N89 - 15610

Considerations in Development of Expert Systems for Real-Time Space Applications

S. Murugesan
NASA Ames Research Center
Mail Stop: 244-4
Moffett Field, CA 94035

ABSTRACT

Over the years, demand on space systems has increased tremendously and this trend will continue for the near future. Enhanced capabilities of space systems, however, can only be met with increased complexity and sophistication of onboard and ground systems. Artificial intelligence and expert system techniques have great potential in space applications.

Expert systems could facilitate autonomous decision making, improve in-orbit fault diagnosis and repair, enhance performance and reduce reliance on ground support. However, real-time expert systems, unlike conventional off-line consultative systems, have to satisfy certain special stringent requirements before they could be used for onboard space applications. Challenging and interesting new environments are faced while developing expert system space applications.

This paper discusses the special characteristics, requirements and typical life cycle issues for onboard expert systems. Further, it also describes considerations in design, development, and implementation which are particulary important to real-time expert systems for space applications.

1. INTRODUCTION

Current and future generation space systems are called upon to perform complex and more sophisticated, and intelligent tasks. The complexity of these systems are increasing in three dimensions: (1) the number of functions to be monitored and controlled, and also the kinds and volume of data to be considered, (2) need for quick response and faster rate of processing, and (3) need for more intelligent behavior.

There is a growing interest and pressing need for using knowledge-based problem solving techniques to cope with the increased demands on aerospace systems. Proper application of these techniques can provide better strategies for solving complex problems as discussed by Heer and Lum [14], and Weinweber [18]: i) autonomous satellite and space station control, ii) electric, propulsion, life support and thermal subsystems fault diagnosis, in-orbit repair/reconfiguration, and servicing, iii) intelligent vision and robotic systems with ability to recognize objects and scenes, and to find their way in places, whose conditions are not known, and far from earth.

Real-time expert systems are appropriate where there is an inherent need to enhance system autonomy without human assistance/intervention, where conventional techniques cannot make use of all relevant information providing intelligent or optimal solutions within a given time. Also, they could be used where humans work under severe psychological tensions, suffer from cognitive overload, fail to effectively monitor and evaluate all available information quickly, or make high-cost mistakes. Further, real-time expert systems are finding applications in domains such as routine operation and control, where qualified personnel who are able to evaluate complex situations and recommend actions are scare and not available all the time. A good discussion on the need for and desired features of expert system to assist human operators in monitoring and control of complex real-time process is given by Dvorak [8].

2. CHARACTERISTICS OF REAL-TIME SYSTEMS

The term real-time is often easier to recognize than to define. Though we understand the meaning of the term within the context of our own work, there is no consensus on a general or global definition. Also, there is a lot of misconception about what is meant by real-time. Some think that a system is considered real-time if it processes data quickly [26]. Another common usage is that real-time means "perpetually fast".

The feature that defines a real-time system is the system's ability to recognize an external event and to give a response by performing a service within a prescribed fixed time, which is dictated by application environment and criticality of the event. Response time - the time computer/expert system takes to recognize and respond to an external event -, is the most important factor in real-time applications. If events are not controlled within the allowed time, the process might go out of control and result in catastrophic effects. If given an arbitrary input or event, and an arbitrary state of the system, the system always produces a response by the time it is needed, then the system is said to be real-time [18]. The desired response time might vary from a few hundreds of microseconds to a few seconds, depending on application. Also, real-time expert systems have to perform their functions continually without ignoring on-going processes. Real-time systems are also known as ***interrupt driven*** and ***reactive*** systems. When an interrupt mechanism is used to signal a request for service/attention, the program/system becomes non-deterministic in that it is not possible to predict exactly what it will be doing a given number of clock cycles after initialization of the system.

In expert systems commonly used for medical diagnosis, design, configuration, financial analysis/advice, and other similar applications data is static and time to respond or give a decision is not critical. These off-line (sometimes called '*soft* real-time') advisory systems operate in non-dynamic domains at static points in time (i.e. data base, knowledge base, decision rules, etc. are fixed during a decision process). They do not have to keep up with the rapidly changing events in the external world, or meet high standards of '*hard*' real-time systems used for critical applications, such as very high reliability, availability and recovery after crashes.

The real-time domains have the following special characteristics, posing a set of complex and challenging problems for design and development of real-time expert systems [18].

