - Without requiring re-testing or re-certification of other independent entities
- How to ***reuse*** applications from one IMA project on the next IMA project?
  - Without having to re-write and re-test the entire application

1 © 2007 Wind River Systems, Inc. WIND RIVER



Certification stakeholders

Certification Applicant

- Responsible for demonstrating compliance to applicable aviation regulations
- Seeking Type Certificate (TC), Amended TC, Supplemental TC (STC) or Amended STC

System Integrator

- Integrating the "platform" and "applications" to produce "IMA System"
- System Configuration, Resource allocation, IMA V&V

Platform Provider

- Provide processing hardware and software resources (including the core software)
- Specify interfaces, shared resources, configuration tables
- Platform V&V

Application Developer

- Develops "Hosted" applications and verifies on "platform"
- Specifies external interfaces and resource requirements of application

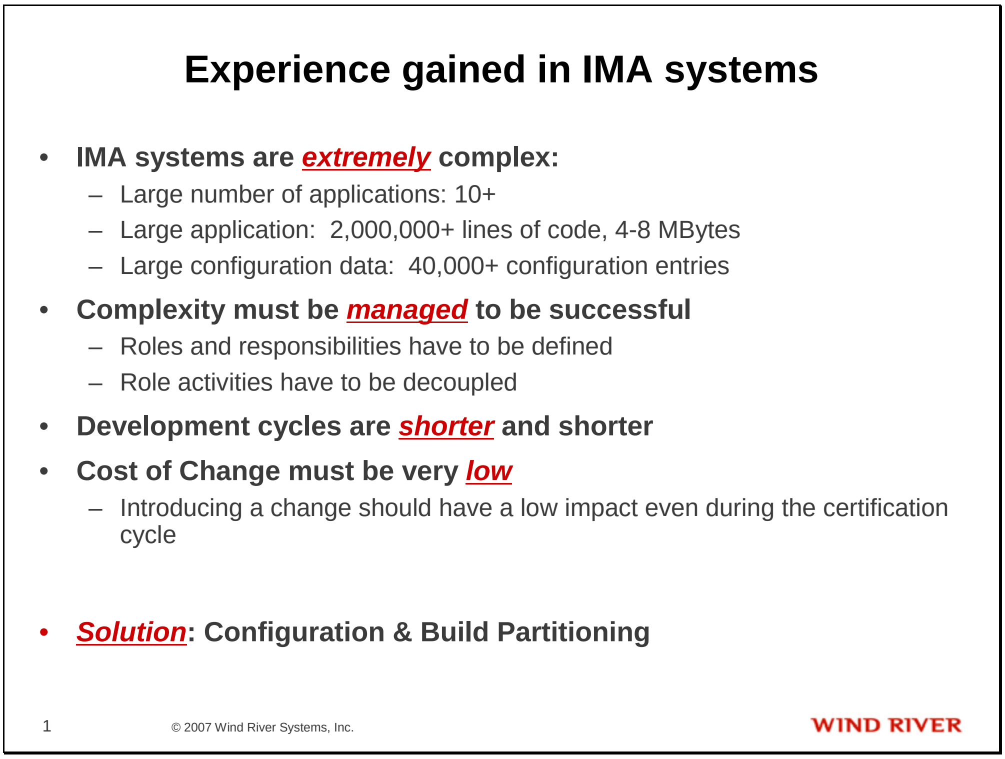
Key implementation and certification challenges:-

*How to keep supplier roles **separate** during configuration and build?*

2 © 2007 Wind River Systems, Inc. WIND RIVER

Keep supplier roles separate during systems design. In addition some suppliers may be hostile against each other (I.e. competitors who were bidding for higher level components).

Typical federated architecture is straightforward from SW viewpoint. When you move to IMA, the process of certification becomes more complex. This is due to actors interdependencies (for example, platform provider and application provider interact regarding performances). IMA system integrator has to resolve all conflicts and trade-offs.



