THIRTEENTH EUROPEAN ROTORCRAFT FORUM

59
Paper No.24

AN INTEGRATED APPROACH TO AIRBORNE SOFTWARE DEVELOPMENT

E. CAMBISE

DATAMAT S.p.A., Italy

S. GAZZILLO

AGUSTA SISTEMI, Italy

September 8-11,1987
ARLES, FRANCE

ASSOCIATION AERONAUTIQUE ET ASTRONAUTIQUE DE FRANCE

AN INTEGRATED APPROACH TO AIRBORNE SOFTWARE DEVELOPMENT

E.CAMBISE
DATAMAT S.p.A.
Via S. Martini 126
00143 Roma, Italy

S.GAZZILLO
AGUSTA SISTEMI
Via Isonzo 33
21049 Tradate (VA), Italy

1 Abstract

The integrated approach used in the development and testing of the Mission SW of the Mission Avionic System (M.A.S.) for the Italian Navy EH101 ASW/ASUV helicopter is discussed, jointly with Paper N.25. The M.A.S. architecture is centered around a 1553B Bus connecting a Computer (Mission Computer Unit), on which the Mission SW is resident, to the other Avionic Subsystems (sensors and operator's interface devices). The MCU is dual redundant to improve the survaivability of the M.A.S. to failures during the Mission exploitation. The Mission SW implements a centralized interface for all the Avionic Subsystems and performs all the Tactical computations.
The approach addresses all the stages of a System/Sw project's development cycle namely:

a. requirement definition and detailing
b. requirement analysis and SW Design
c. SW development, testing and Computer HW integration
d. Support to the Mission Avionic System Ground integration

2 Foreword

Any development of Systems related to flying crafts has to cope with a number of constraints and there are many challenging aspects to be dealt with.
According to the choosen viewpoint, the Hardware, Software or Logistical aspects, the Technical, Managerial or Methodological issues are considered the more relevant.
Whilst the HW/SW and System integration aspects of the EH101 Avionic System are described in Paper N. 25, this paper, given a short outline of the technical characteristics of the System, will concentrate on the different phases of the System Software Development and on the methods utilized to improve the quality of the delivered product, in order to approach the following definition:

"A System is the contemplation of order achieved by directions of its implementation" [1].

The EH101 System Software Development is partitioned into a Flight Critical development (the Aircraft Management SW), carried out by AGUSTA SISTEMI, and into a Mission Critical development (the Mission SW), carried out by DATAMAT.

The two SW systems have been configured in order to exploit the maximum commonality at all levels, namely they use the same:

- HW of residence (a Multiprocessor Computer based on the i80286 Processor) [2]
- Operating System (Frame Driven Deterministic / Statistic Scheduler) [2]
- Design, Development and Documentation tools and methodologies (SADT, PDL, HOL Programming, UDF)
- Testing and integration philosophy (Host/Target testing, Subsystem and Overall Integration Rigs)
- Specification through simulators (Cockpit Simulator and Mission Simulator)

The Mission SW development has been taken as guideline for the discussion, because it is better suited to highlight critical aspects in the high end of the SW development (the Requirements definition) and in the capacity of accomodating changing requirements during both the development phase and the operational life, due to its intrinsic characteristics; in fact, as any Command and Control System SW, it is dynamic (many interactive I/Os), as opposed to static (batch/serial); it is concurrent (multiple simultaneous processes), as opposed to sequential (single process); and it is decisional (high man-machine interaction, query-response driven), as opposed to computational (algorithm based) [3].
Furthermore all the topics covered apply without any modification to the Aircraft Management SW.

3 The EH101 Avionics Architecture

The Avionics Architecture of the EH101 is built upon a couple of dual redundant Mil Std 1553B Busses (fig. 1).

Each Bus System is controlled by a Dual Redundant Computer Unit:

- the Aircraft Management System (AMS) controls the subscribers of the "Basic Bus", typically sensors and subsystems related to the monitoring and control of the Helicopter flight.

- the Mission Computer System (MCS) controls the subscribers of the "Mission Bus", the sensors and subsystems related to the operational mission of the helicopter.

The MCS is also a subscriber of the "Basic Bus", acting as a Gateway between the two avionics systems.
The two systems implement the EH101 overall Avionic System.

The sensors and subsystems related to the flight can be summarized as: helicopter plant monitoring equipments (thermocouples, pressure sensors etc.), environmental and navigation sensors (Inertial Reference Units, Air Data System, etc.) and voice communications.

The primary helicopter missions are Anti Submarine Warfare (ASW) and Anti Surface Vessel Warfare (ASUV), therefore the related mission equipments are active detection sensors (Radar/Sonics), passive detection sensors (Electronic Warfare), communication equipments.

The interface among the crew members (pilots and tactical operators) and the various subsystems is carried out through multifunction terminals (Common Control Units), common to both Systems, and the relevant data are presented to the pilots on the Electronic Flight Instrument System and to the operators on Raster Graphical Displays.
The pilots act on CCUs directly connected to the "Basic Bus", whilst the tactical operators act on CCUs directly connected to the "Mission Bus".
Because the man machine interface of the overall Avionic System is centralized and implemented by software resident in the AMS and MCS, and an interface exists between the two systems, any operator can access the functions resident in the "remote" system.