- **Dynamic data (non-monotonocity):** Incoming sensor data, as well as facts that are deduced, do not remain static during the entire run of the program. They may decay in validity with time or they cease to be valid because external events have changed the state of the system.
- **Guaranteed response time:** The system must be able to respond by the time response is needed. Further more, one would like to achieve best response time within the deadline. Also, behavior of the expert system should be predictable that the response will fall within bounds or constraints.
- **Asynchronous inputs/interrupts:** Real-time systems must be capable of accepting asynchronous inputs and interrupts from external events. Also, they must be capable of interrupting an ongoing decision making process and resuming it after higher priority tasks are processed. It must have interfaces to gather data from a set of sensors or other (expert) systems. These requirements make testing and verification of expert systems more difficult.
- **Temporal reasoning:** Time is naturally an important variable in real-time domains. Typically, a real-time system needs the ability to reason about past, present and future (expected or anticipated) events, as well as the sequence in which the events had occurred. Therefore, knowledge representation schemes should permit representation of temporal relationships. A facility should exist for maintaining, accessing and statistically evaluating historical data.

- **Integration with procedural components:** Must be capable of integrating with conventional real-time software, which performs tasks such as data compression, signal or data processing, feature extraction and other application specific inputs/outputs.
- **Focus of attention mechanism:** When a significant event occurs, a real-time system should be able to reprioritize it goals and focus on important goals first. It could involve assigning context in which certain rules apply, modifying the set of sensors the system is currently looking at and changing the rate at which data is being analysed.
- **Continuous operation:** It must be capable of continuous operation over a long time - until stopped by an operator (through commands) or by other specific external events. Close attention, therefore, must be paid to garbage collection (efficient recycling of memory elements that are no longer needed) and archiving (maintenance) of sensor histories as far back as rules require them. Further, garbage collection must be done 'on the fly', not at processor's discretion.
- **Explanation facility:** It shows how the expert system reached a given conclusion and why the conclusion is justified. It gives a sequence of rules and facts that lead to a particular conclusion and describes its rationale for doing so. It is like an argument in favour of the conclusion. It is very essential in operator-assisted critical real-time systems, since the operator can accept or over-rule the decision reached by the system, after looking at the explanations by the system. Also, it can be used effectively for debugging and maintenance (extension) of knowledge base and to inspire confidence in systems performance and reasoning process [33].

In addition, constraints on realization of reliable and radiation-tolerant systems for space applications using microcomputers, and their processing speed and memory size limitations, is a critical bottleneck in applying knowledge-based techniques to real-time domains. Innovative methods have to be followed to overcome this bottleneck. The following characterstics are especially important to space applications.

- **Robustness:** It refers to gracefully degraded, reather than abrupt or fragile, behaviour of expert systems, while dealing with problems at the periphery of its domain (knowledge).
- **Handling uncertainty or missing data:** The system must be capable of handling reasonably and safely the uncertain, incomplete, vague, and missing information.
- **Reliability:** Extremely high reliability of operation in the targeted application environment and high degree of correctness and consistency of decisions are very crucial.
- **Fault tolerance:** Tolerance to failures of hardware, software (knowledge base, inference engine, operating system, etc.), and monitoring devices; fail-safe operations and graceful degradation
- **Ease of verification, validation and testing:** The system should be designed such that it is easily testable under various operational modes, and credible contingencies. Thorough verification and validation, and demonstration of proper functioning is very essential before actual use.

3. DESIGN AND DEVELOPMENT

Development of expert systems should be considered as a system engineering activity encompassing many tasks. It is a "team work". The division of expert system life cycle into various phases reduces the complexity of design by grouping and ordering main tasks of development [29]. It provides guidance on the order in which a project should carry out its major tasks. Many projects have come to grief, exceeded budget and schedule, and/or didn't deliver what was required, because they pursued their various development and evolution phases in a wrong way. Division of life cycle also helps to enforce an accepted development methodology among various persons involved in different phases and areas of development. Major phases in life cycles of expert systems development include:

- Problem identification/specification
- Acquisition of domain knowledge from experts, documents, previous case history, etc.
- Formulation of knowledge base, knowledge representation
- Choice (and/or development) of suitable inferencing/reasoning schemes and procedures
- Testing of expert system software (residing in development tools) under static (non-real-time) and real-time environments
  — Review human domain experts and specialists; revisions
- Integration of hardware deliverables and complied 'expert software'
- Testing under simulated and real-life environments under various modes of operation
  — Reviews by human domain experts and specialists; revisions
- Verification and validation: It covers the entire life cycle, and not just testing before delivery
- Delivery of flight-worthy Expert System; maintenance, upgrading and evolution

A typical life cycle of expert system is given in Figure 1. End product and outcome of each phase of development of expert system is summarized in Table 1.

