ADVANCED ON-BOARD COMPUTING AND DATA PROCESSING: A PRIMARY CONDITION FOR THE FUNCTIONALITY OF MODERN HIGH PERFORMANCE AIRCRAFT

Dr. H. Scheidt
Dr. H. Rapp
Daimler-Benz Aerospace
Military Aircraft
P.O. Box 80 11 60
D-81663 München

Abstract

Recent developments in the area of civil and military aircraft like the Eurofighter EF 2000 are characterized by an increasing significance of digital avionic and flight control systems. Moreover these systems are highly interconnected between each other and with other aircraft subsystems.

The functionality of these aircraft is now mainly supported or even totally provided by software implemented in freely programmable real time computers linked by high speed data bus systems. Thus, highly unstable and manoeuvrable configurations like the X-31 experimental aircraft, terrain data based automatic low level flight or "carefree handling" in any aircraft configuration are achieved.

These advancements require a new understanding of the man-machine interface: Functionality and information have to be made available to the crew in an appropriate manner depending on the flight situation i. e. the mission phase or threat situation. This leads to new developments in the area of crew assistance systems.

Increasing complexity has a strong influence on the processes to develop these systems. The system integration task more and more becomes a software development process. System complexity requires new management methods and means of communication between development engineers as well as advanced methodologies for certification and documentation. Until today many open questions remain with regard to test and certification of flight safety critical software.

1. Introduction

For decades the share taken by equipment in civil and military aircraft has been increasing steadily. At the same time, the use of digital avionics and flight control systems has improved the functionality, flexibility, operability and safety of aircraft significantly. This development includes freely programmable and distributed real-time computers, the move from electro-mechanical systems to software intensive systems as well as the trend to higher integration. Military aircraft has been a pace setter for this development. Let us consider the following example. Since the basic version of the Tornado aircraft went into service in the German Air Force in 1978 the freely programmable computing power has increased with a factor 100. Simultaneously the number of computers, loadable directly on the aircraft increased from 1 to 6. At the beginning of the development we were thinking 8k words would be enough.

In a similar fashion the introduction of data bus systems induced a network of integrated avionic subsystems as well as in a next step, links between avionics, flight control and general systems. As a consequence, the amount of information available to the cockpit crew increased tremendously. Therefore, to reduce crew workload future crew assistance systems must provide sound mission-oriented onboard preprocessing, consolidation and finally, appropriate representation of the relevant information in the cockpit.

System functions and system performance are becoming increasingly dependent on information processing and real-time software. To guarantee a safe aircraft operation under all conditions, the real-time software has to provide different functionalities,

to automate processes and to perform complex error detection and error recovery mechanisms.

The increase in system functionality implemented in software as well as in networking is not only a matter of quantity but also a matter of quality. That means, this increase resulted not only in more complex avionic and flight control systems, but also in more complex system development processes. Growing cost components for avionics and longer development cycles reflect this trend; about 30% of the Fly-Away costs for modern fighter aircraft are covered by avionics.

The change from fixed, function based links of individual equipment to integrated networks of avionics, flight control and general systems, linked together by high speed data bus systems and corresponding software, has increased tremendously the overall functionality of aircraft and has made functions available like highly unstable aircraft configurations, terrain-data based automatic low level flight or "Carefree Handling" with different aircraft configurations. These characteristics are not achievable solely by aerodynamics and intelligent sensors.

On the other hand, this high level of integration has enforced new procedures for system design and integration as well as for the software development process in view of feasibility and safety.

System integration as a process, which combines individual components of the aircraft to a whole system, will become more and more a software development process. As a consequence, verification and validation of software, especially safety critical software, as well as provable test strategies become especially important. The intangible product software puts new requirements on the development process regarding management, traceability, methods and techniques. Whereas on the hardware side well known procedures for documentation and communication of the progress of the project (like Drawing Office Instructions) exist, tracking progress in a software development project, especially for complex systems, is very difficult. It is also true, that the development of a new aircraft with a high degree of interconnection between individual components - including the airframe structure - enforces overall design at the very beginning. Moreover, this linkage has a feedback on the organisation of the overall development process.

