International Journal of Reviews in Computing

IJRIC

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

SOFTWARE ERROR DETECTION AND CORRECTION (SEDC) USING LAYER BASED APPROACH IN EXPERT SYSTEM

M.S.JOSEPHINE, DR. K.SANKARA SUBRAMANIAN

Research Scholar, Mother Theresa University, Kodiakonal
Supervisor, Sri sai ram Eng.College, Chennai

Email : josejbr@yahoo.com

ABSTRACT

An expert system is a computer program that simulates the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field. Typically, such a system contains a knowledge base containing accumulated experience and a set of rules for applying the knowledge base to each particular situation that is described to the program. Sophisticated expert systems can be enhanced with additions to the knowledge base or to the set of rules. A System called SEDC has been proposed and implemented with an aim to find the errors that encounter during the software development phases in integrated development environment. The main goal of this is to create an application that enables the end-user to use these error solutions in their application program that serves as an error correction mechanism while developing project. The project also comes up with an user-interaction environment which prompts the experts to provide expert solution for a particular error based on their expertise. Thus expert system is initiated with machine learning process of IDE.

Keywords: *Expert system, Software Error Detection and Correction, IDE, Knowledge Capturing*

1.0 INTRODUCTION

Expert system in software development error detection and correction (SEDC) is a challenging task to detect and correct the application errors when the application is compiled. This research project is initiated to develop an expert system for a software developer based on their development exposure using SOA architecture. The developer can secure the knowledge from the development environment based on the error correction and utilizing the error correction mechanism to other development process. The experience gained can be shared with other developers in the same field using the knowledge base and the construction of expert system. This research paper mainly focuses on design of Expert System for Error Detection and Correction in Software Application Development Environment through capture of the error, representation of error in a structured database, collecting the possible solution for the occurred errors, extraction of the solution and the gained knowledge for further utilization in the same development environment using the layer based architecture

2.0 REVIEW OF LITERATURE

The expert system is which provide expert quality advice, diagnoses and recommendations given real world problems. expert systems are meant to solve real problems which normally would require a specialized human expert (such as a doctor or a mineralogist). Building an expert system therefore first involves extracting the relevant knowledge from the human expert. Such knowledge is often heuristic in nature, based on useful ``rules of thumb" rather than absolute certainties. Extracting it from the expert in a way that can be used by a computer is generally a difficult task, requiring its own expertise. A *knowledge engineer* has the job of extracting this knowledge and building the expert system *knowledge base*.

A first attempt at building an expert system is unlikely to be very successful. This is partly because the expert generally finds it very difficult to express exactly what knowledge and rules they use to solve a problem. Much of it is almost subconscious, or appears so obvious they don't even bother mentioning it. *Knowledge acquisition* for expert systems is a big area of research, with a wide variety of techniques developed. However, generally it is important to develop an initial

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

prototype based on information extracted by interviewing the expert, then iteratively refine it based on feedback both from the expert and from potential users of the expert system.

In order to do such iterative development from a prototype it is important that the expert system is written in a way that it can easily be inspected and modified. The system should be able to explain its reasoning (to expert, user and knowledge engineer) and answer questions about the solution process. Updating the system shouldn't involve rewriting a whole lot of code - just adding or deleting localized chunks of knowledge.

The most widely used knowledge representation scheme for expert systems is rules (sometimes in combination with frame systems). Typically, the rules won't have certain conclusions - there will just be some degree of certainty that the conclusion will hold if the conditions hold. Statistical techniques are used to determine these certainties. Rule-based systems, with or without certainties, are generally easily modifiable and make it easy to provide reasonably helpful traces of the system's reasoning. These traces can be used in providing explanations of what it is doing.

Expert systems have been used to solve a wide range of problems in domains such as medicine, mathematics, engineering, geology, computer science, business, law, defence and education. Within each domain, they have been used to solve problems of different types. Types of problem involve *diagnosis* (e.g., of a system fault, disease or student error); *design* (of a computer systems, hotel etc); and *interpretation* (of, for example, geological data). The appropriate problem solving technique tends to depend more on the problem type than on the domain. This research adopted the expert system for the Error detection and correction process for the Software development in Integrated Development Environment.