3.1 Problem identification and domain feasibility study

Expert systems are useful for solving well-formulated problems, for which algorithmic solutions donot exists. These problems could be solved by using predetermined methods and heuristics that human experts have accumulated over years of learning and experience. An expert system consists of two basic elements [13]: a knowledge base, and an inference engine. A knowledge base consists of formalized facts and heuristic in a specific problem-solving domain. Inference engine uses this knowledge to solve problems.

The study of suitability of a domain for expert system application is very important and it involves the following steps [40]: 1) determine the nature of task, 2) determine if experts, who can solve the problem and are willing cooperative to share their expertise, exists and 3) determine whether their expertise can be modelled via an expert system. Domains suitable for expert system application tends to be deep and narrow. If the problems lend themselves to numeric or algorithmic solutions, it would be more effective to use those methods, rather than expert systems. However, if solution involves more of heuristics and human experts can solve the problem within a reasonable time and explain the solution process, expert systems could be considered. In essence one has to see whether the problem is "do-able" by one or more expert systems.

3.2 Requirements engineering

Expert system system development should begin with a complete, consistent and unambiguous idea of the needs of the user or the requirements of the system, and they should be well documented. The main advantages of the requirements analysis and understanding are: It serves as common ground for agreement between the developer and user/coustomer and helps in avoiding misunderstanding between them. It helps in early detection of errors. A survey [1] indicates that about thirty percent of errors in a major software intensive projects are due to faulty requirement specifications. It also helps the programmers to check that all the requirements are met. Further, it helps in defining requirements and specifications of the various real-time interfaces. In addition, it helps in generating good test cases and judging the quality of test cases actually used. Requirements of the system should be reviewed prior to the next phase of activity, by experts and project managers, system integrators, developers of other interfacing systems and must be agreed upon. The persons who are actually going to use the system during the operational phase should also be involved in this task.

3.3 Knowledge acquisition

The essence of an expert system is acquiring and encoding knowledge about a domain and then using it to solve problems in that domain. Knowledge elicitation has been cited as one of the bottlenecks in expert system development. Knowledge acquisition is the transfer of problem solving expertise from several sources, which include human experts, text books, literature, data bases, case histories and previous experiences. Of all these sources, the expertise of human specialists forms the main target of knowledge acquisition. Various techniques for acquiring knowledge from experts can be found in [7, 25, 27].

3.4 Knowledge representation

Knowledge representation refers to structuring of the acquired knowledge into computer recognizable form. Several knowledge representation models [13, 33] such as rules, semantic networks, and frames are being used, and each model has both advantages and disadvantages depending upon characteristics of the domain knowledge. Also, there are schemes which use a mixture of these representation schemes, to gain maximum benefits.

Various principles of software engineering [30, 31, 40] could be used in the design of expert system. They include: information hiding, separation of concerns, layering, and modularity. The principle of ***information hiding*** suggests that the group of rules hide internal details about the system. Also, while designing rules, it is better to separate different functionalities and use different rules to implement these functions. It is called the principle of ***separation concerns***.

The principle of layering suggests that system should be considered as composition of layers. Any layer is aware of only the layer beneath it. The activation of rules at the higher level due to activation of rules at the lower level should be minimal. Such practice not only simplifies implementation, but also simplifies testing.

3.5 Testing and validation

"A system can best be designed if testing is interlaced with designing instead of being used after the design". The purposes of testing and evaluation include: i) guaranting satisfactory performance of the system, ii) locating weaknesses in the system, so that further improvement can be done by making knowledge-base richer and problem solving strategies more powerful, iii) evaluating different functions, and iv) evaluating correctness of the results, response time, etc. Testing creates tangible degree of trust in system reliability. However, difficulties in defining the correct test strategy causes errors to go undetected. Functionality and design bugs not caught during development testing have been found after prolonged time when the system is in actual use.

