# GENERIC COMPONENTS BASED EXPERT SYSTEM SHELL FOR AVIONICS APPLICATIONS

R.Santhi seela* and S. Janarthanan**

**Abstract**

*This paper focuses on the construction of a Generic Components Based Expert System Shell for various avionics applications. Traditional Expert Systems have been constructed using a single monolithic software program for a specific application. In traditional/commercial Expert System, either the Knowledgebase is hard-coded into the program or the linking of the Expert System modules with the knowledge bases takes place during compilation time. These commercial Expert Systems can't be used for complex applications where multiple knowledge bases have to be handled at run time, because the coupling of any particular knowledge base needs either the alteration of program code and/or recompilation. Secondly all these commercial Expert Systems are working in interactive mode during the inference process. To overcome these problems, ADE (Aeronautical Development Establishment) is currently involved in the design and development of a complex competent shell i.e. A Generic Components Based Expert System Shell for Avionics applications. This shell works in automatic mode and the system engineer can easily configure these generic components into a domain-oriented solution for any complex application including applications where multiple knowledge bases are handled. In this paper, we are discussing the role and generic nature of this shell along with its components and how they are configured as a domain-oriented solution for different kinds of applications at ADE.*

## Introduction

Several generic categories of Expert System applications have emerged in the last few years. Some of the categories of Knowledge Based systems relevant to avionics are fault diagnosis, avionics equipments design, pilot associate system etc. To meet these requirements, ADE is currently involved in the design and development of a matured system i.e. a Generic Components Based Expert System Shell For Avionics Applications. The design approach followed is based on decomposing the monolithic Expert System program into generic primary elements (components) independent of any specific applications and builds an Expert System Shell comprising of these generic components. The chief distinguishing feature of this shell is that the knowledge base is not hard coded and these components can be coupled with any number of domain specific knowledge bases at run time using dynamic memory allocation and de-allocation methodology. Special software component called Facts Interpreter was developed and also the knowledge base representation techniques were modified which makes this component based Expert System shell to work in automatic mode during the inference process. ADE has been engaged in the design, development and testing of avionics equipments for different applications. Fault diagnosis is an area, which demands the potential use of Expert System technology especially in avionics systems. By using this shell, we have built a Generic Open Architecture (independent of Test Facility and Test Equipment) based **Automatic Fault Diagnosis System (AFDS)** for the quad-redundant Digital Flight Control Computer **(DFCC)** of Light Combat Aircraft and Expert System based **Tactical Decision Generator (TDG)** system for the Air Combat Simulator in ADE. By almost any measure, AFDS is a successful application of components based Expert System technology, which automatically diagnoses the faults and gives fault reasoning in a various subsystems (92 subsystems) of a complex DFCC system. The AFDS provides channel wise, subsystem level diagnosis by handling multiple knowledge bases and reusing /adapting this AFDS system for the fault diagnosis of any other avionics equipments is easier. This paper discusses about the role of the components and how they are effectively configured for building completely different applications namely **"AFDS"** and **"TDG"** systems in ADE.

* Scientist ** Scientist (Retd)
Aeronautical Development Establishment, CV Raman Nagar, Bangalore-560 093, India
Manuscript received on 01 Jun 2004; Paper reviewed, revised and accepted on 14 Jul 2005

### Components-Based-Approach

A successful application of AI methodologies applies specific knowledge and inference to simulate human reasoning in solving different problems. ***Expert Systems*** are computer software programs that solve problems in well-bounded problem areas (domains) that would require the knowledge and reasoning skills of an expert. Traditional Expert System software has been constructed using simple monolithic software program. The design approach we have followed is based on constructing an Expert System Shell comprising of generic primary elements independent of specific applications or knowledge bases. This is accomplished by decomposing the monolithic software program into generic, independent primary elements (components) like ***Facts Interpreter, Knowledge Base Constructor, Inference Engine, and Expert Reasoner*** etc. These generic elements are re-engineered into a new type of software program based on the application. The main advantage of this approach is that adopting and/or reusing these generic primary elements in order to build a new software program for any specific application becomes easier. These generic components are developed using 'C' language because 'C' is the most popular programming language in use and 'C' compilers produce fast and efficient executable software routines. Secondly there are more system engineers who knows 'C' than those who knows one of the languages that are commonly used in Artificial Intelligence applications like ***LISP*** or ***PROLOG***. And also, when the Expert System routines need to be coupled with host programs coded in some other languages then building the generic components of the Expert System in 'C' becomes the most desirable solution. Before describing the systems built using this Expert System Shell the functionality of these generic components are discussed in detail in the following subsections.