4 Mission Software Tasks Summary

The main tasks that the Mission Software (the SW hosted in the MCS) has to carry out can be summarized as follows:

- Management and control of the Operator's Dialogues
- Generation of the Graphical Presentations
- Management of the Interface and Data Acquisition from the Mission Subsystems and the Aircraft Management Computer
- Computations related to the Tactical Functions
- Monitoring of the Mission Avionic System Configuration

5 Key aspects of the Development

As it can be seen from the outline of the tasks to be accomplished, the Mission SW, as all the similar software systems, has to implement (see fig. 2):

- the Kernel of the high level functions (the integrated ones) required to the System
- the interfaces to both the physical and human environment in which the System has to operate

When these kind of systems are embedded in a platform (the EH101 for instance) in which quite all the components have to be developed in about the same timeframe, a task becomes essential for the developers of all the component subsystems, and mandatory for the developers of the "System Software":
the identification, detailing and formalization of:

- the user requirements in terms of :
  - functions to be performed (allocated to HW or SW)
  - man machine interface (dialogues and presentations)
  - automation level (as a tradeoff between operator offloading and human control upon critical procedures)
- the physical interface requirements in terms of:
  - HW supplied functions (as a tradeoff between HW and SW performance and development costs)
  - Subsystem interface protocols (as a tradeoff between a straightforward system approach and consequent modifications to existing equipments)
  - Testability of the resultant architecture (to assure a unambiguous identification of the source of errors during the integration phase)

Moreover, the ways in which the platform is used in operation is obviously affected by the performances of the different components and by the level of integration with respect to previous crafts used by the users, and becomes therefore essential a thorough understanding of the overall performances and behaviour of the system.

To support the task of determining the requirements, it is common ground that documents have to be produced with a high level of "logical" structuring but using a 'natural' language (as opposed to 'formal' languages) the closest to the user understanding; i.e. the statements have to be expressed in terms of user actions and user perceivable results instead of 'tasks', 'data stores', 'variables', 'scheduling frequences' and so on.

It is common, however, that the writers of such documents (the developers) and the readers (the users), that have to assess the document contets to obtain a product that can be satisfactorily used by them, speak different idioms of the same language.
Moreover, the mental images derived from a written document are always filtered by the past experiences of both the writer and the reader, and then, it is almost sure that even agreed documents will be subject to ambiguous interpretation and therefore will result in implementations deviating from the user needs.

To overcome this underlying problem it has been choosen to develop a "Mission Simulator" (Mission Sw Development Rig, MSDR) to give the user (the 'evaluator') the evidence of what he will get from the System, as soon as possible after the, still necessary, production of the 'Contractual Specification Documents'.

Once the "System Requirements" have been identified and detailed, for the "System Software" developer starts an equally foundamental phase:

- the formalization of the Software requirements and their allocation to the elements of a SW architecture.

To analyze the detailed requirements produced by the first activity, the Structured Analysis and Design Technique (SADT) has been used [4], that allows a formalization of the requirements using a top-down decomposition of them (see fig.s 3,4,5 and 6).
The use of such a methodology has been supported by a tool developed by Datamat (MASTER) that provide for completeness checking, diagram drawing and analysis documentation production.
The use of SADT leads to the definition of the functional subsystems in which the System SW can be partitioned, minimizing the coupling between the different SW subsystems.

The identified functions are then allocated to SW components (processes, interfaces, data stores) that constitute the architecture of the system.
The elements of the architecture are derived from the MASCOT methodology as implemented by PERSPECTIVE (a tool produced by SD), that provides the main functions of a "SW FACTORY", i.e. Compiler, Debugger and Configuration Control.

The steps reported above are what can be called "a theoretical approach" to the Software development, that has its ground on the 'waterfall' model (see fig. 7) of the SW development [5], that will be supported by other 'SW Factories' currently under development [6].

Even though this model has proven succesful in a number of projects and has formalized the steps to be performed, and therefore controlled, to assure a Quality SW Development, it doesn't reduce the risks associated to a poor or ambiguous definition of the SW requirements, and furthermore, this one-dimentional conceptualization of the SW development process fails because, once a stage has been started, it ends only when the SW is replaced or retired.

This risk areas are particularly relevant in the development of systems that by their nature have to be developed and documented in a tightly controlled manner, implementing a rigorous change control policy and formalization of the intermediate baseline documents, resulting therefore in a longer development timeframe and in a reduced flexibility with respect to projects of less criticalness. These projects in fact tend to reach the implementation phase after a long sequence of formalized steps, and only in this phase the detailed design can actually be verified against the 'reality' of SW implementation. The result is that ambiguously defined requirements tend to be discovered at a late stage of the development, incurring in higher costs and delays in their fixing.

To overcome this problem, it has been choosen to exploit the same tool (the MSDR) utilized to detail and evaluate the overall requirements; this tool is in fact used to allow a "rapid prototyping" of the functions to be implemented by the target System (functional rapid prototyping), according to an evolutionary approach to the development of complex avionic systems that can be found in other similar programmes [7].

In this way, that can be called "a pragmatical approach" to SW specification, the specification documents, produced by the Requirement Specification activity for the 'Embedded System', are used to quickly implement SW functions, that even though written in a different language (FORTRAN instead of Pascal) and in a different environment (a large commercial minicomputer instead of a Mil Spec Multiprocessor, see fig. 12), lead to identify the ambiguous definitions or the missing requirements.

This Concept of "rapid prototyping" can obviously be applied to the other critical area that is the one of the physical interface among the different HW subsystems, with a quick implementation of interface protocols and a raw simulation of the expected subsystem behaviour.

The effort for reaching this objective is the natural complement of the implementations needed to comply with other SW development requirements like:

- a thorough SW testing carried out by the SW supplier independently by the actual availability of the HW subsystems

- the capability of monitoring the actual integration of the produced System SW with the Host Computer (the AMS/MCS) and with the HW subsystem prototypes or emulators

The philosophy, the practice and the actual implementation of the EH101 Avionic HW/SW Testing and Integration Policy is described in detail in Paper N. 25.

6 The Requirement Analysis Tool (MASTER)

MASTER is a propretary tool of DATAMAT S.p.A., and it has been used to support the Requirements Analysis activity.