Based on the above mentioned development trends, in the course of this paper more details will be given about the connection between data processing and system integration processes, between automation of missions and operations as well as new technological approaches overcoming bottlenecks and problem areas.

2. System Integration and Information Processing

In particular in Europe, work share is an important characteristic of civil and military aircraft products. That means, that aerospace companies from different nations and their suppliers are working together. In this context, common methods and appropriate agreed procedures are important parts of the development process. This is particularly true for software development and testing. In the following present procedures will be discussed.

2.1 Development Model

The principal elements of the system development model for military aircraft, like translating customer requirements into a design concept, implementing and testing up to the delivery to the customer are depicted in Figure 1. Starting from the customer requirements, which summarize the essential ideas for the capability of the aircraft, a system design concept will be developed, prescribing the technical parts of the product aircraft. The development of the airframe structure, which is not an issue of this paper, is part of the process "structure". The process "functionality" includes the development of the characteristics, which are required by the aircraft functions and which are mainly defined by the equipment of the aircraft. Examples are all-weather navigation and approach capability and combat capabilities for military aircraft.

The link between the "airframe structure" and the "functionality" has been getting tighter in time. Both process models, "airframe structure" and "functionality", should be derived from the overall system design and will end in the integration of the airframe structure and the avionics with the subsequent flight test.
The development of the aircraft systems, their software and their integration is part of the process model "functionality". The system integration process starts with the system requirements, derived from the weapon system specification. Moreover, these requirements are the basis for the subsystem specification (avionics, flight control...). The subsystem specifications again form the basis for the definition of the equipment, which build the subsystems as well as the basis for the software

requirements, which determine the equipment software. The equipment itself will be provided by the supplier industry. Usually, the software development is divided into two categories: one is called equipment specific software, e.g. software, which is a fixed part of the equipment, representing a special item of functionality (for example radar or inertial navigation) and which is dedicated to a specific embedded processor. Another category is the system software, which is part of a freely programmable real-time computer, performing mission tasks like navigation, moding, mission planning, resource management and tactical flight guidance. The development of system software will usually be provided by the airframe company itself. System software includes many aircraft and mission specific elements and is subject to many changes to adapt the weapon system to changing requirements.

In a next step, individual components - hardware and software - will be combined to a running system and will be extensively tested. This is the heart of the system development process. In particular, systems have to be verified and validated. System verification includes the demonstration of the system functions, in accordance with the system requirements in an almost realistic system environment, which is realized by appropriate simulations, capable to stimulate physical values and perform data recording and analysing. The system validation will be performed in the intended operational environment, e.g. in flight test, and it contains the proof of the operational customer requirements. By the end of the flight test, the aircraft is ready to go into service.

It should be mentioned, that the on-board computing system, hardware and software and the test and verification system will be developed in parallel.

On the technical level, the development of the system is accompanied by intensive simulations (Figure 2), trade-off studies, risk analysis, architectural considerations, real-time aspects and rigorous configuration control. In case of multinational programs all these processes have to be discussed, agreed and implemented by all partners, where key activities will usually be delegated into international teams.

Experiences in various international co-operation programmes have shown, that the high amount of integration and degree of connectivity in modern aircraft as well as the necessary iterations in the course of the development process must have an organisational impact on the development teams.

An example for the necessary tight coupling between the development of the airframe structure and the avionics may be the connection between aerodynamically induced structured oscillations, location of gyros as primary sensors of the flight control system and the dimension of the digital notch filters within the flight controller. Furthermore, this example shows that it is necessary to start from the very beginning with teams and to give them the responsibility for the overall system design up to flight testing.

2.2 Development of System Software

System software, defined as software, which is implemented in freely programmable computers and which implements essential system functions, today also includes the software of the flight control computers. But, due to its safety-critical aspects, a modified development process has to be used for the flight control software. In the following, the flight control system of the EF 2000 is considered as a recent example for the kind of functionality which is implemented in software today.