### Facts Interpreter

Facts Interpreter component assigns the cell values for antecedents of the knowledgebase from the user input. Fail-Pass-Array is an array structure, which holds the cell value for attributes in the knowledge base. The cell value 1 indicates that the particular antecedent or consequent is ***True*** and 0 indicates ***False***.

### Knowledge Base Constructor

The ***Knowledge Base Constructor*** module reads the information (antecedents and consequents) from the ***Knowledge Base Information File during run time*** and arranges them in the form of a tree structure using link -list. Each branch of tree represents a rule and has a consequent at the end. The consequent of one rule can also be an antecedent of another rule. The antecedents are arranged at the nodes of the branches along with the ***Index Value***. The ***Index Value*** is an important parameter, which the system engineer can use for accessing the cell values in the Fail-Pass-Array. Each attribute has its cell value at the location indicated by its ***Index Value***. Transferring problem solving expertise from an expert to software is the critical problem in building Expert Systems. Knowledge representation requires translating collected knowledge into a collection of knowledge base information files readable by the software module called ***knowledge base constructor***. The information in the ***knowledge base information file*** is represented using ***production rules***.

### Inference Engine

The ***Inference Engine*** is a specialized routine, which finds out the valid conclusions within the knowledge base, based on the initial information given by the user. The ***Inference Engine*** uses ***Forward Chaining Mechanism***. The ***Forward Chaining Mechanism*** determines the conclusions that can be reached based on the current informations. The ***Inference Engine*** acts on the tree structured knowledge base and by using the interpreted facts, it derives the conclusion/conclusions (consequents) based on the cell values of Fail-Pass-Array. As the inference engine processes, the cell values of the Fail-Pass-array gets automatically updated. The system iterates till it finds no more new conclusions.

### Expert Reasoner

The Expert System should also provide an explanation facility that describes the system's reasoning to the user. Explanation and tracing a program's behavior is another area of research in AI. ***Expert Reasoner*** is an important software module, which acts simultaneously with an ***Inference Engine*** and keeps a record of reasoning process. At the end of the inferential episode, this module creates an output file containing information of its reasoning in order to support the explanation and justification to the user.

### Systems Built Using Generic Expert System Components

**AFDS** (Automatic Fault Diagnosis System) for DFCC of LCA and **TDG** (Tactical Decision Generator) for the Logical Pilot in an Air Combat Simulator are the two important on-going Expert System projects in ADE. DFCC is the Digital Flight Control Computer, which is the central processing and driving unit of the fly-by-wire flight control system of LCA. Logical Pilot is a computer driven aircraft in an Air Combat Simulator and the Air Combat Simulator is used for giving training to the Combat Pilots.

### Digital Flight Control Computer (DFCC)

The DFCC is a complex Quad-redundant (consists of 4 identical channels) Digital Flight Control Computer, which controls the total functionality of Flight Control System (FCS) of LCA. The LCA is a single pilot, single engine, supersonic, delta wing aircraft. LCA FCS is a full authority, quadruplex digital Fly-By-Wire Flight Control System. Each channel of DFCC includes five-printed wiring boards consisting of ***Digital Module, Analog Modules and Power Supply Module. The Digital Module*** contains CPU, Memories, Transputer, Watch dog Monitor, 1553B , RS 422, D/A and A/D, Serial Controller (SILC) and Cross Channel Data Link. The ***Analog Modules*** contain various driving circuits for five primary actuators, six secondary actuators and various interfaces to FCS related sensors. ***Power Supply Module*** consists of ***Power Supply*** monitoring circuits. DFCC contains 23 sub systems in each channel.

### Preliminary Study

The development stages were preceded by preliminary studies accessing the applicability of these generic components of the Expert System. In case of fault diagnosis, the fundamental driving force behind the consideration of Expert System technique for a particular application is a need/desire to replicate or automate the diagnosis process, which is performed well by human Experts. The success of an Expert System project depends heavily on a match between the targeted application area (including requirements and resources) and the applicability of these items to knowledge-based techniques. The success or failure of the Expert System will also be dictated by the amount of application knowledge that can be ascertained and appropriately harnessed in the system's knowledge base. The objective of the initial study is to evaluate the existing environment and tools available for development. Clearly stated requirements and operational concepts are just as important for Expert Systems as they are for Non- Expert System software. But ***Expert System*** projects frequently have requirements and operational concepts that are less detailed than the non-Expert System projects, especially in initial phases. But as it becomes better understood how experts do their jobs, the optimum role of an Expert System can be defined more accurately.