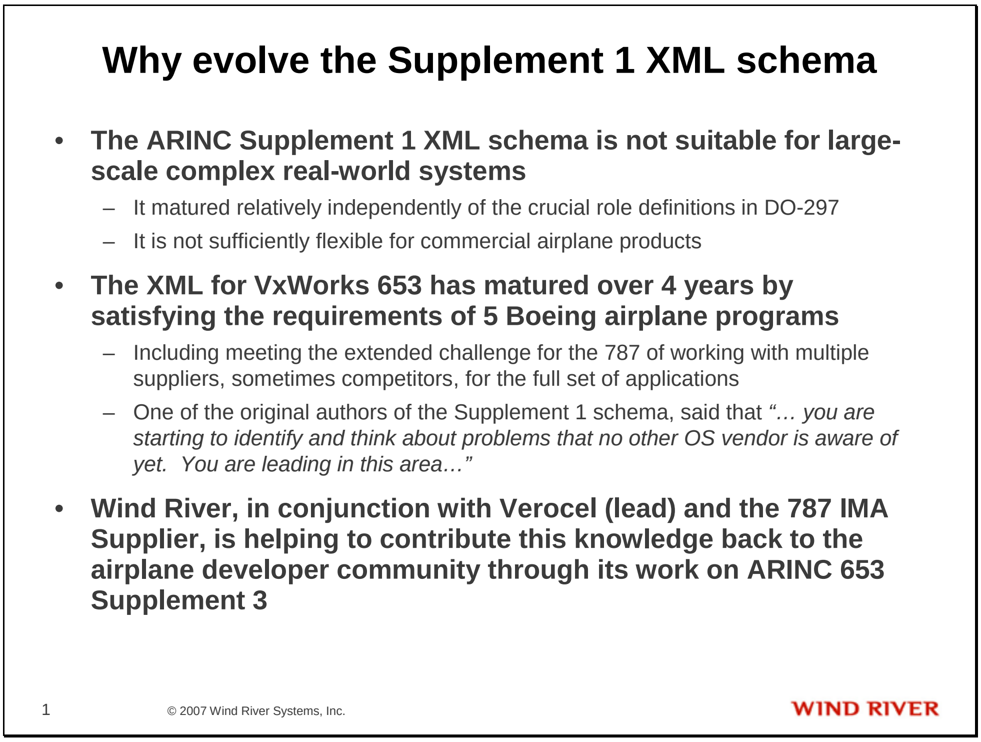
Experience gained in IMA systems

- **IMA systems are *extremely* complex:**
  - Large number of applications: 10+
  - Large application: 2,000,000+ lines of code, 4-8 MBytes
  - Large configuration data: 40,000+ configuration entries
- **Complexity must be *managed* to be successful**
  - Roles and responsibilities have to be defined
  - Role activities have to be decoupled
- **Development cycles are *shorter* and shorter**
- **Cost of Change must be very *low***
  - Introducing a change should have a low impact even during the certification cycle
- ***Solution*: Configuration & Build Partitioning**

1 © 2007 Wind River Systems, Inc. WIND RIVER

XML based configuration

Supplement 1 XML schema serves for system partition configuration. Comments follow:

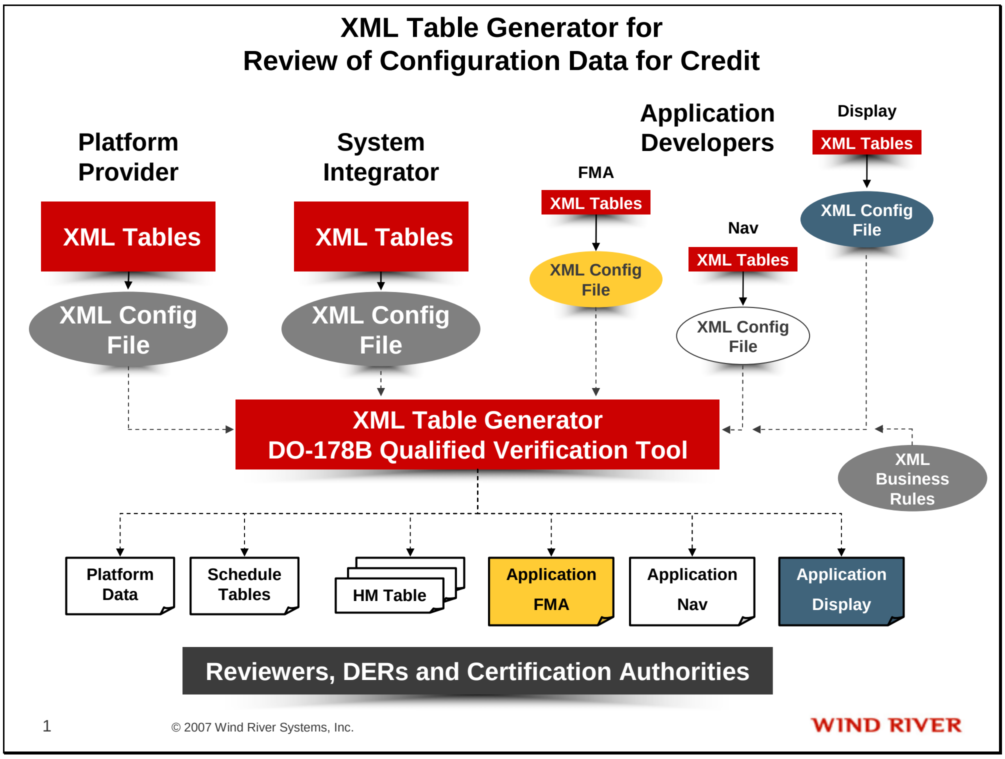


Why evolve the Supplement 1 XML schema

- **The ARINC Supplement 1 XML schema is not suitable for large-scale complex real-world systems**
  - It matured relatively independently of the crucial role definitions in DO-297
  - It is not sufficiently flexible for commercial airplane products
- **The XML for VxWorks 653 has matured over 4 years by satisfying the requirements of 5 Boeing airplane programs**
  - Including meeting the extended challenge for the 787 of working with multiple suppliers, sometimes competitors, for the full set of applications
  - One of the original authors of the Supplement 1 schema, said that *"… you are starting to identify and think about problems that no other OS vendor is aware of yet. You are leading in this area…"*
- **Wind River, in conjunction with Verocel (lead) and the 787 IMA Supplier, is helping to contribute this knowledge back to the airplane developer community through its work on ARINC 653 Supplement 3**

1 © 2007 Wind River Systems, Inc. WIND RIVER

How to avoid the problem that a change in requirements (Schedule, resources etc) for a partition could affect the entire system? A solution is proposed to cope with this problem by using a specific representation of all ARINC 653 resources for partitions, in terms of XML tables. This XML schema is proposed as part of ARINC 653 Supplement 3, with Wind River as author of the proposal. By defining the XML configuration data in this way the XML Schema can be split into the configuration data required for each role and then validated against the standard. Thus the supplier modifies his XML tables and the system integrator can re-generate the entire ARINC653 system. Wind River provides a qualified tool that validates the XML configuration data and together with qualified XML compiler produces bit level configuration data. This provides support for traceability of the XML tables. This illustrated in the figure below.



XML Table Generator for
Review of Configuration Data for Credit

Platform Provider: XML Tables → XML Config File

System Integrator: XML Tables → XML Config File

Application Developers

FMA: XML Tables → XML Config File

Nav: XML Tables → XML Config File

Display: XML Tables → XML Config File

XML Business Rules

XML Table Generator
DO-178B Qualified Verification Tool

Platform Data

Schedule Tables

HM Table

Application FMA

Application Nav

Application Display

Reviewers, DERs and Certification Authorities

1 © 2007 Wind River Systems, Inc. WIND RIVER

Benefits:

- clearly defines responsibilities and ownership of configuration data
- enables configuration entities to be submitted independently
- incremental changes without impacting entire program
- contracts between roles
- preserves confidentiality of IP

Discussion

Q: how can you avoid the change in a partition scheduling to affect the entire system? ***A: see above using our XML configuration compiler.***

Q: in this XML configuration tool what is specific to your OS and HW platform? ***A: This is defined in the ARINC 653 specification. The XML Schema follows this***

specification as does the OS*Q: maintenance, how easy is it t upgrade OS version against certification?* ***A: make sure that you check also that applications running on top of it have to be changed. It is recommended to keep the OS stable****.*

Replacing the core OS would requires entire re-certification due to the fact that the Applications rely on this stable tested base platform in order to meet their own requirements. Any changes in this base platform would require re-certification.. Replacing an application may be OK following an impact analysis with only partial re-certification, particularly with the help of the qualified XML tool to see the impact of that change..

Q: J Rushby complains about cost of DO 297 and complexity. ***A: DO-297 is a world wide standard and is recognized by both US and EU companies and certification authorities. As such, and along with the fact that companies such as Boeing and Dassault are adopting it, it will become the standard guideline to follow when producing an IMA system. Rushby pushes for formal methods and compositionality, but we expect to see this a lot more for MILS when security is concerned. this is because formal methods are then mandatory and compositionality is a must.***

Q: (Airbus) is not following exactly the DO 297 recommendations; in particular, configuration tables are directly managed by the system integrator.

Chris J. Walter, WW Technology Group: Dependable solutions for IMA

Past experience includes being an architect for various fly-by-wire systems. He worked a lot about IMA requirements for Boeing. (Company's mission: Dependable solutions and tools for IMA)

Lessons learned

Integration

may reduce the physical connections but radically increases information/logical connections

Not all connections are apparent, new failures can happen due to sneak paths, priority inversion, bus management, meta-stabilities and loading factors.

What's on paper not always translates into reality.

Increasing functionality and complexity are stressing design capabilities

- requirements for fault tolerance
- certifying interactions can become difficult, requiring X-domain analysis

Modularity

Modularity implies packaging of functionality and instantiations. How to assess coupling and cohesion:

- information locality?
- levels of security/risks?
- operator locality?
- maintenance locality?

Providing modules that support additional goals. Does modularity support:

- performance
- security
- safety
- maintainability

Perceptions and reality

Some perceived solutions: channelize functionality, distribute; who is right: on-line controller or monitor? Reality: truth vs. consensus, meta-stability.

Strategy

Step 1

First address modularity. Define associated attributes: functions, “ilities”. Identify relationships.