The EF 2000 is a single seat military aircraft for the air defense role under all weather conditions. The required performance characteristics for overall mission accomplishment and manoeuvring include

- minimum turn-radius
- high turn-rates (sub- and supersonic)
- high climb-rates
- high acceleration and deceleration capabilities
- maximum agility (quick and precise change of aircraft flight parameters).

These can only be accomplished with an aerodynamic instable Delta-Canard-Configuration. An advanced Fly-By-Wire flight control system guarantees the necessary artificial stabilisation throughout the full flight envelope. This CCV (Control Configured Vehicle) concept delegates the whole and solely authority for the manoeuvring devices to the flight control system.

The basic functions of the aircraft consist of satisfactory re-stabilisation throughout the full envelope (Stability Augmentation) as well as the creation of acceptable aircraft behaviour based on pilots steering commands (Command Augmentation). An essential part of the flight control system are the Carefree Manoeuvring Functions. These are implemented to allow the pilot the almost unlimited use of his controls for all aircraft configurations and

avoid safety-critical uncontrolled flight conditions by controlling the limits of aircraft parameters at the same time. This includes:

- safety-critical uncontrolled flight conditions
- avoidance of stalls and spins
- limitation of vertical acceleration based on aircraft configuration and actual grossweight
- limitation of g-onset
- manoeuvring within the flight envelope

The realisation of these functions is based on formal fault requirements, which prescribe the behaviour of the system in case of fault and after successful fault recovery. Basic electronic requirements including the sensors are:

- no primary fault will lead to degradation of system performance
- after a secondary fault still satisfactory control must be possible.

These considerations were the basis for the quadruplex structure of the EF 2000 flight control system. The flight control system is a channel oriented system, where each channel contains all safety-critical functions. The channels are electrically fully decoupled. The communication between the channels, which is necessary for fault supervision and data consolidation is defined by optical decoupled data buses between the four flight control computers. The signal comparison of redundant information is the basis for fault detection and fault identification.

The heart of the flight control system is build out of four flight control computers, identically in hardware and software, with a multi-processor structure based on a standard microprocessor. Their main tasks are the processing of control laws, air data and I/O data. These computers build a closed control circuit with primary and secondary control surfaces.

In accordance with the above mentioned tasks of the flight control system the whole software of the flight control computer is safety-critical. Interference of the pilot with the flight control circuit and the assessment of its effects are not possible to accomplish in real-time. As result, the structure of the software, the development model and the test procedures used exhibit some special characteristics. The avionics, flight control and general systems software of the EF 2000 aircraft was generally developed in the high level language Ada. Ada as a standardized language was chosen to reduce the efforts in software maintenance and software updating later on when the aircraft will be in service. Moreover, further standardisation issues, like the definition of a common type of processor for all equipment and a guideline for common software development methods and tools were implemented. All supplier companies were involved in this process.

The development of the software for the flight control computer induced the definition of a subset of Ada and an approach adjusted to safety-critical aspects, which guarantee that

- fully logical and physical separation of software modules is realized
- the development process is strictly deterministic
- the interpretation on source code level is unambiguous
- the integrity of the compiling process is ensured.

The software development is accompanied by extensive error tree analysis and intensive verification and testing activities. This includes the determination of all possible error configurations of the system and their effects, the tracking of the requirements through all software components, independent code investigations, static and dynamic modelling as well as extensive ground tests in a Flight Control Rig. This Rig contains additional components of the flight control system, in which all forces acting against the structure of the aircraft can be simulated and which allows checking system behaviour under error conditions.

The development and updating of real-time software for different aircraft models, here shown for the EF 2000 flight control system, in time has led to indepth experiences and to continuous improvements of the development process. Meanwhile it has become clear, that the widespread view that software is easier to change than hardware, is not always true. Technically, changes in software are often easy to realize, but they could enforce considerable efforts regarding the subsequent verification, validation and certification of the software. Considering all accompanying activities (configuration control, document development and processing, change of flight manuals and training procedures, customer training e.t.c.), the effort for software changes, at least regarding safety-critical, technical real-time systems is comparable to hardware changes.
Software changes are essentially based on the following factors:

- unclear, inconsistent and incomplete software requirements

- errors in the definition of technical facts, as the basis for the software requirements
- tight schedules and therefore the tendency to shift detailed software requirements and real-time considerations into late stages of the programme
- changes of customer requirements
- restrictions due to system and hardware, which prevent the implementation of reqirements in the form envisaged
- errors in coding.