3.0 METHODOLOGY

The following procedure is adapted to design and implements the Error detection and structural data base construction for Software development environment

Study the Basic functional process of design and deployment of Expert system

Study the applications of Expert system and various design models and on going researches in the Expert system

Study the design procedure of Expert system using layer based approach

Study of Expert system Architecture

Adopt Expert system architecture for Software development environment to design Error detection and Correction

Define he Error detection procedure for Integrated Development Environment (Visual Studio)

Design Expert system Architecture for Software development environment to design Error detection and Correction

Capturing errors from different phases of software development in IDE with multi domain based project.

Classify the errors according to the predefined standard error numbers which is prescribed by IDE.

Validate identified errors with available solutions such as Help, Manual and web links.

Gathering the experts' solution and update the same inline with standard errors.

Provide hands on suggestion and recommendation from the knowledge base using rule based approach.

Analysis of Captured Error and classify the same

Determine the Knowledge structure and learning ration

Finings and interpretation

4.0 EXPERT SYSTEM

The expert system is an application program that emulates the human expert decision-making capability. It provides a solution to a problem solving using knowledge base which requires the human expertise. An Expert system is a computer program that simulates the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field. Typically, such a system contains a knowledge base containing accumulated experience and a set of rules for applying the knowledge base to each particular situation that is described to the program. Sophisticated expert systems can be enhanced with additions to the knowledge base or to the set of rules.In another way, a software based system which makes or evaluates decisions based on rules established within the software. Typically used for fault diagnosis.

An expert system is a computer program that provides expert-level solutions to 'important problems and is:

heuristic -- i.e., it reasons with judgmental knowledge as well as with formal knowledge of established theories;

transparent -- i.e., it provides explanations of its line of reasoning and answers to queries about its knowledge;

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

Flexible -- i.e., it integrates new knowledge incrementally into its existing store of knowledge.‘.
The key ideas have been developed within Artificial Intelligence (AI) over the last few years, but in the last few years more and more applications of these ideas have been made. The purpose of this is to familiarize users with the architecture and construction of one important class of expert systems, called rule *based systems.*
Computers are general symbol manipulating devices, the non-numeric and heuristic aspects of problem solving can be encoded in computer programs as well as the mathematical and algorithmic aspects. Artificial Intelligence research has focused on just this point. Work on expert systems is, in one sense, the applied side of AI, in which current techniques arc applied to problems to provide expert-level help on those problems. However, there is more to building an expert system than straightforward application of AI techniques.

4.1 Expert system architecture

The following general points about expert systems and their architecture have been illustrated.
1. The sequence of steps taken to reach a conclusion is dynamically synthesized with each new case. It is not explicitly programmed when the system is built.
2. Expert systems can process multiple values for any problem parameter. This permits more than one line of reasoning to be pursued and the results of incomplete (not fully determined) reasoning to be presented.
3. Problem solving is accomplished by applying specific knowledge rather than specific technique.
This is a key idea in expert systems technology. It reflects the belief that human experts do not process their knowledge differently from others, but they do possess different knowledge. With this philosophy, when one finds that their expert system does not produce the desired results, work begins to expand the knowledge base, not to re-program the procedures.
There are various expert systems in which a rule base and an inference engine cooperate to simulate the reasoning process that a human expert pursues in analyzing a problem and arriving at a conclusion. In these systems, in order to simulate the human reasoning process, a vast amount of knowledge needed to be stored in the knowledge base. Generally, the knowledge base of such an expert system consisted of a relatively large number of "if then" type of statements that were interrelated in a manner that, in theory at least, resembled the sequence of mental steps that were involved in the human reasoning process.
Because of the need for large storage capacities and related programs to store the rule base, most expert systems have, in the past, been run only on large information handling systems. Recently, the storage capacity of personal computers has increased to a point where it is becoming possible to consider running some types of simple expert systems on personal computers.
In some applications of expert systems, the nature of the application and the amount of stored information necessary to simulate the human reasoning process for that application is just too vast to store in the active memory of a computer. In other applications of expert systems, the nature of the application is such that not all of the information is always needed in the reasoning process. Based on this approach the layer based expert system design adopted for software development error. The process includes error detection, classification and recommendation for correction.