The tool gives support to the following models of the system:

- a requirements model, implemented by the list of the requirements of the system, each with its identifier and a list of keywords that can be used to classify the requirements from various points of view [8];

- a functional model, based upon the SADT actigrams (see fig.s 4,5 and 6), but with extensions and modifications needed to exercise a sufficient level of control upon the correctness of the model [4];

- a data model, based on the widely used concepts of conceptual data modeling and obtained from the Entity-Relationship model [9];

- an events model, used to describe the dynamic behaviour of the system with a level of abstraction comparable with that of the other analysis models;

- a "language independent" architectural model, with the purpose of giving a first, high-level description of the architecture of the system to be developed; this model uses two distinct types of components to represent the active and the passive modules of the system, and has been designed to address the most common languages and models for the development of real-time systems (for example the MASCOT approach used in PERSPECTIVE, or the Ada one).

The support offered by MASTER focuses on the following aspects: interactive data handling, consistency and completeness checks, automatic design, documentation production.

Interactive data handling covers all the operations related to the introduction, retrieval and modification of the project information, and is handled by means of forms; of such forms, some (one for each type of object in the project library, where objects can be functions, data entities, software components ...) allow the specification of the information concerning objects; from such forms, it is possible to input an extended description of each object by means of an automatic invocation of the system editor.
Other forms allow to specify links between different objects, or to perform other, more specialized interactions.
During forms interaction the immediate checks on the consistency of the data, allowed on-line, are performed.

Consistency and completeness checks covers the production of a list of all the errors that are present in the project library; typical checks performed are checks on the inheritance of dataflows between SADT functional diagrams, completeness of the coverage of the requirements by the analysis model and so on.
These checks complement, and in some cases duplicates, the checks performed on-line during form interaction.

Automatic design represents one of the most innovative aspects of the tool: the steps needed to obtain an high level architectural design from the analysis data have been pointed out, togheter with standard rules for each of these steps.
For each of these steps, MASTER has an automatic procedure that applies the standard rules for the step; obviously such rules cannot cover all the possible situations in a design, and the user can influence the resulting design in different ways:

- by attaching additional information to the elements of the analysis; such information is used by the automatic design procedures as part of its rules;

- by performing manually part of the work involved in a design step: the choices of the user are not altered by the automatic procedures;

- by modifying the results of a step to suit the particular needs of the project: the automatic procedure must be reexecuted after the modifications (for consistency purposes) but, again, the choices of the user are not altered.

The principal goal of this approach has been that of allowing the maximum possible flexibility to the designer, meanwhile avoiding the need for routine work.

Documentation production covers the production of various types of documents from the information contained in the project library; there are documents for each of the models supported by the tool, that can be freely composed to suit the user needs; other documents are specifically oriented to the SADT functional analysis, giving all the information related to each of the diagrams.
Apart from these documents, that are organized in chapters and sections containing the description of the objects, and are processed by a text formatter, there are two additional kinds of documents: the graphical printout of the SADT diagrams, produced with a plotter using algorithms that optimize the intersections between dataflows, and synthetical reports of various kinds, that help the user during his work giving summaries of the various aspects of the analysis and design.

All the described functions (and also, for example, all the administrative operations of the tool) can be activated by a tree-structured menu.

The tool is built upon a Relational DBMS; all the project information is contained in the relational database, except for the files containing the description of the objects; the tool can manage more that one project in its database, and for each project can handle multiple simultaneous users, handling all concurrency problems.

7 The SW Factory

The SW Factory is an HW and SW System which configuration is reported in fig 8.

By a SW point of view, the Factory is centered on PERSPECTIVE, which is a proprietary tool of System Designers plc, that provides a multi user Programming Support Environment (PSE), for the development of embedded Sw using an extended ISO Pascal (Concurrent) as HOL. The tool has been utilized also by other developers in similar programmes [10].

The tool supports a modular technique, based on the MASCOT [11] approach, which decomposes the SW system into Processes, Modules and Interfaces.
The Design process is supported by a diagrammatic notation (see fig. 9).
The Processes are the components that implements the actual 'parallel processing' and communicate with each other trough the Interfaces.
The Interfaces implement a Data Hiding concept, offering only controlled access to the data structures.

The Modules, that implement the Interfaces can use other Interfaces implementing a SW layering concept.

For the peculiar AYK204 target, the Run-time Support of the Pascal has been tailored to access the primitives of the Operating System managing the target HW resources.

The key element of the PERSPECTIVE PSE is the multi-user Data Base in which the source code files, and all associated derived products composing a SW System, are contained.
The Data Base is configured in independent user domains allowing an concurrent, and controlled, development of the SW components.
These components may exist in different versions and facilities are provided to keep track of versions, relationships, status, access rights etc.
In this way the basic functions of the SW Configuration Control are supported.
DATAMAT inserts in the Data Base the PDL that implements the Detailed Design of the code to be produced and process this information to produce the SW Development Documentation.

SW development within the PSE is carried out in two major phases, Host development/testing and Target testing.
During the first phase the SW components are compiled and debugged through simulation on the Host computer.
Checkout facilities are available to enable source level code debugging.
Components are then compiled for the target and downline loaded to it, where can still be debugged via an Host terminal.

The tool has been improved, to accomodate Agusta requirements, adding an Assembly level debugger and modifying the breakpoint technique, in order to minimize the distortion to the SW real time evolution.

Because, in any case, the SW Debugging alters the real-time performances of the target SW, an In Circuit Emulation facility has been configured, that allows a full speed, parallel processor debugging, even for time critical events.

To assure a thorough SW testing an MCU Rig (detailed in Paper N.25) has been configured in the SW Factory.