While errors in coding ("Bugs") usually are relatively easy to repair (but often result in new errors), the adaptation of the software requirements is a much more costly process. From that point of view it is very important, to define precise software requirements in early stages of the programme, e.g. during an intensive requirement engineering stage. Herewith the expected behaviour of the system will be modelled by simulations, experimental prototyping and dynamic system modelling. The software requirements will be defined iteratively in the course of this process. This is an area where much can be gained by further improvements.

Unlike hardware, where product status may be determined by direct inspection, the maturity of software is not directly assessable.

To make the software development process more transparent and better controllable, new methods and assessment criteria have to be defined for this process. In particular for the release and certification process of software it is common to use the representation of errors over time , supposing that a horizontal trace of the curve shall mean, that the software is errorfree to a large extent. Usually this curve is determined by a fixed test scenario, that means that changing the test conditions leads to drastic jumps in the trace of this curve. Critical for this process is the test coverage for the particular test scenario which is difficult to quantify.

The amount of software already developed for a fixed functionality can give only roughly predictions on the delivery date and on the correctness of the whole software package, because different software modules might have different degrees of complexity and consequently different test requirements.

Finally, it is very difficult to fix the memory size and the throughput of the used embedded computers at the beginning of the development process. The experience has shown that the above mentioned changes in software requirements and uncompleted software design at the point of procurement of the computing equipment lead to increased computing equipment sizes. The foreseen reserve capacities and an early risk assessment may partly limit these risks. Presently these features are not completely predictable. A better solution becomes feasible realizing scalable multi-processor architectures. These issues are discussed in more detail later on in the course of the paper.

To make the development risks more predictable and the development processes more transparent, above examples have shown that new assessment procedures should be introduced or existing ones improved. Furthermore, there is a need for improved communication structures including communication media, for the introduction of complexity measures and metrics regarding complex real-time software as well as early prototyping stages. Moreover, the development model should support an evolutionary growth of the system until its full functionality is reached. This approach will be accompanied by foreseen iterative steps. Herewith, the quality of anticipation of the development process within the overall planning determines decisively the degree of transparency of the overall development process.

The underlying structure of the international co-operation and the management of the whole development process are at least as important as technical aspects of the software development. Let us for example consider the development contract for the EF 2000 aircraft. The aim of this contract has been to develop a technically superior fighter aircraft within restricted budget, to guarantee compliance with national interests, workshare as well as to perform technology transfer between the nations. These aims are not free of contradictions. Participation of all partners in each system means for example that for the flight control system 23 different companies were developing parts of the software for the FCS. Nevertheless, they have produced the software based on the requirements developed within a single multinational joint team. The development process itself is an iterative process. Consequently, within international programmes having a sophisticated workshare an increase of the effort for the software management and for the agreement procedures among participating companies has to be accepted.

This approach is against experience also. It says, in order to minimize time and costs the responsibility of specification, implementation and testing of the software of a particular system or subsystem should be left in one hand.

Rapidly growing aircraft functionality due to software requires from all participants an expansion of intellectual and communication capabilities. This means, that besides general knowledge regarding classical aircraft disciplines, as for example equipment systems, flight control or avionics, additional knowledge in the areas of data processing, software-engineering, verification methods and validation procedures is required.

As a result, in addition to specialised technical engineers, "general managers" or "system integrators" with a general background are needed.

This specific kind of engineer is precious and consequently very hard to find within the companies. Indeed, it seems to be, that they hold the real key know-how, namely the "system integration capability".

Former experiments, to cultivate these special "system integrators" did not have the expected success. One reason could be that real project experience is a key factor in this process. In fact, this also could raise a problem in the future: Too little projects and as a consequence too few people with the know-how required.