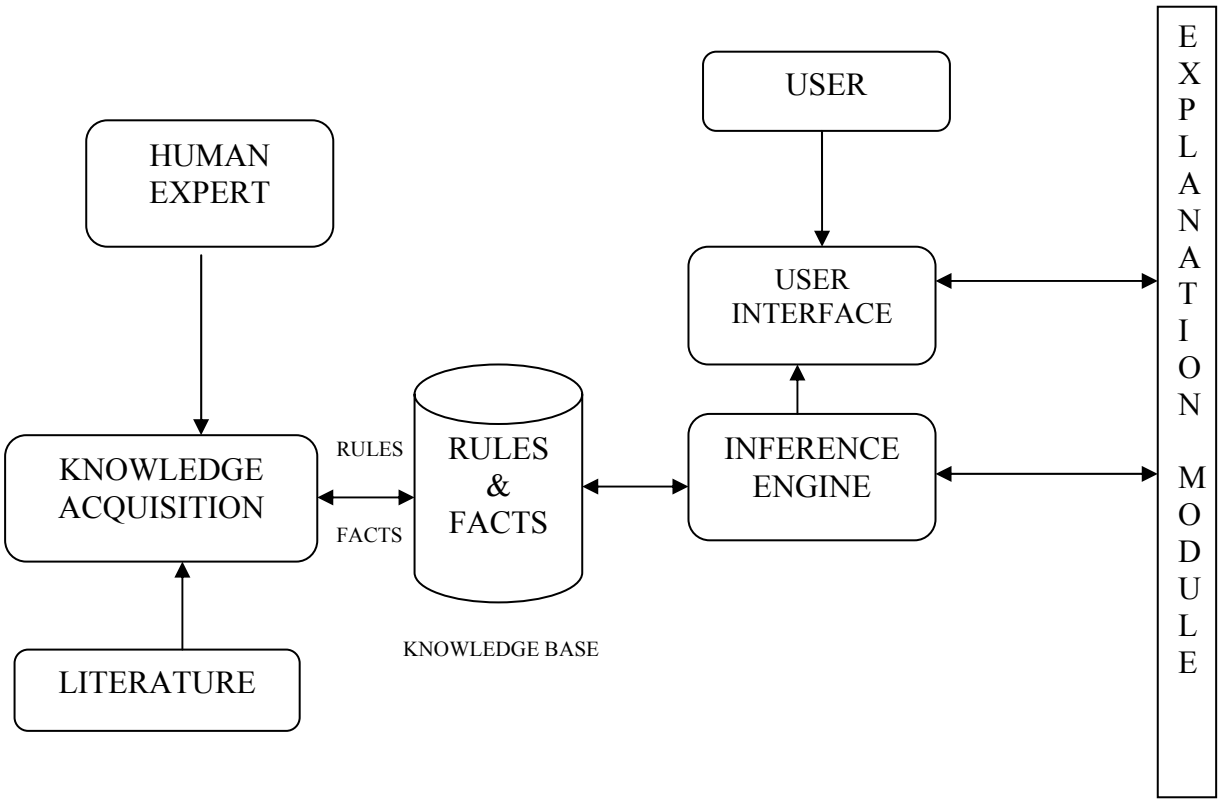
4.2 Expert System Architecture & Development

An expert system is a computer program that uses domain specific knowledge and inference techniques to simulate the problem solving behavior of a human expert of the same field. The system allows the modeling of information and knowledge at higher level of abstraction to implement the human logic of problem solving. Rule based programming is one of the commonly used techniques to develop expert system and the same has been used in the present work too. The rule based expert system store large body of facts called declarative knowledge and rules i.e. procedural knowledge to manipulate the facts. The system is capable in using its internal knowledge and rules to formulate its own solution procedure based on problem definition. The proposed system, like the other rule-based system, has the following four functional modules:

- Knowledge base (KB)
- Inference engine
- User`s interface and
- Explanation module

IJRIC

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336



The interconnection and arrangement of these modules is shown in Fig... The development process of the expert system is very systematic and can be carried out in different stages. The whole process followed has been presented in the flow chart diagram shown in the Fig...The KB of the present system consists of two modules. First module of the knowledge base for identification of errors and other is for recommending appropriate solutions.