Another important factor that must be considered before the development of Expert System for fault diagnosis is the identification of realistic test cases both for evaluation and for the development itself. For engineering applications, this concept needs to be extended to incorporate the presentation of the test cases to the Expert System. In this application the Expert System needs to access the data (that will drive the analysis and reasoning) directly from the failed test cases and automatically diagnose the faults in each channel and for each subsystem without any human interruption. Development efforts undertaken without incorporating realistic interface concepts for data access in overall design may result in serious risk of requiring significant redesign. Application of this generic components-based Expert System shell is more suitable for building an automatic fault diagnosis system for a complicated system like DFCC. First of all for several different subsystems (92 sub systems) to be diagnosed, it is not feasible to think in terms of independently developed monolithic Expert System for each one. Rather one can think in terms of using this generic components based Expert System Shell since the coupling of the inference engine with the knowledge base takes place at run time, and a single shell is capable of handling fault diagnosis for all 92 subsystems successfully.

In case of Air Combat Simulator application, we mainly concentrated on the feasibility of interfacing the Expert System components with the existing host system (Logical Pilot). Since the Logical Pilot works in real-time, it is necessary to check, whether the Expert System components can produce the results with in the expected time limit, and also the compatibility between the Expert System modules and the Logical Pilot system.

**Automatic Fault Diagnosis System (AFDS) of DFCC**

*Architecture Overview*

The block diagram of the **Expert System Based Automatic Fault Diagnosis System of DFCC** architecture is shown in Fig.1.

This open architecture achieves the separation of diagnostic and testing functionality at design, storage and execution levels. The architecture is partitioned into three tiers of services. The **Testing Services** tier provides the facility for the development, execution and maintenance of test procedures/ test results. The **Diagnostic Services** tier contains the core modules of the architecture called the Expert System components for diagnosis/diagnostic reasoning. This service diagnoses the channel wise fault /multiple faults based on test results produced by the Testing services. **User Interface Services** is a powerful user front-end tool, which provides a sophisticated display and graphical representation of channel wise subsystems with the diagnosed faults and diagnostic reasoning information, as commanded by the user. As an aid for understanding the roles of Expert System for Automatic Faults Diagnosis System of DFCC, we first present a description of the existing Testing Services tier of DFCC.

*Testing Services*

The Expert System modules developed, functions along with the existing **Testing Services** tier. The Engineering Test Station (ETS) is the main test equipment of this tier, capable of simulating all sensors and actuators and also generate various inputs as required for testing the total functionality of DFCC in an integrated environment. The Laboratory Test Monitor (LTM) is a software program that executes on VAX, acts as a user interface module to the ETS and provides the facility for development and execution of Acceptance Test Procedures (ATP) to check the hardware functionality of DFCC.

The Acceptance Test Procedure contains a separate Test Procedure file for each subsystem and each Test Procedure contains a number of Test Cases. ***Each test case is assigned with a unique step number***. There are totally 74 main test procedures containing more than 13000 test cases to check the total functionality of DFCC. Once the testing is initiated, the test procedure of different subsystems are automatically executed in sequential order and the corresponding test report files containing '*' symbol for the failed test cases are created. These test report files will be analyzed in detail, by a team of experienced experts to diagnose the faults in different subsystems of DFCC.

The next two tiers of services replace the work of these experts using the Expert System technology. The preprocessor software called EXTRACT runs on VAX and filters out the step numbers of the failed test cases channel wise and generates channel wise failure information data files for each subsystem. The failure information data file contains information about the channel ID and the failed step

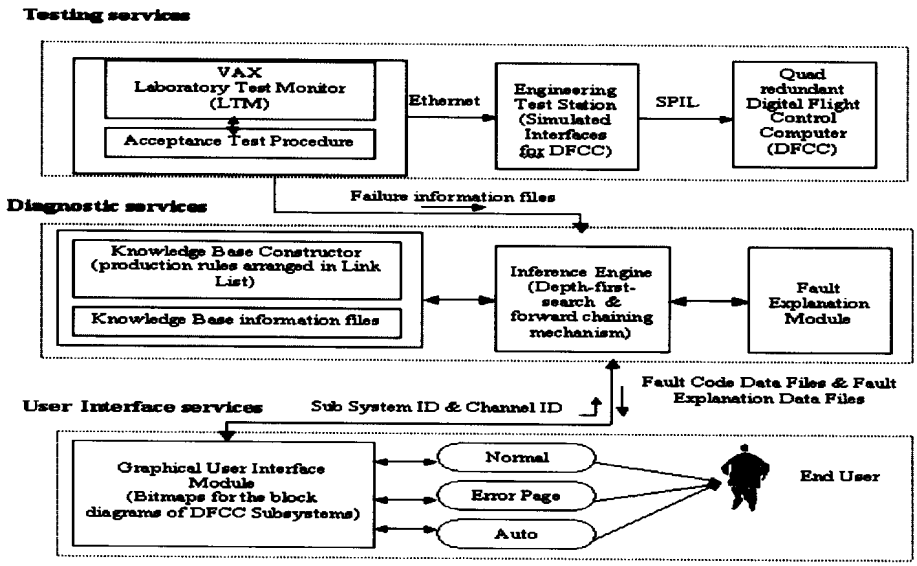


*Fig. 1 Architecture and interconnections of automatic fault diagnosis expert system modules with the existing testing services facility*

numbers. Each subsystem of all channels has its unique failure information file. These failure information files are passed on to the next tiers of services.