2.3 System Test and Validation

The testing of software based aircraft systems comprises software integration tests, hardware/software tests, equipment tests, subsystem tests, system tests and tests in the aircraft itself. Due to safety and certification requirements this process is very expensive with regard to time, personnel and technical infrastructure. In order to optimize the processes and the available test equipment, the Military Aircraft Division of Dasa uses since several years a stepwise test concept with project specific modifications for different aircraft programs.

This concept is characterized by a structured sequence of test phases where the test object is subject to different test requirements and test phases. Stage A tests investigate modules of autonomous software functions and of operational flight programs within each single computer. These tests shall demonstrate that the implemented software contains the required functions.

Stage B tests are partial integration tests. They are performed with all computers, all software and all connections necessary for the functions of a multicomputer system. Whereas stage A tests can be performed in principle on a single general purpose computer, stage B tests are using aircraft equipment with a minimum of hardware environment. These tests are intended to demonstrate that the system software as a whole performs as required.

Stage C tests are the system integration tests and comprise all software and all equipment of the system involved. Software and equipment are tied together in an integration rig, e.g. the flight control rig for EF 2000. These rigs allow for the investigation of all interactions between the components of a system and for all external stimulations (signals, loads etc.). These tests shall demonstrate that the system delivers the required performance with the real aircraft hardware and that all fault conditions are covered. The results of these tests clear the way for the flight test to be performed in stage D. In this stage the whole system is tested and validated in the intended operational environment.

This process offers a high degree of transparency for the development engineers, the approval authorities and the final user of the aircraft. In case of equipment or software failures fast correction cycles are possible. Most software failures are discovered in early test phases where the effort for correction and retesting is relatively low.

Flight safety critical software requires additional measures in order to guarantee failure free performance, the integrity of the system and the required functions. In addition to the tests already described code walk through, static and dynamic code analyses and standardized conformity checks are performed. The support of the software development process by appropriate methods and tools is essential.

Experience has shown that software specifications, verification concepts and test procedures should be developed together early in the development cycle. Late known test and demonstration necessities mean time delays. The definition of test and verification concepts early on helps to avoid poorly defined specifications and to determine the effort for test and integration.

Flight safety critical software in particular imposes the question of the necessary test coverage for flight test approval. The number of possible states of complex systems is too large to test all possible system states and their environment conditions completely. In practice the experience of the system and software engineers decides about test effort, tool support and coverage.

Obviously there is a need for test strategies containing the demonstration of the completeness of the coverage of all essential system functions. In addition test redundancies should more easily emerge. Steps in this direction are still in the research phase. It can be seen however that these strategies and therefore automatic software tests are closely related to formal software specifications. These formal specifications offer the possibility to demonstrate the completeness of the specifications early in the development cycle and to define suitable test cases. These formal approaches are the most reliable ones, but complex systems offer difficulties with regard to applicability, efficiency and performance. Therefore today emphasis is still put on formal development processes rather than the product "safe software". Since safety critical software will find its way in more applications in the future and is expensive to develop, these approaches will be investigated more intensively in the future.

3.Automation

The current capabilities of sensors and computers allow for the automation for certain mission phases and operational procedures. Many examples are known in the civil and in the military aviation up to complete automatic systems like drones or missiles.

There are however distinct differences between the automation goals of civil and military systems. Common to both areas is crew workload reduction. Beyond this survivability and mission effectiveness in complex fight and threat scenarios are of paramount importance for military systems. Decision support, reduction of routine duties and the identification of alternative solutions are additional elements.

On the other hand, complex and fast changing tactical scenarios require high performance levels with regard to sensor data fusion, extraction of target data and identification of options. Therefore the limits of computer performance, software effort and verification capability are easily reached.