Step 2

Next, address integration (coupling factors, integration factors, and complexity metrics). Ensure that no violations are made in policies or modularity strategies.

Step 3

Address avionics domain needs. Extend IMA to be fault tolerant (can it be made compositional, observability, schedulability; are there asymmetric dependencies, is one component more important than the other, are asymmetries justified?)

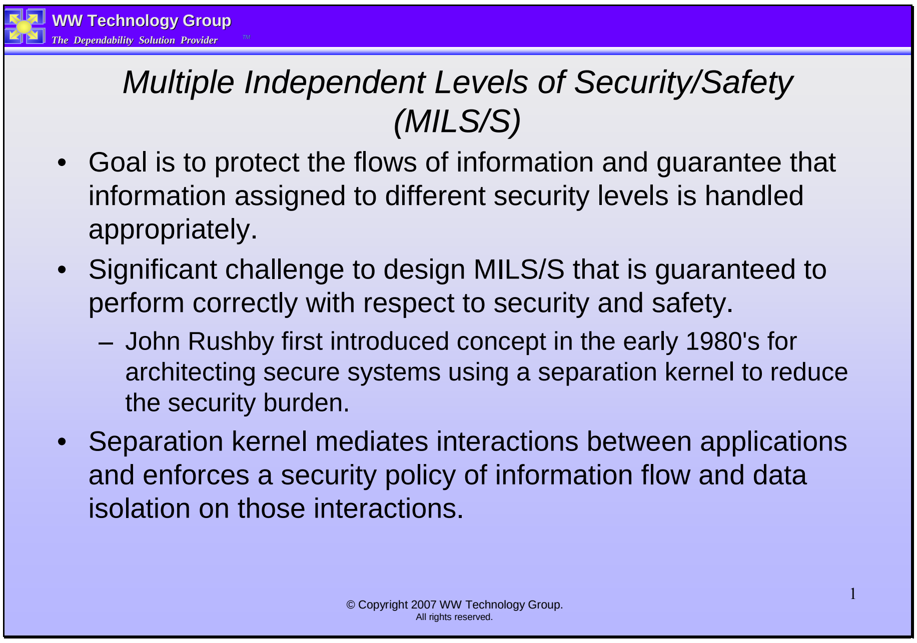
Partitioning

Partitioning is a useful concept. It can be used to contain errors, functionality, restrict information flow.

Model based approach

It is highly valuable for system level design. System architecture plays a central role; it influences many of the key system development activities.

MILS: Multiple Independent Levels of Security/Safety



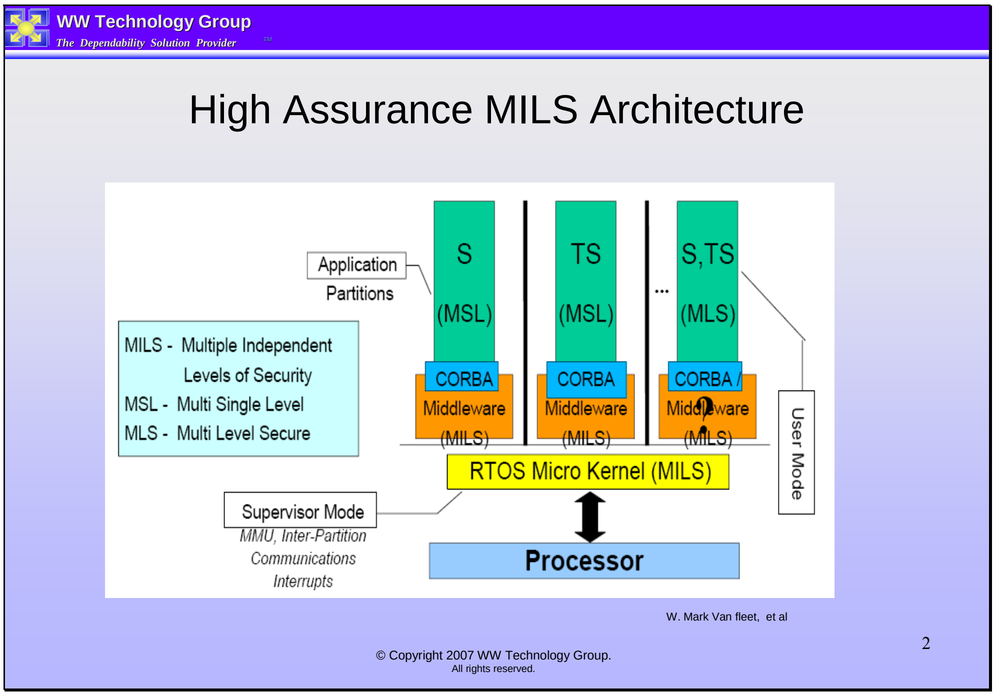
WW Technology Group
The Dependability Solution Provider ™

Multiple Independent Levels of Security/Safety (MILS/S)

- Goal is to protect the flows of information and guarantee that information assigned to different security levels is handled appropriately.
- Significant challenge to design MILS/S that is guaranteed to perform correctly with respect to security and safety.
  - John Rushby first introduced concept in the early 1980's for architecting secure systems using a separation kernel to reduce the security burden.
- Separation kernel mediates interactions between applications and enforces a security policy of information flow and data isolation on those interactions.