Knowledge base (KB)

Knowledge acquisition is the base process to construct the knowledge base. In this phase of expert system development cycle domain specific knowledge is transferred to knowledge engineers. The acquisition knowledge of the proposed system obtained from human experts by gleaning the knowledge from standard references. It contains the description about possible errors during the code design development and testing. All possible classification in terms of development environmental error/symbols represented in a accessible format.

Inference engine

Inference process is extract the information based on the rule based system with symbols, which requires translation of domain specific knowledge in the standard symbolic forms. The facts and rules are been represented as per the adoptable and representation syntax and architecture of defined knowledge base. In general rule based approach with factorization adopted to represent the knowledge in the knowledge base. It is easily makes an inference with the symbol existed in the knowledge base and extracted based on rule based approach.

User`s interface

User interface model provides the way of interaction between the user and the expert system. Query based approach has been implemented to extract the error related information and its corresponding recommendations. The logical sequence of symbol has been arranged (classification of error with its number) such that the system capable in making right question at the right moment during the development environment.

Explanation module

In this proposed software development environmental error detection and correction provides explanation of its own for the standard errors which frequently occurs. It's determining

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

own decision for the recommendation by expert's suggestion. This is important for the validation and understanding of the solution and reasoning process of the expert system. This proposed system is aimed to provide an extendable facility of auto tracing of inference engine during the development of a specific environment. Most of the rule based expert systems provide an explanation that to be maintained and implemented in the proposed error detection and correction system.

4.3 Layer Based Architecture for Expert System

As per the layer based semantic web Expert system layer based approach, the following model is described. The Layer architecture built on RDF(s) knowledge layer a native inference engine with adopted and optimizes reasoning for specified domain. Layer architecture is composed of four main layers each being able to hole sub-layers and modules. The layer based system and its modules represented below:

This layer based approach contains the following layer

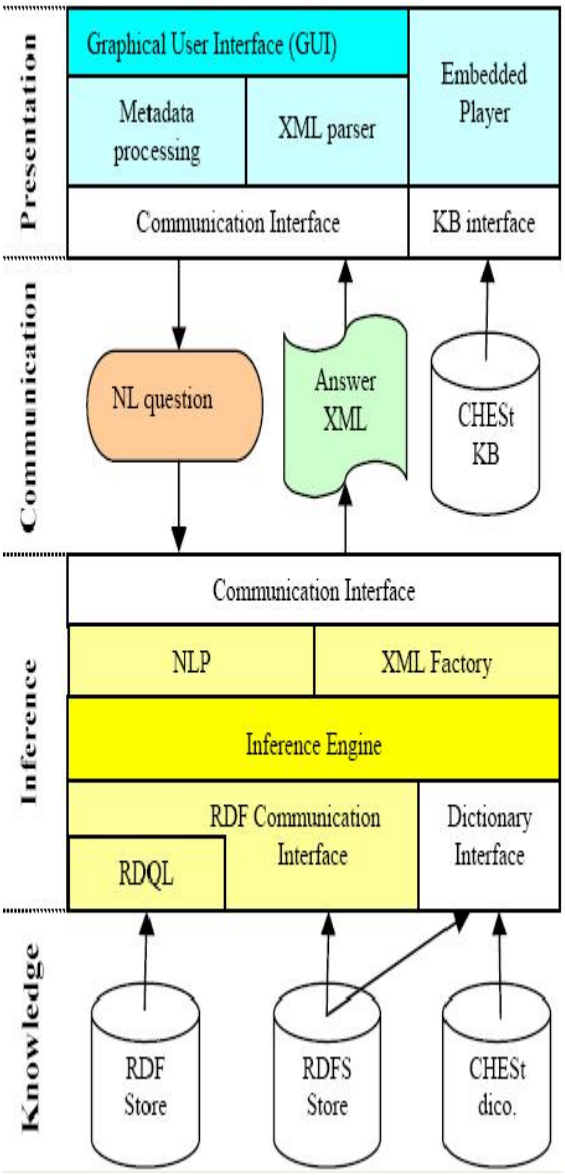
- Presentation Layer
- Communication Layer
- Interface Layer
- Knowledge Layer
- Knowledge Layer