Certain mission phases have been automated relatively early, TORNADO low level flight for example. A next step in this direction has been the LATAN (Low Altitude Terrain And Navigation) experimental programme. The terrain following radar of the TORNADO aircraft had been substituted in this programme by a terrain following system based on stored terrain and elevation data. The objective was to demonstrate low level missions independent from an active radar and its radiation which could lead to the detection of the aircraft. LATAN basically consists of a computer with stored terrain data and uses the radar altimeter data to determine the terrain profile overflown. The actual position of the aircraft is then derived from the comparison between the stored and the measured terrain profile and leads to a change of the flight vector via the flight control system if necessary. The terrain data memory can be expanded for the automation of other functions. Applications in the civil area are possible as well.

Passenger aircraft however are flying along fixed routes, are subjected to the laws of economy and direct operating costs and are guided from controllers on the ground. Since these basic factors underwent only slight changes in the course of the years, the development of civil aircraft has been rather evolutionary in nature. New variants are derived from standard models rather than from revolutionary new approaches. Therefore experiences from development and operation of in service aircraft can be used to a large extent for new systems. Automation is more consequential since it is possible to advance step by step from one aircraft generation to the next. Since routes, airports, approaches and departures are fixed to a large extent automation can be driven quite far.

"Crew Assistant" is one of the concepts where these objectives are investigated currently. The comparison between tasks and human capabilities (Fig. 3) already gives some hints which functions can or should be automated. With appropriate data bases, information processing and decision support functions on board of the aircraft it seems to be possible to present to the crew the relevant information only and to automate several processes.

Advanced information processing capabilities are also useful for extended test and diagnosis capabilities. Improved fault localisation processes on the basis of all available information and intelligent expert systems lead to reductions of false alarm rates and turn-around times, since precise informations about faulty parts are available even before the aircraft has landed.

4. Future Aspects

4.1 System Prototyping

Experiences from various development programs have shown that electronic systems in aircraft can no longer be dealt with in terms of size, weight and cooling power. It is also no longer sufficient to discuss or to specify isolated criteria such as

detection ranges or bandwiths. The process of system definition finally leading to equipment and software specifications must include early on an analysis of the complex and interacting real time effects of integrated avionic and flight control systems.

Since user requirements quite often are not specific enough or can lead to misunderstandings in particular at the beginning of a new program, effective communication means for the dialogue between users, system and software engineers have to be found in order to derive well defined specifications.

With these goals in mind Dasa's Military Aircraft Division in Ottobrunn installed within the last years a facility called "System Prototyping Rig" which fulfills a similar function as wind tunnels for the airframe development. The main objectives of system prototyping can be described as follows:

- Empirical investigations of new system architectures and their interaction phenomena in real time
- Feasibility studies including the rapid prototyping of software
- Optimization of the man-machine interface
- Investigations of new equipment in a realistic system environment
- Definition and evaluation of critical real time algorithms (e. g. sensor fusion)
- Investigation of data transfer processes between real or simulated equipment.

From these objectives the actual set up of the system prototyping rig is derived. It mainly consists of micro computers, graphics workstations, mission computers and aircraft equipment and comprises various displays and a full functioning cockpit. The hardware can be interconnected in very flexible ways via different data bus systems and allows for the representation of different avionic system configurations. The software installed provides aircraft models, sensor simulations, the development environment for all computers and a powerful test system.

Aircraft components and the realistic system environment allow for accurate estimations of the development risk in early phases of the development whereas pure simulations are either very expensive or limited to specific areas. System prototyping is typically performed in parallel to the specification process and supports critical early design decisions.

This approach represents a "front end investment" in order to reduce technical risks, to deliver the required quality of the product and to derive trustworthy time schedules and budgets for the whole development process. With regard to software development it should be noted that time spans for coding and testing can be predicted fairly good whereas the time to derive unambiguous software specifications and to remove remaining errors was much more difficult to take into account in many programs.

This process is related to the total quality management approach with regard to risk orientation at the systems level, user involvement and continuous process improvement as essential elements.

Additionally prototyping as a system simulation has to be integrated more in the system development process. In the future prototyping results will form the basis for automatic software code and test case generation in order to reduce significantly time and effort spent for the system development.