© Copyright 2007 WW Technology Group.
All rights reserved.

1



WW Technology Group
The Dependability Solution Provider ™

High Assurance MILS Architecture

W. Mark Van fleet, et al

© Copyright 2007 WW Technology Group.
All rights reserved.

2

MILS architectures are analyzed using a component-based analysis approach. It is very important to understand the information flows. Certain aspects are analogous to the analysis of error propagation that are available in current version of EDICT but needs some adaptation.

EDICT tool to simplify Model and analysis navigation for MILS system design. EDICT evaluates error propagation and impact. Helps identifying issues in error containment.

Design for certification

It applies to a wide range of systems.

Discussion

Q: what kind of analysis techniques can be applied to AADL models? ***A: currently dataflow analysis for possible error propagation. Stochastic analysis of fault propagation is performed, e.g., fault tree analysis.***

Q: do you encompass HW? ***A: yes, through the way it is viewed in AADL****.*

Q: what kind of semantics does AADL offer? ***A: roles are clearly defined (thread concept is defined via hybrid automata models). In the original standard it was assumed that the system is synchronous. Now there is an extension to deal with asynchrony, because some asynchronous buses are widely used.***

Q: precise semantics implies that this comes up with effective mathematics offering properties. When considering HW description languages, this is very important. ***A: people working in the formal methods area contribute to the AADL standard toward providing this semantics for AADL concepts****.*

Panel Session on expectations from research for IMA

Benefits of moving to IMA

Enabling changes in the organization of the sector

The sector is shifting

- from an industry where lots of important development were performed in-house or at least tightly kept under control via a small number of supplier layers
- to an industry where higher degree of outsourcing is performed, with more levels of suppliers and less visibility on the sub-systems or components being integrated

This move would be increasingly costly in terms of components by maintaining a federated architecture approach.

IMA and the work performed by standardization bodies have allowed to clearly identifying the roles of the different stakeholders:

Certification Applicant

- Responsible for demonstrating compliance to applicable aviation regulations
- Seeking Type Certificate (TC), Amended TC, Supplemental TC (STC), or Amended STC

System Integrator

- Integrating the “platform” and “applications” to produce “IMA System”
- System Configuration, Resource allocation, IMA V&V

Platform Provider

- Provide processing hardware and software resources (including the core software)
- Specify interfaces, shared resources, configuration tables
- Platform V&V

Application Developer

- Develops "hosted" applications and verifies on "platform"
- Specifies external interfaces and resource requirements of application

There are key implementation and certification challenges. Keeping the above roles separate during system development is considered important.

Avoid the explosion of computing and networking resources

Reduce diversity in components

Same resources can be used for different variants of aircrafts.

Simplify maintenance

2nd level of maintenance removed, thanks to LRM concept (Line replaceable module) at a potentially higher component replacement cost.

Reduced weight, volume, power consumption

This is due to the sharing by several application of a same MDPU.

Open architecture and standardised interfaces, allows coping with obsolescence and provides room for upgrades

Allow more flexibility while splitting into sub-systems

Subsystems can be just middleware, or bare computing hardware, or OS components… Systems design requires better componentization and use of contracts: helps moving to modular and incremental certification.

Dependability and maintenance

Integrated architectures allow for more efficient types of redundancy to achieve the level of fault tolerance requested by the application than federated ones (e.g., two-pair redundancy). In fact, fault tolerance can be achieved by replicating functionality distributed over the architecture while exploiting partial utilization of resources for primary functions without the need of replicating architectural modules.

Maintenance is simplified by keeping control over the number of different items.

Risks in moving to IMA

Can we have IMA while keeping advantages of federated architectures in terms of safety?

Neither Boeing nor Airbus moved entirely to IMA at once. There are some fears of risks. It is good to move in a progressive way.

Even in federated architectures, buses are separated for critical and non-critical systems.

IMA does not mean that we don't have dedicated resources – this is of course allowed; just we don't want this to be the rule. Whenever possible and convenient, resources can be shared among functionality. All shared resources must be explicit (communications).

Risk coming from non predictability of supporting architecture

Lots of what has been presented today relies on the capability of making precise WCET evaluations and predictions on the complex architectures encountered now. One should start dialog with HW makers as to whether they are prepared to facilitate predictability in time for their architectures, targeting the safety critical embedded systems sector.