### *Diagnostic Services*

The **Diagnostic Services** and the **User Interface Services** tiers form the core of this Automatic Fault Diagnosis System Architecture. The **Diagnostic Services** tier contains the Expert System components like ***Facts Interpreter, Knowledge Base Constructor, Inference Engine, Expert Reasoner etc*** and also number of ***knowledge base Information Data Files***. Automating the fault detection and fault reasoning in a complex system like DFCC presented a significant challenge. Capturing of the expertise knowledge and formally representing them in the form of ***knowledge base Information File*** readable by the ***Knowledge Base Constructor*** module is a critical process in any Automatic Diagnosis System.

### *Knowledgebase Information Files*

The overall goal of this knowledge engineering process is to analyze, identify and define the knowledge structures that encompass the whole DFCC fault diagnosis domain. The definition phase determines the scope and the complexity of the task at hand by identifying the methodology to represent the knowledge base in the internal format. The primary source of this knowledge came from the experts who currently perform the fault isolation task in DFCC hardware. Knowledge representation requires acquisition of knowledge from the expert and encoding it into knowledge base information files readable by the ***Knowledge Base Constructor*** module. The design of the diagnostic strategy determines how the diagnostic rules in the ***knowledge base Information Files*** would be represented in software form.

The Knowledge Structure defines the internal format of all antecedents and consequents of the diagnostic rules. Each subsystem has its own knowledge base information file, and the number of rules in each knowledge base information file varies from 100 to 400. Each rule has antecedents in the range of 1 to 7. There is no restriction in the ordering of the rules in the knowledge base information file because of the iterative nature of the Inference Engine. Every antecedent and consequent is attached with a unique ***Index Value***. The Index ***Value*** is used to relate the antecedents with the corresponding test case. The Fail-Pass-Array is an integer array created during run time containing cell values, at the locations indicated by ***Index Values*** of the antecedents and consequents of the loaded knowledge base (Fig.2a).

The ***Inference Engine*** uses the ***Index Value*** to access/or update the cell value in the Fail-Pass-Array. In this application, the ***Index Value*** is used for interpreting the facts of the knowledge base from the failed test cases. Care is taken in assigning a unique ***Index Value*** to each antecedent in such a way that when the particular test fails the ***Index Value*** of the related antecedent can be derived from the corresponding step number of the failed test case and the cell value is set as 1 in the ***Index Value*** location of Fail-Pass-Array. It should be noted that the actual value of the step number of test cases couldn't be used directly as an ***Index Value*** of the corresponding antecedent because the actual value of the step numbers of the entire subsystems of the DFCC is in the order of several thousands. But the total number of step numbers pertaining to each subsystem is in the range of hundreds. The number of locations in the Fail-Pass-Array corresponding to the antecedents of the failed step numbers is also in the order of hundreds. In order to interpret the failed step numbers (Test Cases) into their corresponding facts and to accommodate these facts in the corresponding locations (***Index Values***) of Fail-Pass-Array, we need to map the locations of the Fail-Pass-Array to that of the failed step number. This is done by using a ***Index Determining Constant (IDC)***.

### *The Role of Expert System Components in AFDS System*

***KnowledgeBase Constructor*** reads information from the KnowledgeBase Information File and arranges them in the form of tree structure using link-list. A portion of the Right Inboard Elevon subsystem knowledge base is shown in Fig.2b. The ***Facts Interpreter*** component reads