8 The Mission SW Development Rig

The MSDR is a System based on commercial computers and commercial equipments or specifically built mock-ups of the components implementing the EH101 Man Machine Interface (see the artist's view of fig. 10).
The logical components of the MSDR (see fig. 11) are:

- the Cabin and 'Cockpit' Mock-ups, containing the mock-ups of the actual consoles, CCUs and Displays.
- the Supervisor and 'Cooperating team' positions through which the scenario is altered and the simulated helicopter configuration is affected.
- the Modification interface, through which the configuration of the SW appearence of the Operator interface is modified.
- the Processing facilities that implements and support the Mission Simulation.

The HW configuration (see fig. 12) is centered around two Vax computers, containing the Scenario and Sensor simulation SW (system A) and the MCU and M.A.S. simulation SW (system B).

The 'Mission Bus' is, only at a logical level, simulated by an intercomputer link (Ethernet).

The Display Generators are commercial equipments that emulate the on board Common Waveform Generators mainly in terms of the Display appearence, but with a great flexibility in their setup, to allow also the evaluation of alternate solutions.

The Common Control Units mock-ups have been built by Datamat, and are based on HW 'dumb' mock-ups and a SW simulation of the actual CCU on a commercial Single Board Computer, developed and tested using the same SW Factory of the Mission SW.

The Software, that actually implements the MSDR functions, has been designed and is currently under development according to the following requirements:

- Extended modularity
- Incremental growth capability
- Maximum flexibility

The above requirements are all aimed to synergistic objectives that can be summarized as follows:

- The possibility of a significant development in a reduced timeframe

- The capability of supporting evaluation session at intermediate levels of the development, in order to supply the relevant inputs to the Mission SW Development program with the correct phasing
- The capability of allowing on-line changes to the appearence of the operator interface during the evaluation sessions
- The mimimization of the time required to implement the changes that, by their impact on the system, can not be executed on line

The above objectives have been reached basing the architecture of the MSDR SW on a Relational Data Base that manages the setup information for the different functions of the system (scenario simulation, operator dialogues and presentations etc.).

This data base is accessed by Fortran programs that actually implement the above functions and that provide the real-time response of the MSDR.

As detailed in paragraph 5, the MSDR, supports the most innovative life cycle models [12]: being designed and developed using an **incremental development** approach, it supplies to the Mission SW Development both advantages of **rapid throwaway prototyping** (the early investigation of shadow areas) and of **evolutionary prototyping** (the early and easy implementation of well known areas).

9 Conclusions

To have the possibility to succeed in the Complex Systems Development arena, it is necessary to utilize an hybrid approach:

- to use **methodologies** and **tools** helping in the different design and development phases

- to apply the **rapid prototyping** to the requirement definition

To assure the customer that the proposed approach is viable and useful by an operational point of view, the hybrid approach is still worthy:

- the use of **methodologies** guarantees the Customer about the quality of the product

- the use of **rapid prototyping** guarantees the Customer that the product is feasible and answers to his needs

10 References

1. R. G. De Sipio, The art of Avionic System, Defense Science & Electronics, August 1986

2. A.di Giovanni and P. Garro, H/W and S/W redundancy techniques for 90's rotorcraft computers, XI European Rotorcraft Forum, paper No 60, London, September 1985

3. P.C. Jorgensen, Tutorial on requirement specification, Computer SW and Applications Conference 86, IEEE 86CH2356-4, 1986

4. D.T.Ross and K.E. Shoman Jr.,Structured Analysis for Requirements Definition, Second International Conference on SW Engineering, October 1976

5. B.W. Boehm, Software Engineering, IEEE Transactions on Computers (C25), Pages 1226-1241, Dec. 1976

6. M. Slissa and P. Laroche-Levy, Atelier de conception de systemes avioniques et de realisation de logiciels embarques, AGARD Avionic Panel Symposium, Paper No 36, Las Vegas US, April 1987

7. P. Schirle, Maquette des specifications fonctionnelles du logiciel embarque - experience du systeme avionique Rafale, AGARD Avionic Panel Symposium, Paper No 10, Las Vegas US, April 1987

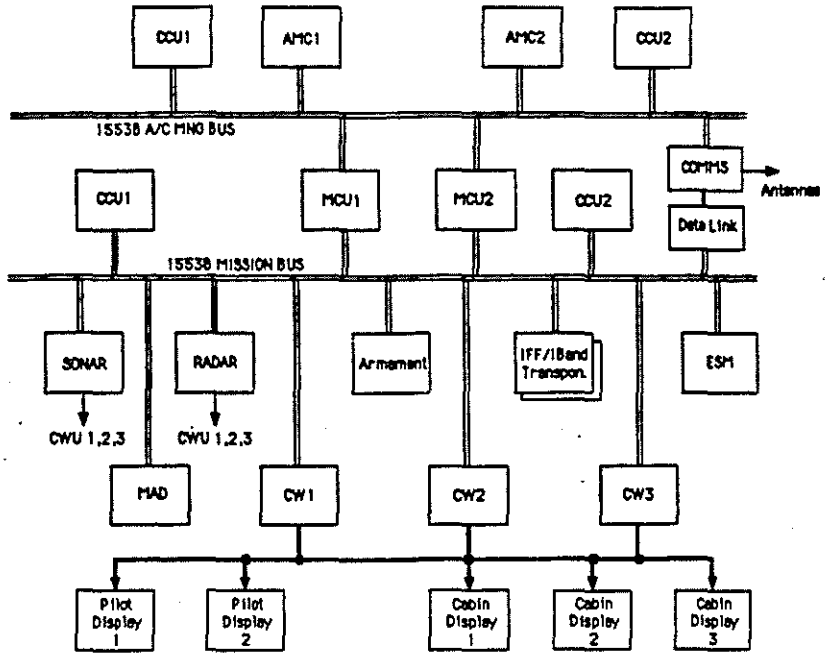
8. T. De Marco, Structured Analysis and System Specification, Yourdon press, 1978