Note that this is not a new problem coming with IMA (it was already present for federated architecture and addressed there). But IMA will need to integrate more and more applications on the same Core.

There are current lobbying activities toward chip manufacturers to put pressure in this direction. There is a planned ARTEMIS event on this topic. There is a need to explain that it is not only about average cases and throughput that matter but rather reducing WCET and jitter for safety-critical applications. Even a small improvement can help, e.g., in cache management policies.

ESA (European Space Agency) has developed a series of radiation hardened computers. The latest is the Rad Hard 32-bit SPARC V8 embedded processor called LEON2. The implementation is based on the European Space Agency LEON2 fault tolerant model, and is available from Atmel. It is a 100MHz low power (1 Watt) design, providing 86MIPS (Drystone) and 23 MFlops (Whetstone). Currently a new iteration AT697F is in manufacturing which will also include MMU support. The LEON3 is available as IP-core for custom SoC design, as well available in a standard edition as in a Fault tolerant edition. ESA is currently initiating studies to produce a multi-core processor based on the same SPARC architecture.

The automotive/avionics market is not going to drive the chip manufacturing market

However: today a fairly complex chip costs 50million $ to design. A microprocessor for computing applications such as an Intel chip takes over 2 years and more than one thousand designers to complete. At a cost of 200,000 $ per designer average loaded cost, with mask sets that cost more than a million dollar and with a manufacturing plant of costs that reach more than 2 Billion US $ that has to be amortized over one or two years, a modern microprocessor may cost in the range of 600Million $ to design. We cannot afford to customize any non trivial processor. Because of design and manufacturing cost, small volume, customized applications cannot be afforded unless the price of the component is very high as it may be the case for the defense industry. Avionics and automobiles are not going to be the drivers; cell phones and game consoles are.

We should look instead at architectures that are going to mass market and look at how to use them to our advantage. The economy of scale will favor this approach. To achieve the goals that safety critical application demand, we should rather develop *platforms* at higher levels of abstraction instead, that allow obtaining desirable architectures on top of less desirable ones.

We need more performance to integrate more applications in a single chip from the market. So we insist on having methodologies and architectures and design tools supporting this. With the current technology we will find limitations in integration. One consequence is that WCET estimates need to get more precise.

Complexity of processors

Several types of processors are complex. This includes, e.g., networking chips and graphical and signal processing processors. Processor complexity is a major issue. There are DO recommendations for FPGA-based processors.

In general everything with HW is becoming very complex. When you have the capability to put many things into a single chip, then complexity comes in very rapidly. Complexity is about the number of design decisions that has been made to perform the design.

Those risks must be considered. There is no silver bullet. A combination of counter measures must be considered.

Complexity of problems

Problems are complex too, also in applications. There are lots of possible ways to implement the same thing. Huge numbers of options follow. Putting more asynchrony in architecture adds more problems than you need to.

Validating OS? What does it mean to validate OS w.r.t. properties offered for partitions?

As an example, small OS for Java card have been certified. *Certified OS does not mean correct OS.*

Wind River, for instance, provides all evidence to the authorities that OS is correct for its usage. Related assumptions are documented in detail. Full responsibility is not to the OS only. For the VxWorks 653 configuration data a qualified code generator helps you for the configuration, mostly regarding traceability. Validating robust partitioning must be done on case by case, not in a generic way, as it relates directly to hardware and systems design.

MILS is a special case, due to the complexity of validation and requirement for formal methods to support it. The separation kernel used for MILS will have to go through a formal proof.

Partitioning risks

Partitioning may provide a false sense of safety when dealing with the separation of different functions.

Lots of hidden assumptions are made regarding partitions by teams of different backgrounds (it’s implicit to them, not to the other) – [This is why you need the Platform Provider role to ensure this does not happen and that robust partitioning is maintained]. Concurrency and communication are major sources of problems.

- partitions cannot affect other partitions in terms of resource use, not always true, see figure above; a partition may affect the functioning of another one;
- partitions cannot affect OS services: not always true…

- scheduling analysis is partition insensitive: wrong; assumed analysis results must be updated and revisited, see figure above.
- fault tolerance through redundancy; partition does not guarantee as much as physical separation or does not guarantee in the same way as physical separation but it may be more effective if the issues is handled explicitly.
- Inter-partition is always phase-delayed: not so simple

Such assumptions must be documented as they are not avoidable. Make this explicit and include the problem in the analysis, do not hide it.

Risks of putting all your eggs in one basket

Diversity in design is important. The common mode issue is more severe than for federated architectures. Taking topological and timing considerations carefully into account should be the standard practice. The safety analysis must be revisited when shifting from federated to integrated architecture. Analysis cannot be performed in isolation. Transversal aspects must be addressed. [ALL of these things need to be done regardless of whether this is an IMA design or a Federated Design – are these really IMA issues?]