4.2 Modular Avionic Structures

Current avionic systems can be characterized as a set of loosely coupled subsystems (equipment) confined to realize exclusive, predetermined functions and connected via relatively low speed data busses. Standardization occurs at a very high level, that is at the level of the interconnections of the Line Replaceable Units (LRU's) building the functions. Each equipment requires its own maintenance concept and offers only limited room for growth. New systems therefore require new equipment developments and are expensive due to low production runs and the unavailability of shared ressources.

Farther reaching requirements such as situational awareness and cockpit automation as well as software reuse, adaptability to new operational requirements, maintenance free operation over longer time spans and reduction of life cycle costs are difficult to realize with current avionic architectures.

Advancements in technology, in particular in the areas of microelectronics, fault tolerance and software have enabled the avionics and aircraft industry to develop new design concepts which result

in highly integrated digital avionics under software control. This approach, generally known as "Modular Avionics", is centered around an open systems architecture based on powerful computers with an operating system that allows independent processing of application software while maintaining robust partitioning between the software modules for even the most critical functions. These computers or processing modules are housed in a rack or cabinet together with several other hardware modules such as mass memory modules and power conditioning modules and are interconnected via standardized high speed data networks. The modules itself obey standardized form factors and interfaces and therefore represent an open avionic systems architecture with room for additional developments by different manufacturers. The interface to the outside world such as transducers, sensors, actuators, displays, controls and radio frequency units is handled by standard high speed optronic links. This type of system architecture allows for new levels of fault tolerance by the reconfiguration of tasks among the different modules.

Reconfiguration mechanisms also allow for the maximum use of processing power in each mission phase since not all system resources are necessary in all phases of the mission. Updates are relatively easy since the architecture is scalable and new tasks are realized by reconfiguration or additional processing modules. Shared development costs, larger production runs, lower unit costs and reduced maintenance costs should lead to lower life cycle costs.

These concepts are supported in Europe by initiatives like EUCLID CEPA 4 and ASAAC Phase II and are already in an advanced stage in the US. It can be expected that these new architectures will influence the development of all new avionic systems beyond the year 2000.