There are two basic approaches to testing: *Block box* and *white box* methods. The block box approach is based on specifications and functional requirements analysis and input output characteristics [16]. The white box method is driven by the way the system (rules, inference strategies, etc.) are designed and implemented. The block box testing is performed by generating a test case (a set of test input) [11]. Test case is generally prepared manually based on specification of the system and when available from real-life data. The white box testing focuses on correctness of implementation, without much regard to overall system functionality. A combination of both the black box and white box techniques would be most effective.

By documenting test comprehensiveness goals in test plans one can lower the probability of missing tests (see IEEE Standard 829). Test comprehensiveness are defined in terms of four types of coverage: requirements, input domain, output range, and structure. Most probable errors in testing process are: 1) creating too few tests (they leave many bugs undetected in the delivered system), 2) creating wrong tests (they detect 'wrong bugs' rather than 'right bugs' that are critical and cause serious trouble), and 3) creating too many tests (doing unnecessary and redundant checks. Redundant testing can be avoided if one finds the test coverage each test provides). The IEEE standard 1008 for software unit testing gives some guidelines for testing.

Verification and validation (V&V) of expert system is one of the very important and difficult task of development of expert systems. Verification refer to confirming that the system has been developed correctly according to accepted methodology and system requirements, while validation means ensuring that system correctly serves the purpose for which it is intended. A V&V process is expected to catch user input errors, incorrect rules and facts, redundant rules, incorrect behavior of inference engine, and incorrect output after having reached a correct conclusion. 'V&V is not just a one day concern just before testing or delivery of a system; it spans the entire life cycle of expert system'. Currently , there is a lot of interest and concern in this area, and more information can be found in [3, 5,6 9, 10, 21-24, 28 36, 37].

4. MAINTENANCE OR SYSTEM EVOLUTION

Many expert system applications are characterized by lack of consistent and complete knowledge at the representation level, especially at the beginning of a project. Hence, it becomes necessary to modify existing knowledge base continually and maintain its consistency as new knowledge is imparted. Though this is a very important part of expert system development and operation [34], it is often gets least attention. Unlike in conventional software, rules and knowledge about the domain evolve with experience of their use, and hence, may have to be modified more often than algorithms. Belief support of rules might vary with feedback from earlier decisions. Further, validity of some rules, which might be time-dependent, can change over time, necessitating modification. Also, modification becomes necessary to correct errors found during various phases of development and during actual operation in the targeted environment.

Thus, as given by Ramamurthy [29], expert system maintenance could be *perfective* which encompasses changes asked by the user, *adaptive* which encompasses changes in environment, and *corrective* which corrects undiscovered errors and mistakes. In larger systems, about 65 percent of maintenance is perfective, 18 percent is adaptive, and 17 percent is corrective [20].

Mostly maintenance of expert system have to done by people not involved in the original development and hence they have to learn first about the system they are planing to maintain. This calls for better clarity, accuracy and completeness of different kinds of documents, besides skill and experience of people concerned. The more difficulty to understand the system, the more difficult it is to maintain, and hence, higher the maintainability risk. Many people prefer to call this phase as "system evolution" or "system enhancement" phase, rather than "maintenance".

Most expert systems, being large software projects, will suffer from what is known as *deadline effect*, limiting maintainability and reusability [29]. Most projects have completion deadlines. In the debugging and testing phase, with deadlines near, top priority of developers is to fix errors/bugs. The worst part is that many difficult bugs tend to get detected near deadlines. This forces to resort to "quick and dirty" fixing and hence systems loose their maintainability. Such practices have to be avoided.

5. SUMMARY

Though expert systems have found wide spread use in many applications, their use in critical real-time applications are very few. Development of real-time expert systems are much more difficult than the traditional consultative and advisory expert systems. Further, testing and validation of them still remains as a major problem. Expert systems cannot solve all types of problem. It is very important to understand the scope and limitations of current expert systems technology (both hardware and software) for critical aerospace applications, which pose many constraints.

Knowledge elicitation is a very important activity and a lot of attention has to be paid to it, as the knowledge is the key to success of an expert system. Also, experts must be cooperative, invest time, and must help in testing the system. Good design, documentation, adherence to accepted development methodology, enforcement of discipline in program design and modifications, and thorough testing and validation are very important for successful operation of expert systems in space.