The Knowledge Layer regroups all knowledge resources that are used for the reasoning processes. Knowledge Layer is the set of data sources which are accessed by the upper interface engine for reasoning about the knowledge. The raw data in the knowledge- base can be in a text format. Indeed, interpretation over the knowledge are only promising if its content is semantically annotated with sufficient Meta data. Therefore it seems evident to use a consistent semantic representation of the knowledge and to offer a uniform interface to the upper layer. The knowledge base layer has two sub-layers called raw data and semantic annotation.

Inference Layer

The Inference Layer is technically the most important, because it implements the retrieval system. It is composed of the natural language processing(NLP) module, the inference engine for reasoning over the knowledge base with respect to the user questions, and generating system's answer. In this layer, it receives the questions in NL from the upper layer translate it into a semantic query, launches it against the knowledge base and finally returns the pertinent text to the upper layer. It contains the ontology and classification.

A domain ontology O(L,H) jis composed of a domain language L and a hierarchy of concepts H. L is the set of all words over a certain alphabet, $L \subseteq \sum^*$ that are known by the system for the given domain ontology. The hierarchical classification of concepts $H=(V,E,v_0)$ is a rooted tree, with V the set of nodes representing the concepts, E the set of edges and v_0 the root node.



A multimedia clip d is classified under a concept v, if d is about v and there does not exist a more specific concept v' under which d could be classified.

The inference layer sub-module is used to extract the information from the knowledge-base. The

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

reasoning mechanism supports the appropriate result. As the user question is expressed in NL, and the knowledge-base description is in RDF(S), a direct comparison is not possible because both representations are not compatible. Therefore we must transform the user's question into a RDF query that is launched against the knowledge-base. This operation is done by the interpretation function I.

The interpretation I of a user question q in NL for a domain ontology O and an allocation function g is written as $I[q]^{O}_{g}$= R with R being the set of relevant documents.

The communication inference to the knowledge layer uses the RDF capabilities. The Inference Layer can be seen as a black-box. It receives a NL question, performs some reasoning about it, and returns an answer. It communicates via standard interface; one to the upper communication and presentation layers, and one to the lower knowledge layer. Both will be described below. But let us first recall that the inference engine must be platform independent, even portable, and accessible over a network by client applications.

Communication Layer

The Communication Layer specifies the exchange of information between the distributed Inference and Presentation Layers. It must ensure a transparent but error-free communication. The Communication layer allows a transparent communication between the client application(Presentation Layer) and the inference engine (Inference Layer). It should not be important if these two layers are on the same machine or not. Furthermore the communication must be error-free, simple and hardware independent. It seems that the best solution lies in socket communication.

Presentation Layer

The Presentation Layer implements the Human-Machine Interface, and thus needs an ergonomically adapted look. It allows the user to freely formulate a question in natural language(NL). The presentation Layer represents the interface between the user and the machine. It gets a question from the user and transmits it to the inference engine via the communication Layer. The presentation is represented in two ways such as Pedagogical Aspects and Technical Aspects.