System Development Model

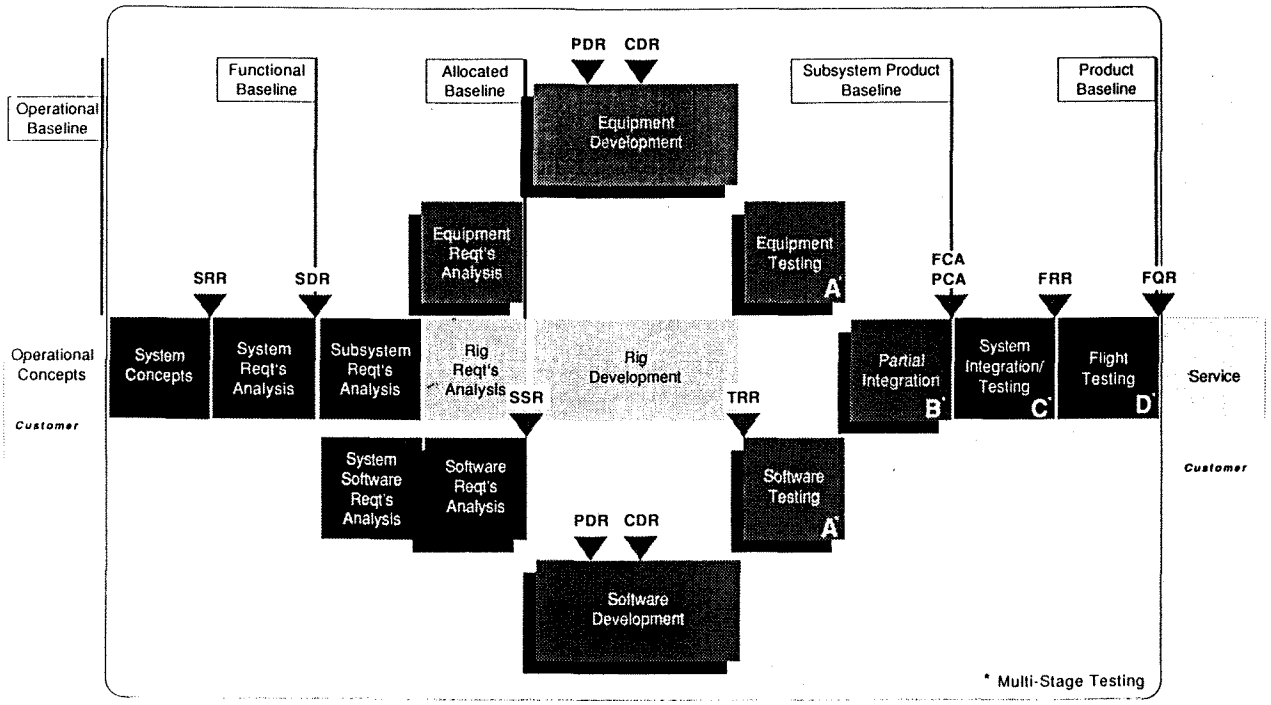

Fig. 1

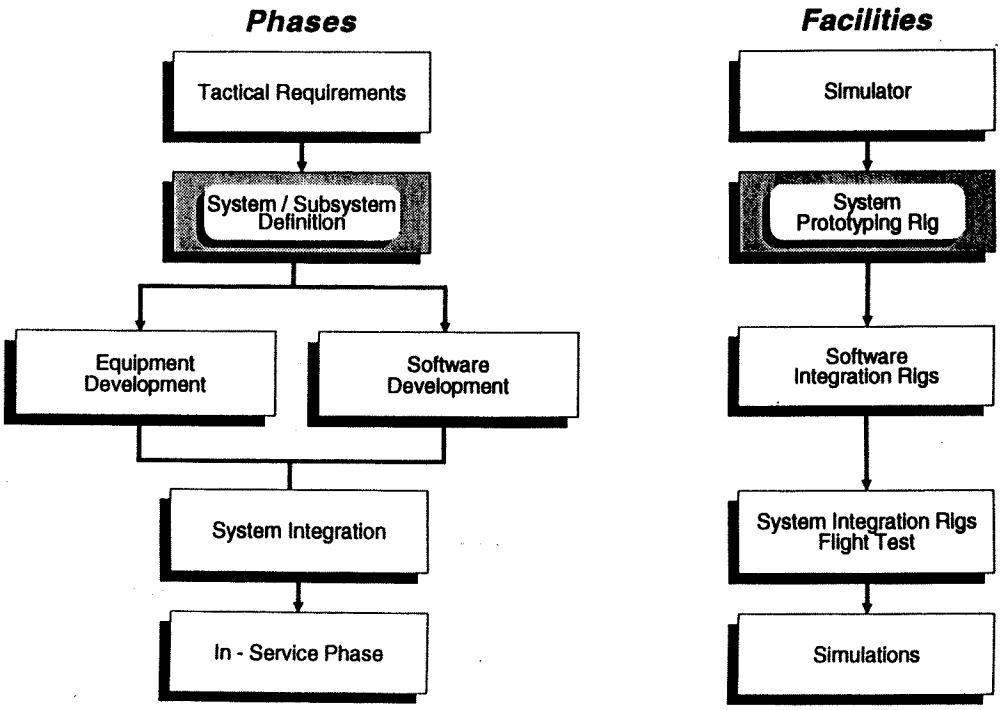

Fig. 2

"HUMAN SYSTEM" CAPABILITIES

TASK	ADVANTAGES	DISADVANTAGES
Regulator Actions continuous generation of commands according to A/C states	o highly adaptive o fault tolerant o nonlinear	o small bandwidth o low speed o few states o high deviations
Rule based actions decisions made according to previously learned rules	o adaptive o priority based o fault tolerant	o low speed o small logical depth o workload dependent
Knowledge based actions decision made according to rules which are created during actions by applying knowledge	o ultra high amount of knowledge o self adapting o fault tolerant o superior pattern matching o online learning	o fuzzy knowledge o experience dependent o workload dependent o high influence of unconcious knowledge

Fig. 3