Do we have a sufficient toolset to support IMA?

Everything seems to be developed in-house in terms of tools. Is this the right way? In fact, this issue is not specific to IMA but rather related to the aeronautics sector in general. Does this sector favor an ecosystem of tool vendors to support their design process?

Regarding IMA specifically, there is a general risk in introducing it because of the new processes that go along with it – even worse we sometimes forget that there should be a new process to be developed and we don't do it.

What about power management in IMA?

For example, simultaneous demand for more power from different modules can occur: is this considered as part of the new design process?

Power management must be globally controlled, together with the control of functionality. There is a problem with the corresponding time-scales: you must react quickly to something, however power decisions take much longer to deploy. In addition, power does take a comparably much longer time to dissipate with respect to change in functionality.

For IMA several options can be made regarding power generation: fully decentralized or partially decentralized.

Also, local power management inside modern processors affects the behavior of processors and therefore increases their non predictability.

Increase in the risk of common mode design as a result of reducing the number of parts

Common modes in computer cooling systems were a cause of an accident in an airplane recently.

Redundancy concepts were clearer in federated architectures than it is in IMA. Therefore, clean and finer dependability analysis is more critical. Dependability analysis can no longer be performed in isolation from functional specification.

In turn, in integrated solutions, with a good global view, one can save redundancy and still achieve better fault tolerance, as compared to federated.

With IMA we must move, from an independent stand alone risk analysis, to a joint application, platform, and allocation risk analysis.

What about IMA in handling legacy code or designs?

To exploit fully the characteristics of IMA, applications should be re-implemented. However, the cost of doing so may be prohibitive and the verification process for the new implementation may take too long. Handling legacy implementations is a key issue in embedded systems. Federated architectures are fine at doing this. Encapsulation of legacy designs and dealing with this design as a component in IMA is a feasible solution.

- The skeleton applications were redesigned and adapted when moving to IMA (Dassault).
- Recoding in Ada the interfaces of legacy designs was performed at Honeywell.

Should safety of IMA entirely rely on infrastructure?

Not possible. Peter Feiler mentioned the case of latency in his presentation.

Radiation tolerance

May be an area for non-competitive cooperation?

Gap between higher level design (programming guidelines) and lower level architecture aspects?

People write SW the way they can and then they ask analysis tools to carry the burden. One should instead write most possible correct-by-construction SW.

How architects and SW engineers do communicate with control engineers about the features of their architectures and requirements for the architecture?

What are the relevant features of our architectures that we need to communicate to control engineers for them to develop robust control? No good research for this has been developed in the academic area. Features are communicated from architects to control engineers and requirements are communicated from control engineers to architects, see 0

Application reuse is a major issue

1 line of code costs 100$ under DO level A. One would like to maintain certification through reuse when moving from one IMA generation to the next one.

Research issues

Mitigating with the complexity of processors and architectures

Risk coming from non predictability of supporting architecture

Lots of what is developed today for IMA relies on the capability of making precise WCET evaluations and predictions on the complex architectures encountered now. One should start dialog with HW makers as to whether they are prepared to facilitate predictability in time for their architectures, targeting safety critical embedded systems sector. Research should look in two directions:

- A systematic approach toward predictability at all stages of the design, encompassing both the HW engines, and how to program them (putting important non-functional characteristics on an equal foot with the functions, in the proposed programming formalisms).
- Low cost but clever adaptations of mass market designs intended to facilitate in large part the analysis needed to support partitioning and other important concepts of IMA.

We should look at architectures that are going to mass market and look at how to accommodate them. We should develop concepts of *platforms*, that allow getting desirable architecture on top of less desirable ones

This is going the opposite direction as the previous point. Both are worth exploring – either direction will have its supporters.

Dependability and related risks arising from IMA

Additive layers for safety and dependability

IMA increases risks of unwanted hidden interactions between partitions, due to potential for interactions at various layers of design (from chip to middleware). In his paper on just-in-time certification, John Rushby suggests looking for *additive layers* contributing in a validated and explicated way to partitioning and other features of importance for dependability under IMA.

Increase in the risk of common mode design as a result of reducing the number of parts

Reducing the number of parts requires more applications to share the same resources. Verification becomes increasingly difficult. Analysis requires a level of modeling of the distributed inter-task interactions that has not been confronted before. Risks can be reduced by requiring that every task is allocated to only one resource, thus simplifying the analysis of the interactions due to the communication infrastructure. However, this approach will not exploit the integration aspects at its full capability. There is a need of developing theories to guarantee correct behavior for control and other functionalities when physical resources are shared. In general, there is a need to develop low cost formal methods based on formalisms friendly to application engineers.