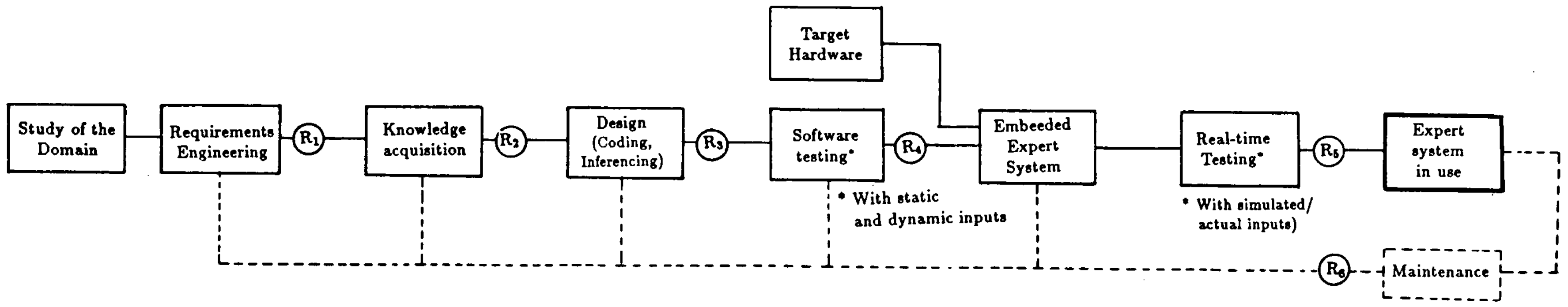
Acknowledgements

This work was done while the author held a National Research Council-NASA Ames Research Center Research fellowship. The author thanks Hamid Berenji and Terry Grant for their careful reviews and comments on draft of this article. He is also grateful to Dr. Henry Lum for his guidance and support.

References

1. Basili V.R, and Perricone, "Software errors and complexity: An emprical investigation", Comm. ACM, Jan. 1984
2. Berns G.M, "Assessing software maintainability", Communications of ACM, January 1984, pp 14-23
3. Boehm B.W, "Verifying and validating software requirements and design specifications, IEEE Software, Jan. 1984, pp 75-88
4. B.W. Boehm, "Spiral model of software development and enhancement," Computer, May 1988
5. Culbert C, et. al., "Approaches to verification of rule-based expert systems," SOAR'87, NASA-CP-2491, August 1987
6. Culbert C, et. al., "An expert system methodology which supports verification and validation," IEEE Conf. on AI Applications, 1987
7. Evanson S.E, "How to talk to to an expert," AI Expert, Feb. 1988, pp 36-41
8. Dvorak, "Expert systems for monitoring and control," Proc. Artificial Intelligence and Advance Technology Conf., Long Beach, Calif. April 1987.
9. Gaschnig J, et al., "Evaluation of expert systems: Issues and case studies," in D.A. Waterman, et al. (ed.) Building Expert Systems, Addison Wesley, 1983
10. Geissman J, "Verification and validation of Expert systems," AI Expert, Feb. 1988
11. Goodenough J.B, and Gerhart, "Toward a theory of test data selection," IEEE Trans. Software Engineering, Vol. SE-1, June 1975,
12. Green P. E., "Resource limitation issues in real-time intelligent systems," Proc. SPIE Conf. on Applications of artificial Intelligence, Vol. 635, Orlando, Fl, April 1986