9. Chen, The Entity Relationship Model - Toward a unified view of data, ACM Trans. on Data Base Systems, Vol. 1 N. 1, 1976

10. W.E.R. Kellaway, SW Engineering for the British Aerospace Experimental Aircraft Programme (EAP), AGARD Avionic Panel Symposium, Paper No 32, Las Vegas US, April 1987

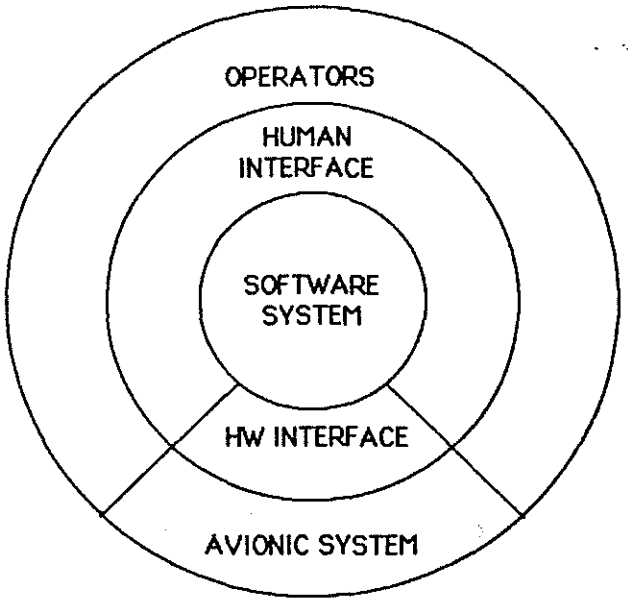
11. Royal Signal and Radar Establishment - Malvern, The Official Handbook of MASCOT, Issue 2, March 1983