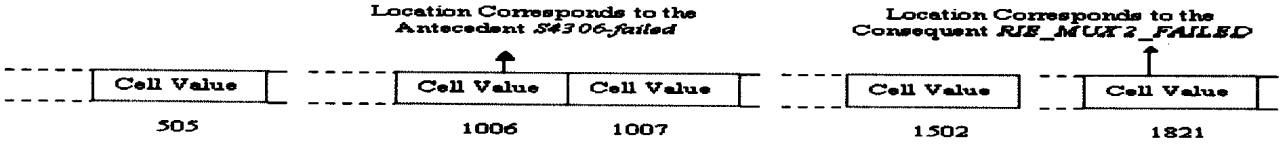


*Fig. 2a Fail-pass-array*

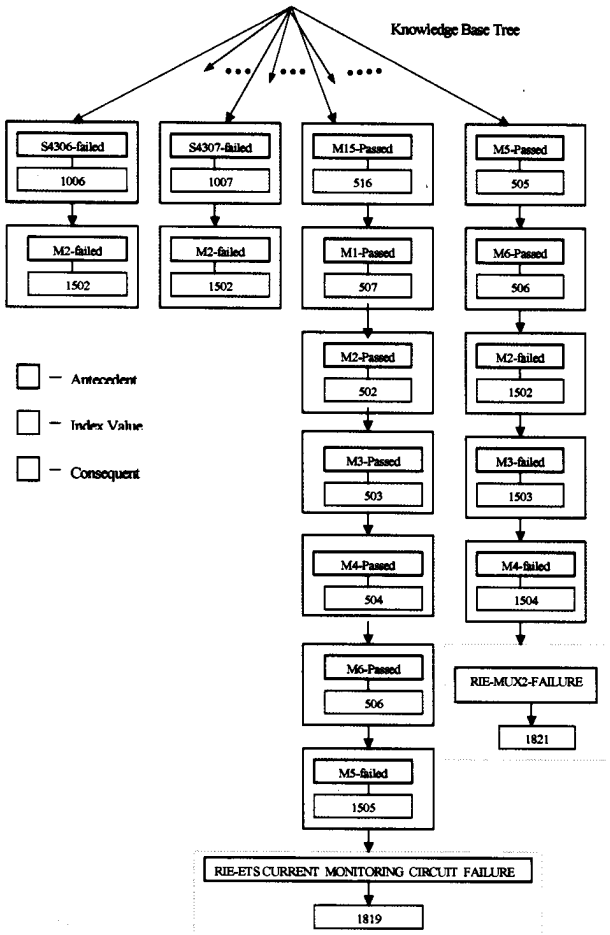


*Fig. 2b Portion of right inboard elevon subsystem knowledge base tree structure built by knowledge base constructor module*

the specific failure information file (containing failed step numbers and channel ID) and interprets the failed steps numbers in to their corresponding facts. From the failed step number the index of the corresponding antecedent is derived by subtracting the IDC value (e.g. for Right Inboard Elevon subsystem, the IDC value is 3300) and sets the corresponding cell value as 1 in the Fail-Pass-Array. Except for the failed test cases antecedent's cell values the rest of the cell values are set as 0. The initiation of diagnostic episode starts from here . Each subsystem has one or more IDC values. Assigning the value for IDC of each subsystem and ***Index Values*** for the antecedents of all the nodes of the knowledge bases involves careful calculation. The subsystem, which has multi range of step numbers, has multiple IDC value. The ***Inference Engine*** acts on the tree structured knowledge base, and derives the conclusion /conclusions (consequents) based on the cell values of Fail-Pass-Array. At the end of the diagnostic