13. Harmon P and King D, "Expert systems," John Wiley and Sons, New York, 1985
14. Heer L and Lum H, "Raising the AIQ of the space station," Aerospace America, Vol. , Jan. 1987, pp 16-17
15. Heny M.S.H, "Why evolutionary development," Future Generation Computing Systems, No.3, pp 103-109, 1987
16. Howden W.E, "The theory and practice of functional testing," IEEE Software Sep. 1985, pp 6-17
17. Irland E.A, "Assuring quality and reliability of complex electronic systems: Hardware and Software", Proc. IEEE, Vol. 76, No.1, Jan. 1988, pp 5-18
18. Laffey T.J, "Real-time knowledge-based systems", AI Magazine, Spring 1988, pp 27-45
19. Leinweber, D "Expert systems in space," IEEE Expert, Vol. 2, No.1, pp 26-36, 1987
20. Lientz B.P and Swanson E.B, "Software maintenance management", Reading, MA: Addision-Wesley, 1980
21. Miller E and Howden W.E, "Software testing and validation techniques," IEEE Computer Society Tutorial, 1981
22. Nguyen T, "Knowledge base verification," AI Magazine, Summer 1987
23. Nguyen T.A, "Verifying of consistency of production systems," Proc. Third conf. on AI Applications, Feb.1987, pp4-8
24. O'Keefe, et. al., "Validating expert system performance," IEEE EXPERT, Winter 1987
25. Olson J.R and Ruter H.H, "Extracting expertise from experts: Methods of knowledge acquisition," Expert systems, Aug. 1987, pp 152-168
26. O'Reiley, C.A, and Cromarty A.S, "Fast is not real-time: Designing effective real-time AI systems," Proc. SPIE 548, 1985, pp 249-257
27. Prerau D.S, "Knowledge acquisition in the development of large expert systems," AI Magazine, Vol. 8, Summer 1987, pp 43-51
28. Ramamurthy C.V, et. al, "Application of a methodology for development and validation of reliable process control software," IEEE Trans. Software Engineering, Vol. SE-7, No.7, Nov. 1981, pp 537-555
29. Ramamurthy C.V, et.al., " Programing in large," IEEE trans. Software Engineering, Vol. SE-12, No.7, July 1986, pp 769-783
30. Ramamurthy C.V, et. al., "Software development support for AI programs," Computer, Jan. 1987, pp 30 - 40
31. Reeker L.H, et.al., "Applying software engineering to knowledge engineering (and vice-versa), 27th Annual Technical ACM Symposium, Washington D.C., 1988
32. Shirely R.S, "Some lessons learned using expert systems for process control," IEEE Control systems Magazine, Dec. 1987, pp 11-15
33. Special issue on knowledge representation, Computer 1983
34. Soloway, E, et.al. "Assessing the maintainability of XCON: Coping with the problems of a very large rule-base," Proc. AAAI-1987, Seattle, WA, July 1987
35. Sorrells, "Time-constrained inference strategy for real-time expert systems," IEEE Proc. WESTEX 1985, pp 1336-1341
36. Stachowitz R.A, et. al. "Building validation tools for knowledge-based systems," First annual workshop on Space Operations Automation and Robotics, SOAR'87, NASA CP-2491, 1987
37. Suwa M, et al. "An approach to verifying completeness and consistency in a rule based expert system," AI Magasine, Fall 1982, pp16-21
38. Turner M, "Real-time experts", System International, Vol. 14, No.1, 1986 pp 55-57
39. Wright, M, et.al., "An expert system for real-time control," IEEE Software, March 1986, pp 16-24
40. Zualkernan, et.al., "Expert systems and software engineering: Ready for marriage?," IEEE Expert, Winter 1986, pp 25-31



Legend:
R_1: Requirements review
R_2: Review of domain knowledge/expertise
R_3: Design review
R_4: Software review, evaluation, validation
R_5: Test review, validation
R_6: Review of request for modification/enhancement

Fig.1 Life cycle of an expert system

Table I: Outcome of each phase of life cycle of an expert system

Phase of life cycle	**Outcome**
Study of suitability of the domain	Whether the problem can be tackled
Requirement Analysis/Engineering	Clear understanding of needs and requirements of targeted system, Interface specifications Documents: – *Functional Requirements Specifications* – *Test requirements - A preliminary study*
Requirements Review	
Knowledge Acquisition	Documented expertise (in natural language)
Knowledge verification & Review	Certified expertise
Coding Knowledge & Desiging problem solving strategies (Inference Engine)	Knowledge-based system design *Design document*
Design Review	
Testing in development environment under off-line simulated conditions	Removal of bugs/errors, inconsistencies *Test report, record of modifications*
Testing under real-time simulated environments	Checks and verifies time-dependent features - synchronization, response time, etc. *Test report, record of modifications*
Validation & Review	Certified Compiled knowledge
Transport to target hardware	Embedded expert system
Testing embedded system under simulated (static and dynamic) environments	
Testing under actual environment & Validation	Flight-worthy Expert System
Test and readiness Review	*Final documents with relevant revisions*
Flight operation (Maintainance and reuse)	Desired operations - Feedback for correction, further improvement - Review & authorization of modifications Document update **Evolving expert system in use**