12. E.H. Bersoff, B.J. Gregor, A.M. Davis, A New Look to the C3I Software Life Cycle, SIGNAL, April 1987



MAS Architecture

Figure 1



The Total System

Figure 2

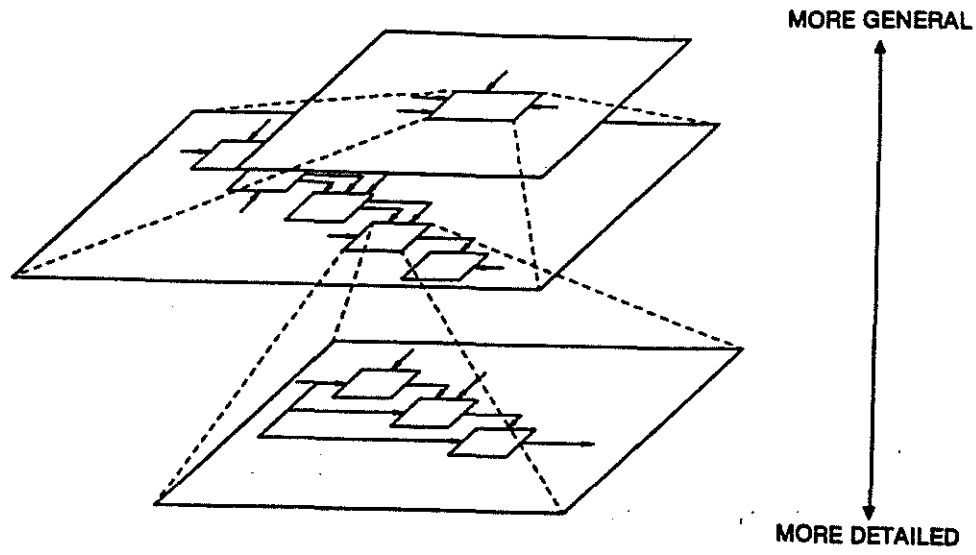


Figure 3