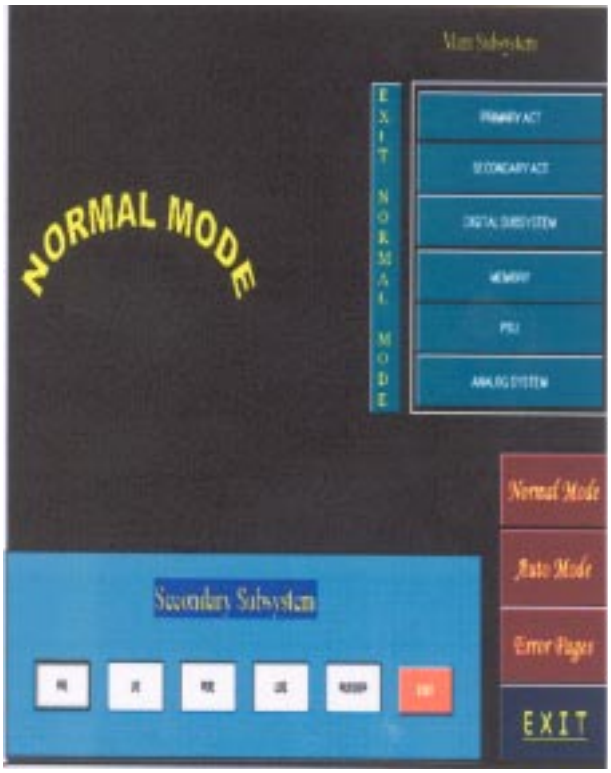


*Fig. 3 Normal mode of AFDS system*

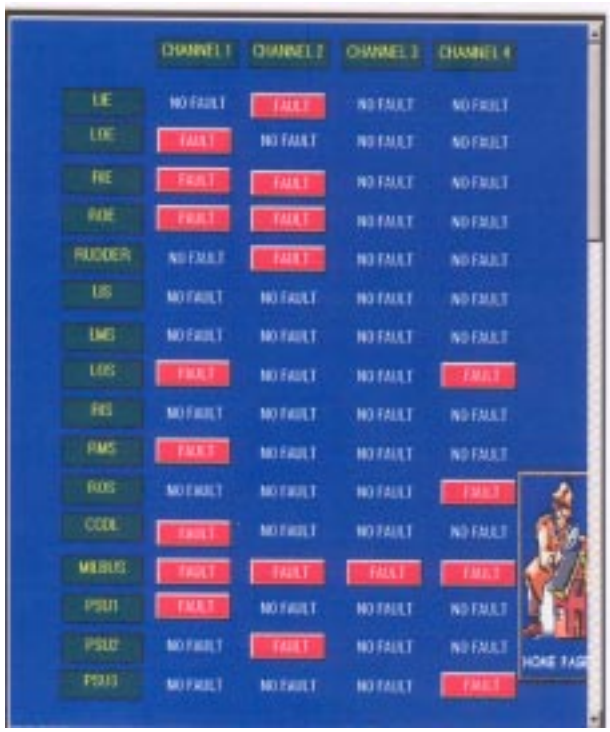


*Fig. 4 Error page mode of AFDS system*

episode, it creates two data files called ***Block.dat*** and ***Mon.dat***. The ***Block.dat*** file contains the information about the inferred functional blocks/devices that have failed in the particular subsystem and Mon.dat file contains the monitoring lines, which have failed in the system. These data files are passed on to the next tier of services called **User Interface Services**. At the end of the diagnostic episode, ***Expert Reasoner*** Component creates an output file containing information of its reasoning of the diagnostic process and passed on to the next tier of services.

### *User Interface Services*

The **User Interface Services** consist of user interface modules. The important software component in this tier is the ***Graphics Constructor*** program. This software component is specific to this application and was developed using VC++ platform. This module maintains the database required to construct a graphical block diagram of each sub system of DFCC. ***Graphics Constructor*** acts like a powerful user front end to the entire system. This software provides three modes of operation namely, i) **Normal** ii) **Error Page** and iii) **Auto**. The **Normal Mode** of User Interface Services tier allows the user to select any subsystem of his choice out of 92 subsystems.

When the user selects a particular subsystem e.g. 1553B subsystem, it displays the block diagram of that subsystem along with Channel ID buttons. Whenever the particular sub system is selected the graphics constructor module passes the corresponding Channel ID, and Subsystem ID to the **Diagnostic Services**. Based on these ID values the **Diagnostic Services** modules load the knowledge base from the corresponding ***Knowledge Base Information File*** into the system and based on the channel specific failure information file developed by the **Testing Services**, diagnose the faults and passes the diagnosed result files to the **User Interface Services**. By using these diagnosed result files, the ***Graphics Constructor*** displays the block diagram with the diagnostic information (Fig.5).

When the user commands for justification, it displays the explanation or the diagnostic process made by the Diagnostic Services during the diagnosis of that particular fault/faults (Fig. 5). The error page is a very sophisticated mode, which depicts the entire status of 92 subsystem of DFCC (Fig. 4).The user can see the status of any specific