The MILS case

MILS is a special case, due to the complexity of validation and use of formal methods to support it. The separation kernel used for MILS will have to go through a formal proof.

Research on MILS is already looking at Compositionality and the requirement to prove the MILS architecture is valid – i.e. Kernel, middleware and applications all validated separately and then merged together.

Validating OS? What does it mean to validate OS w.r.t. requirements for IMA?

As an example, small OS for Java card have been certified. Note that *certified OS does not mean correct OS.*

Research should be undertaken to facilitate the qualification of the OS in the context of IMA. The same holds for certification.

Regarding certification, proofs should consider explicitly assumptions regarding the resources and environment of the OS. To make validation easier, the OS in its particular configuration, not its generic form, should be considered. Alternatively, validation may concentrate on smaller parts of the OS considered most difficult, leaving aside other aspects for inspection using more traditional means.

Do we have a sufficient toolset to support IMA?

Everything seems to be developed in-house in terms of tools. Is this the right way? In fact, this issue is not specific to IMA but rather related to the aeronautics sector in general. Does this sector favor an ecosystem of tool vendors to support their design process?

Regarding IMA specifically, there is a general risk in introducing it because of the new processes that go along with it – even worse we sometimes forget that there should be a new process to be developed and we don't do it.

Vendors are not aggressively developing tools and frameworks for IMA. This may be due to similar issues confronted when developing custom HW: cost of developing products vs. total available market in terms of potential revenues.

IMA requires the creation of configuration tables including explicit parameters more or less described in standards. Tools are needed to configure easily an IMA, to give a clear view between functional requirements, non-functional requirements, and platform dependencies. Then these tools must validate the tables and mainly mode changes.

Pursue research on methods and toolset support for the design process of IMA

This includes providing support for concurrent development of different systems by different suppliers, and by different aspects of a same system by different teams. Models of target architectures must be available. A virtual exploration, validation, and possibly optimization of the entire design (application + platform) at different levels of detail must be supported, encompassing both functional, timing, and energy aspects. Important advances in the academia are now underway (see the presentation by Gert Döhmen) and the aeronautics sector could take advantage of recent developments in the automobile sector with the support of AUTOSAR architecture and process.

What about power management in IMA?

For example, simultaneous demand for more power from different modules can occur. Power management must be globally controlled, together with the control of functionality. There is a problem with the corresponding time-scales: you must react quickly to something, however power decisions take a long time to deploy; power takes also a long time to dissipate. For IMA several options can be made regarding power generation: decentralized or partially decentralized. Also, local power management inside modern processors affects the behavior of processors and therefore increases their non predictability.

Research topic on power management

Power management approaches should consider the effect that decisions may have on the correct behavior of the system. Reducing power consumption on a computational module may reduce its performance so much so data may not be available to another unit at the time it is required thus creating serious functional problems. In addition, power management should consider the power demand on a global scale to prevent serious HW faults. Because of a sudden surge of computation requested by the application, the processing elements may weak up all together and provoke power spikes and other dangerous events. Power management must have a complete view of the application evolution. Interesting research is about developing a decentralized power management process that would guarantee correct overall behavior of the system.

How architects and SW engineers do communicate with control engineers about the features of their architectures?

Control engineers now develop their control loops with modern techniques based on certain techniques for specifying uncertainties or tolerance bounds regarding the plant (H^2/H^∞ robust design techniques). It is unclear whether these techniques also encompass features of the computing platform. At least, little has been done in the academic community regarding this. On the other hand, it makes little sense to communicate to the control engineer a *model of the computing architecture* according to the software engineering viewpoint. There is a need for a design approach that exposes the control designer with an abstract view of the computation and communication resources that can be considered in the design of the control algorithm. On the other hand, there is a need for propagating constraints from control designers in a form that can be accommodated by HW and middleware designers so that the overall behavior of the system is guaranteed.

Proper level of abstractions for IMA computing platforms for use as assumptions by control engineers should be provided; associated techniques of control design should be developed

Applicatiion prediction and reusability

Currently, standards allow developing applications based on some API. This gives the "plug" functionality. But with current specification mode, it is difficult to predict behavior and performance (for example WCET of an API call under heavy load). This problem is getting worse with modern process and IMA facilities because applications and platform are now developed in parallel, and allocations and requirements must be given very early to each application or platform supplier without knowing implementation constraints. To help both sides, what kind of information should be provided to the system designer? How to perform virtual design space exploration suited to IMA needs?

Regarding the behavior, one of major problems is the prediction of communications. How to perform this easily? How to perform this for COTS? A good way could be to create a set of predictable real-time application design patterns.

One major issue is to reuse applications at efficient cost. What does it means in terms of application architecture, componentization, and code portability, binary portability, taking into account hidden or implicit constraints of the current platform? What are the rules or methods to ensure that a given application developed in 2007 can still be reused in 2020 or later?