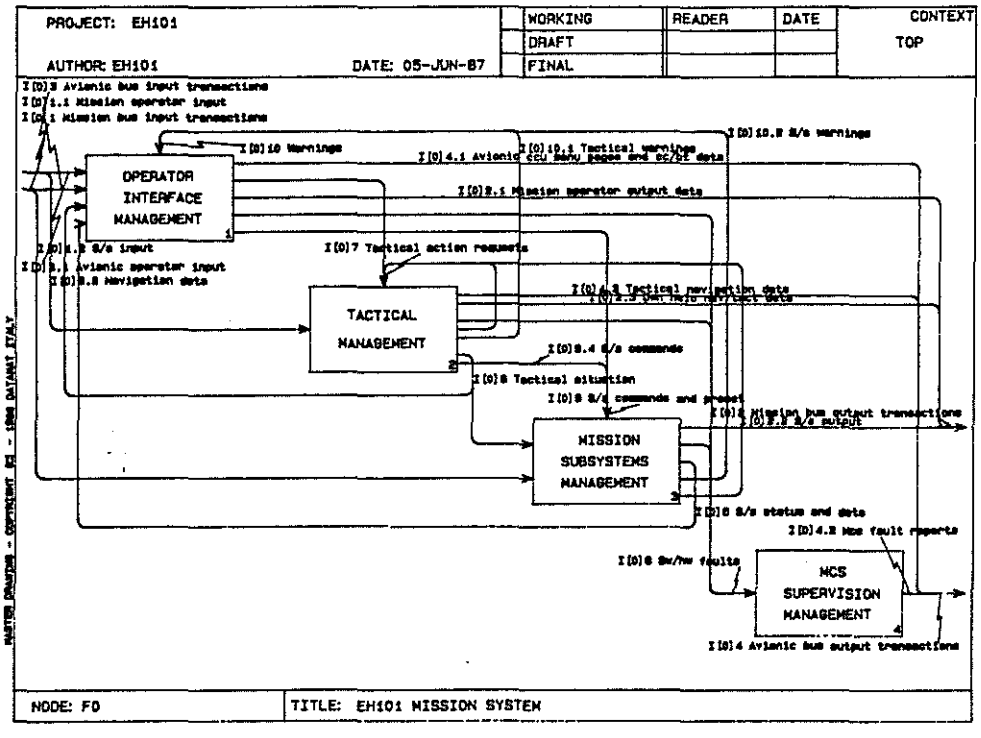


Figure 4

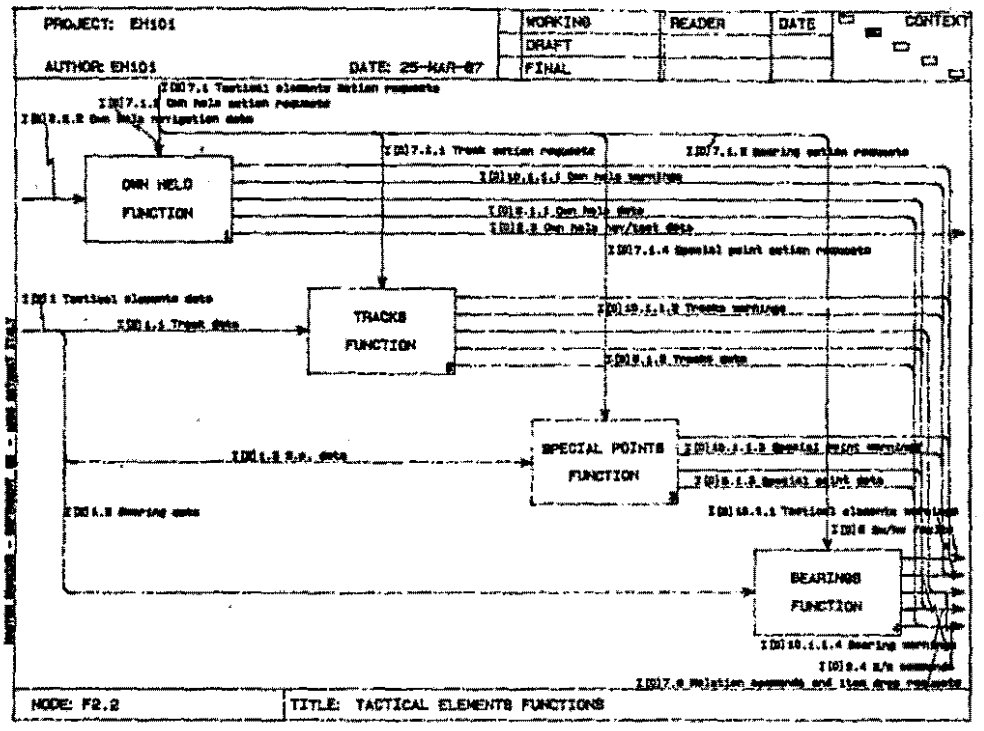


Figure 5

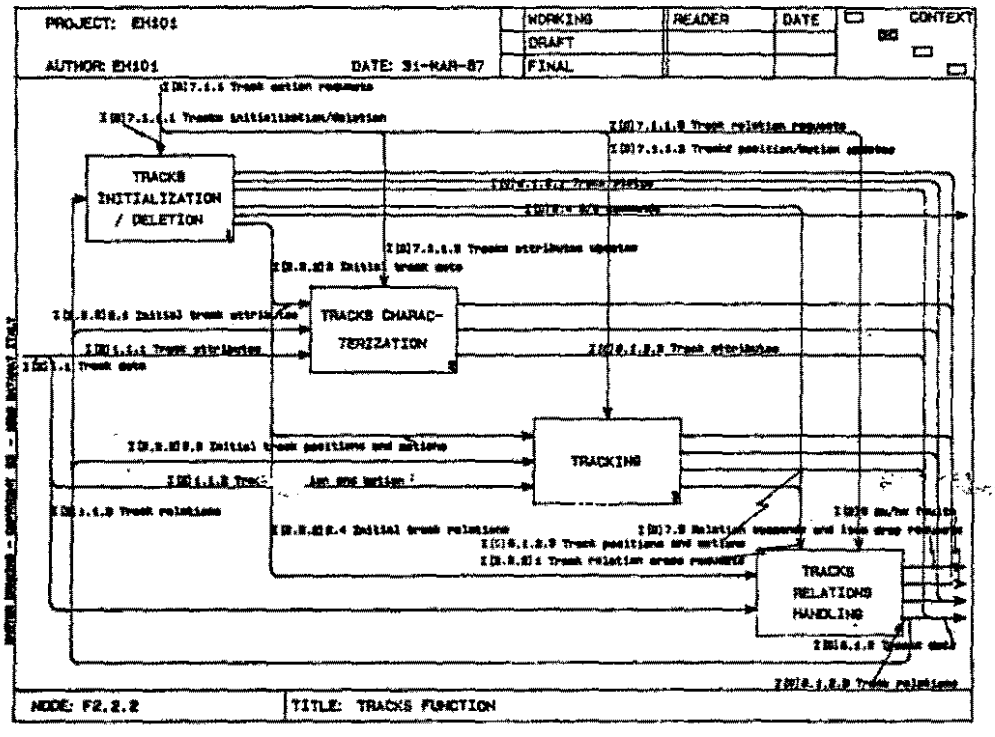
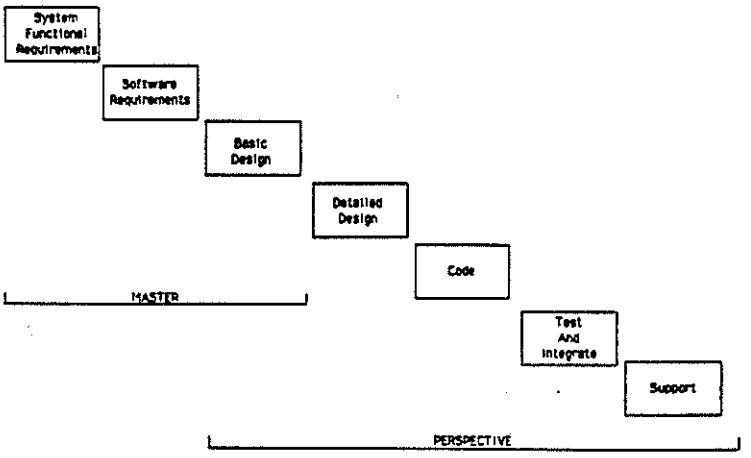
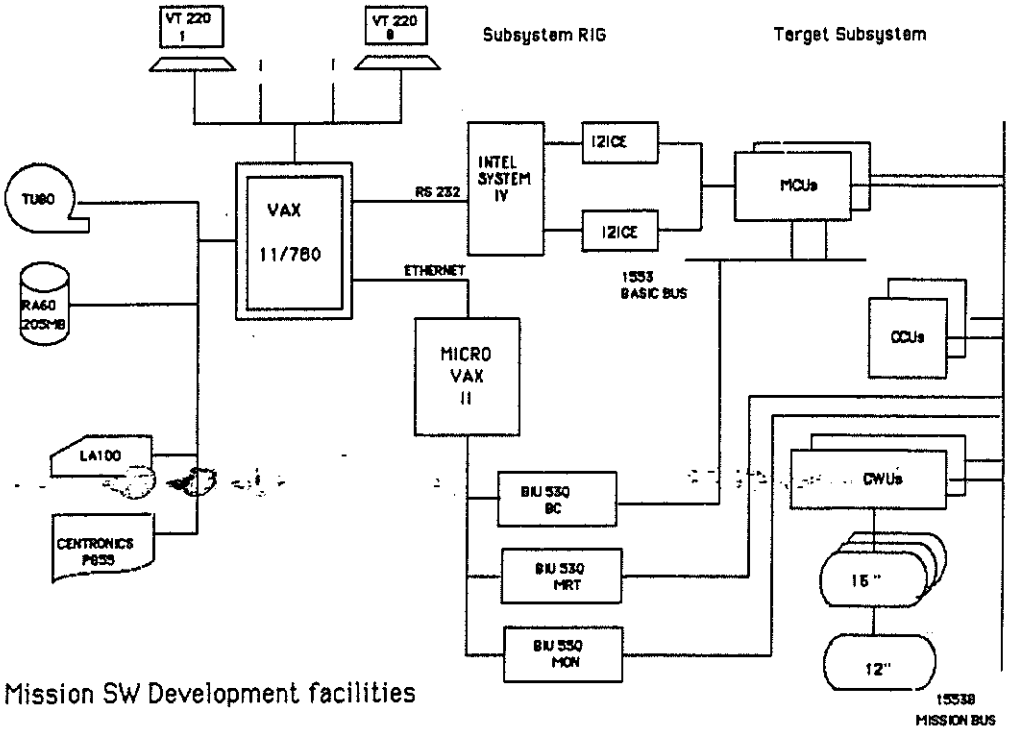