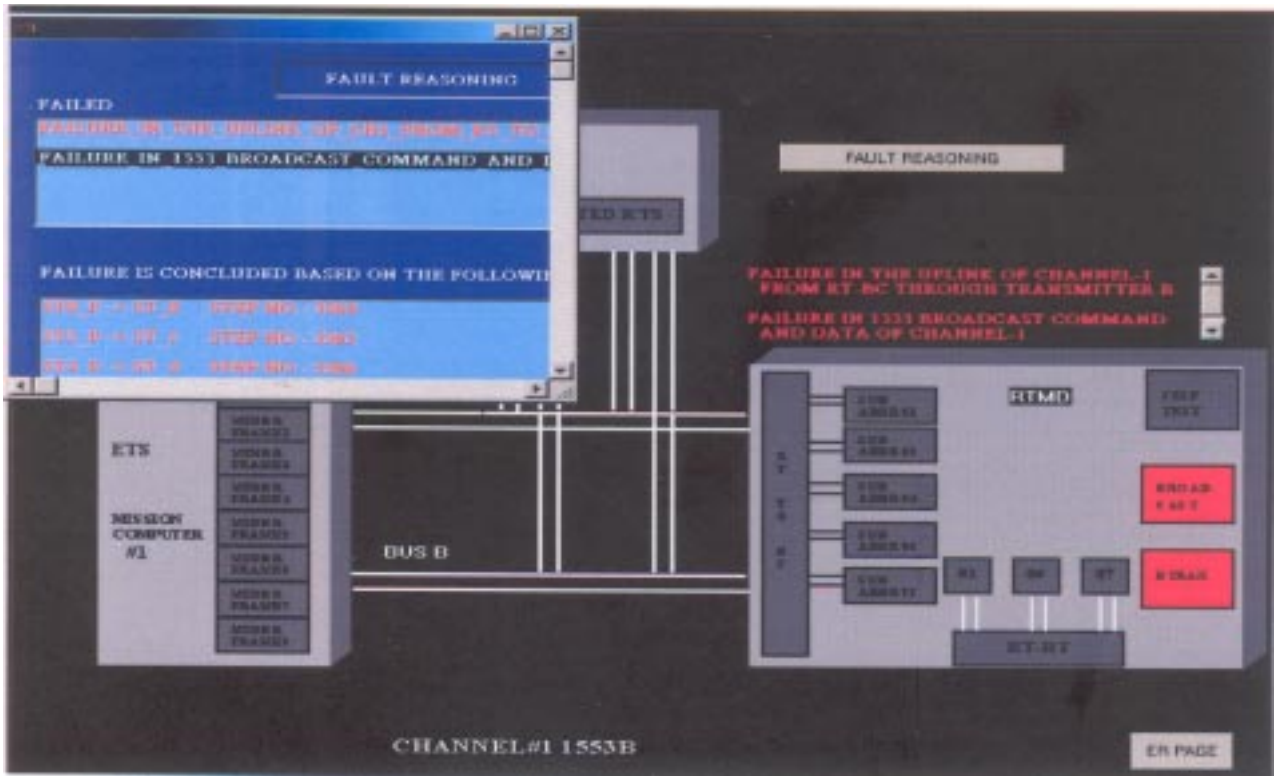


*Fig. 5 Block diagram of 1553B channel-1 subsystem with the diagnosed faults and fault reasoning details*

system by selecting it on the **Error Page** mode. In **Auto** mode, all the faulty sub systems of DFCC are connected through association network. The ***Graphics Constructor*** automatically displays the status of faulty subsystems one by one in sequential order. The user can traverse forward and backward in the network using the Meta keys (NEXT and PREVIOUS).

### Tactical Decision Generator System for Air Combat Simulator

As an aid to understand the TDG system, we first explain the existing Air Combat Simulator facility in ADE.

#### *System Overview*

Tactical Maneuver Simulator (TMS), Fig. 6, which simulates the worldviews in a 40 feet diameter dome provides an Air Combat Simulation environment for the Combat Pilots. This facility is intended for the real time simulation of air combat engagements between the human piloted aircraft (attacker) and the computer driven (logical pilot) aircraft.

In this Logical Pilot system, the parameters which the pilot decides using his intelligence/experiences during combat in the battlefield are determined by the Expert System based TDG system using knowledge bases. These parameters are Throttle Value (which determines the speed of the aircraft), Mode Of Operation (which is decided by the complex decision making process of the human pilot), and the Decision Interval Time (the time between the starting point of the two elemental maneuver). The logical pilot system has separate modules for aircraft simulation and for (Expert system based) decision making components .The simulation module does the simulation of the aircraft and the execution of the selected maneuver at each interval. The decision logic components contains well framed rule based knowledge bases and a systematic control strategy to determine the above parameters.


*Fig. 6 Tactical decision generator system for the logical pilot*

The block diagram of the expert system based logical pilot is given in Fig. 7. At each decision point (The time at which the Logical Pilot decides the next maneuver), the simulation module passes the information on relative geometry between the aircraft and the opponent and other necessary parameters required for the decision logic components. The system has two knowledge bases, one for determining Throttle Value and the other for determining Mode of Operation and Mode Dependent Decision Interval Value. Heuristic function is implemented in the search process of the Inference Engine. The rule whichever is having more number of already passed attributes will be selected for the next process. The TDG system handles Mode Selection knowledge base and Throttle Value knowledge base sequentially and passes the concluded parameters to the simulation module. The system has distinguished knowledge base information files for Throttle Setting and for Mode Selection and mode dependent Decision Internal Value.

#### *Mode Selection KnowledgeBase*

Five modes of operations shown in Table-1 have been incorporated in TDG. The mode selection knowledge

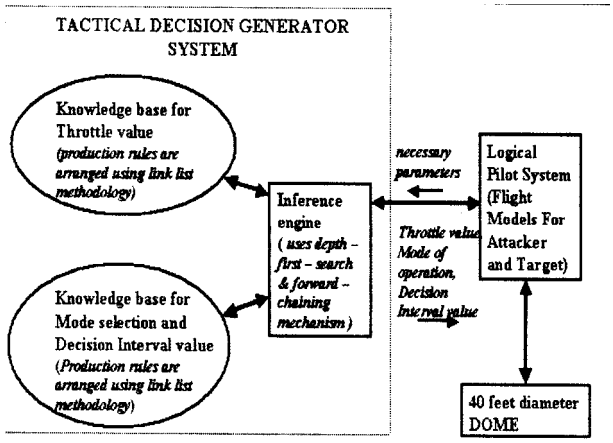


*Fig. 7 Expert system based logical pilot*

| Table-1 : Mode of operation | |
|---|---|
| **Mode** | **Decision Interval** |
| Aggressive | 0.25 sec |
| Defensive | 0.5 sec |
| Evasive | 0.25 sec |
| Disengage | 0.5 sec |
| Neural | 1.0 sec |

base is executed at the start of each decision interval before the maneuver selection. The mode selection knowledge base consists of complex rule base to determine the system's current mode of operation. This knowledge source is used to model a pilot's situational awareness and his changing problem solving strategies. Just as a pilot will recognize the difference between the aggressive and evasive situations and react accordingly, the situation assessment knowledge base provides information allowing the TDG to adapt its problem solving strategy based on the aircraft's mission, the current state of the system, the relative geometry between the aircraft and its opponent and the opponent's instantaneous intent.

### *Throttle Controller KnowledgeBase*

A rule based active throttle controller was developed to adjust the throttle setting based on the current mode of operation. The TDG module is called at the start of each decision Interval and can set the throttle to any position between idle and full afterburner. (0.0 = flight idle, 1.0 = military power, 2.0 = full after burner). The current mode of operation and the relative geometry information select target acquisition mode, or fine tracking mode or target / missile avoidance mode. Each mode has a set of throttle control rules that are used to maximize system performance in that mode.

### *The Role of Expert System Components in TDG System*

The generic components of the Expert System shell make decision logic components in Tactical Decision Generator system. The ***Facts Interpreter*** component interprets the parameter values sent by simulation module into their corresponding facts and updates Fail-Pass-Array accordingly. The ***Knowledge Base Constructor*** component reads the knowledge base information file and constructs a tree structured knowledge base along with the Index Values.

The ***Inference Engine*** acts on the tree structured Knowledge and derive the conclusions. The concluded parameter values are passed on to the simulation module. The ***Expert Reasoner*** prepares a inferential history for each pass. The Expert Reasoning information is useful for the evaluation of the accuracy of the knowledge bases in off-line Mode.

## Conclusion

We described in detail, about the components-based approach to build the expert system related applications. The Generic Components Based Expert System Shell is system independent, and also knowledge base independent, which provides the base for a open system. As mentioned earlier, these components can be coupled with any domain-specific knowledge bases and the system engineer can effectively configure these components to build a domain-oriented solution for any specific application. We also explained briefly how these components were configured for completely different applications (AFDS and TDG systems). The open architecture described for AFDS is independent of Test Facility and Test Equipment, and the same configuration can be used for diagnosing any other avionics systems with some minor modifications. Apart from these applications, the usage of our generic components based Expert System Shell continues to enhance and extend in all directions of its user community more successfully and effectively.

## Acknowledgement

Authors would like to take this opportunity to thank Sri M.D. Aravamudhan, Director, Associate Directors Sri Suresh Kumar, Sri G. Elangovan and CST Division Head Sri K.G. Ramamonohar for their for their inspiration and support during this development activity. The authors would also like to thank DFCC hardware team and ACS team who played an important role in the success of development of Expert System related projects by providing their domain expert knowledge, and also by evaluating /integrating the system. Supports received from FTP team are gratefully acknowledged.

## Reference

1. Santhi seela, R. and Janarthanan, S., "An Expert System for Automatic Fault Diagnosis of a Quadruplex Digital Computer", International Conference on "Advances in Aerospace Science", PS-294-301, 2003.

2. John W. McManus. and Kenneth H.Goodrich "Application of Artificial Intelligence Programming Techniques to Tactical Guidance for Fighter Aircraft", AIAA Guidance, Navigation and Control Conference, 1989.

3. Thomas J. Laffey., Preston A. Cox., James. L. Schmidt., Simon M. Kao. and Jackson Y. Read., "Real Time Knowledge Based Systems", A I Magazine, Spring, 1988.

4. Dutta. S., "Strategies for Implementing Knowledge Based Systems", 20132 IEEE Trans. Engineering Management, 44:79-90, February 1997.

5. Donald.A.Waterman., "A Guide to Expert Systems", MA, Addison-Wesley Publishers Co.

6. Jay Liebowitz., "Expert System Application to Telecommunication"., A Wiley - Interscince Publication.