Figure 6



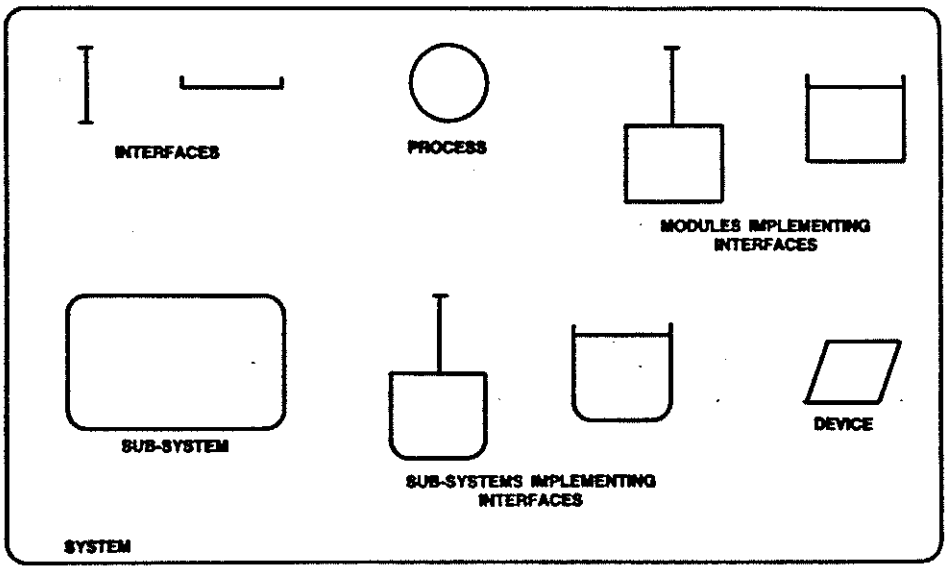
SOFTWARE DEVELOPMENT WATERFALL MODEL

Figure 7



Mission SW Development facilities

Figure 8



PERSPECTIVE NOTATION

Figure 9

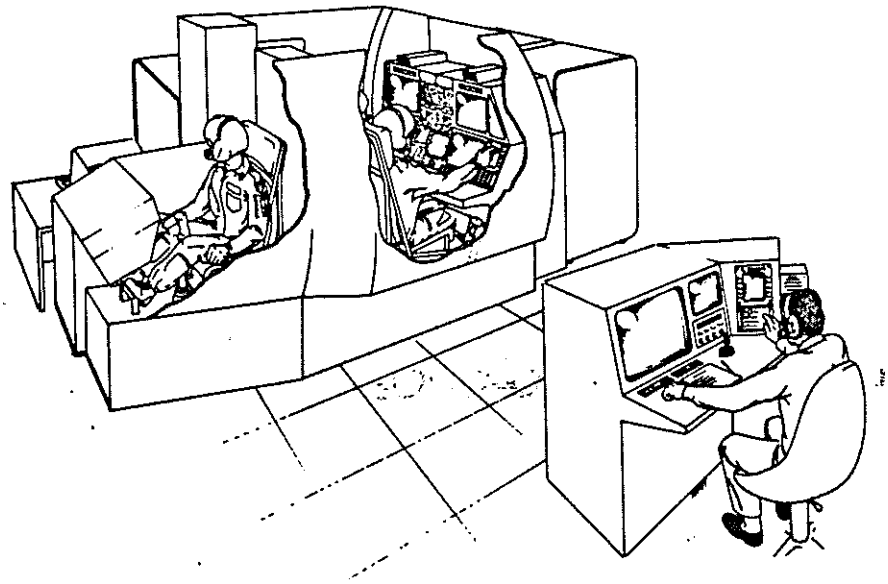


Figure 10

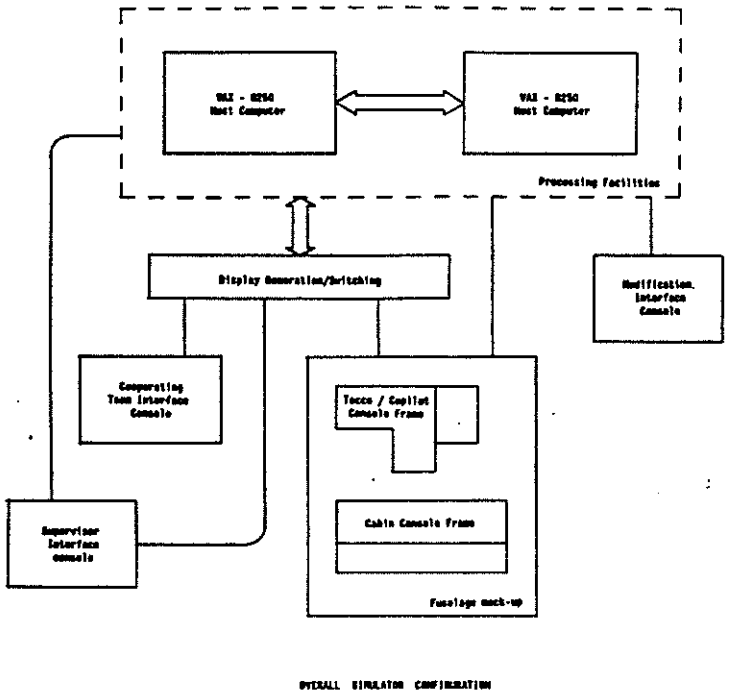
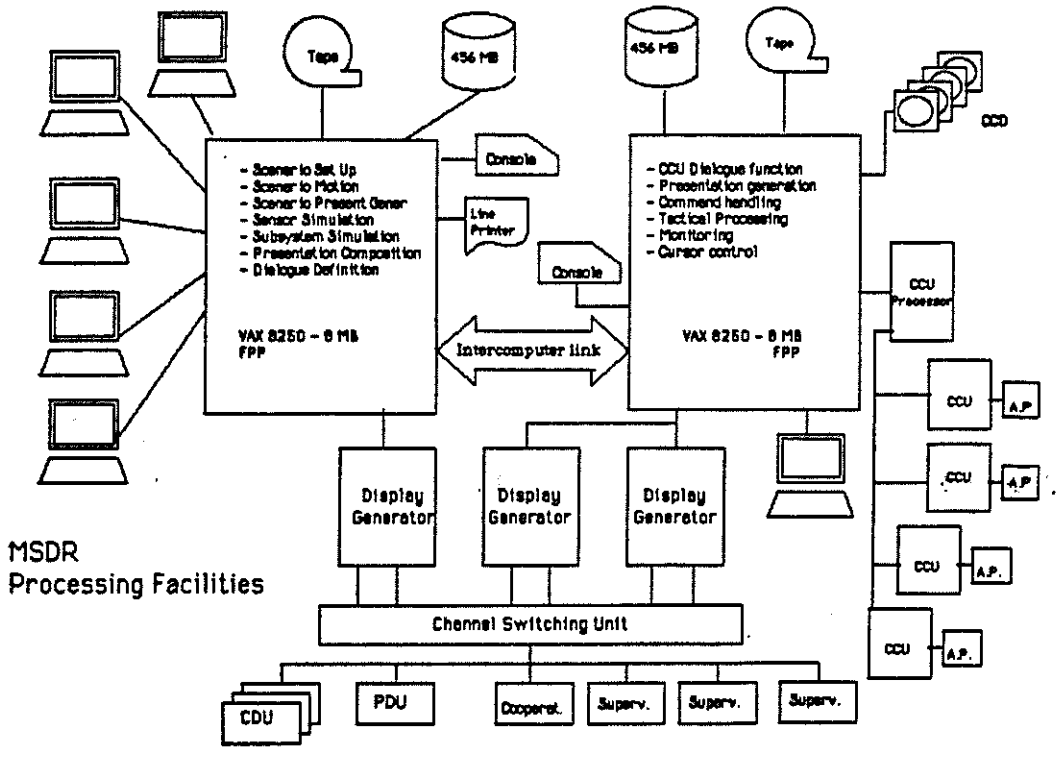


Figure 11



MSDR
Processing Facilities

Figure 12