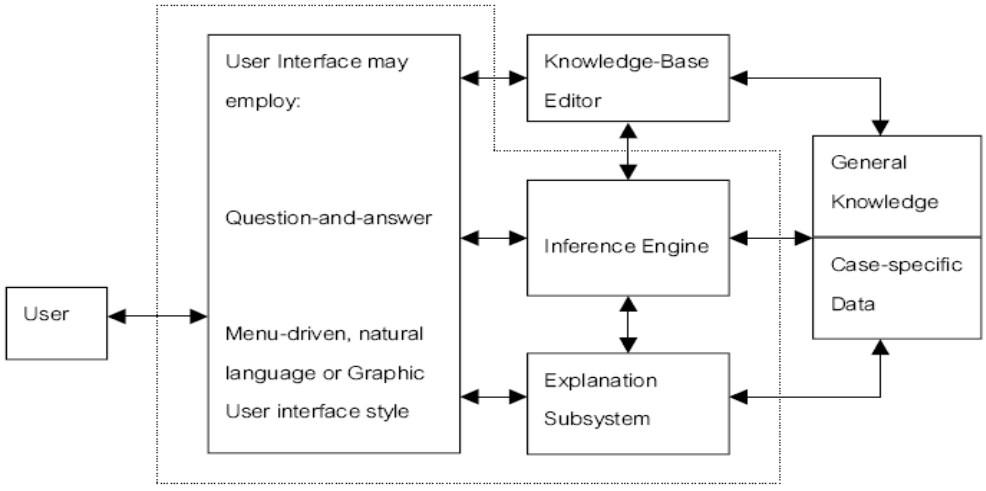
5.0 DESIGN -ADOPTED STRUCTURE OF EXPERT SYSTEM

5.1 Components of Expert system for SEDC

Knowledge base: which is a collection of the rules and facts from expert who has expertise in the domain problems.

Expert system shell is composed of user interface, explanation subsystem, inference engine and knowledge base editor.

User Interface:The user interacts with expert system through a user interface that makes access more convenience for the human. It has a variety of interface styles including question-and-answer,



Architecture of a typical expert system

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

menu-driven, natural language or graphics interface.

Explanation Subsystem:Explanation subsystem will show the path of reasoning to the user for tracking the output result. This will also make the system itself more reliable in providing the problem solution. The explanation subsystem allows the user to ask two questions which are "Why ?" and "How ?" In a "why" query, the user could ask "Why does the system ask the question?" In a "how" query, the user asks "How does the system get the result ?" The answers to the above two questions are the sequence of rules that are fired.

Inference Engine: In a rule-based system, two general methods of inferencing are commonly used as the problem solving strategies: Forward and backward chaining.

Forward chaining is reasoning from facts to the conclusions.

Backward chaining involves reasoning from a hypothesis to the facts that support the hypothesis.

Knowledge Base Editor:The editor is used for manipulating the knowledge base. (the command such as edit, delete and add).

5.2 Expert system for Error Detection and correction

The principal distinction between expert systems and traditional problem solving programs is the way in which the problem related expertise is coded. In traditional applications, problem expertise is encoded in both program and data structures.

In the expert system approach the entire problem related expertise is encoded in knowledge base only; none is in frontend applications. This organization structure has several benefits.

5.3 ERROR DETECTION AND ERROR CORRECTION

Definitions of Error detection and error correction

Error detection is the ability to detect the presence of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

Error correction is the additional ability to reconstruct the original, error-free data.

Two basic ways to design the channel code and protocol

There are two basic ways to design the channel code and protocol for an error correcting system:

Automatic repeat-request (ARQ): The transmitter sends the data and also an error detection code, which the receiver uses to check for errors, and request retransmission of erroneous data. In many cases, the request is implicit; the receiver sends an acknowledgement (ACK) of correctly received data, and the transmitter re-sends anything not acknowledged within a reasonable period of time.

Forward error correction (FEC): The transmitter encodes the data with an **error-correcting code** (ECC) and sends the coded message. The receiver never sends any messages back to the transmitter. The receiver decodes what it receives into the "most likely" data. The codes are designed so that it would take an "unreasonable" amount of noise to trick the receiver into misinterpreting the data.

It is possible to combine the two, so that minor errors are corrected without retransmission, and major errors are detected and a retransmission requested. The combination is called hybrid automatic repeat-request.

5.4 ERROR DETECTION SCHEMES

Level Error Classification Method

Errors, which are agreed as valid, will be categorised as follows :-

Category A - Serious errors that prevent System test of a particular function continuing or serious data type error

Category B - Serious or missing data related errors that will not prevent implementation.

Category C - Minor errors that do not prevent or hinder functionality.

5.5 Level Error Classification Method based its impact

Catastrophic: **Defects that could (or did) cause disastrous consequences for the system in question. E.g.) critical loss of data, critical loss of system availability, critical loss of security, critical loss of safety, etc.**

Severe: **Defects that could (or did) cause very serious consequences for the system in question. E.g.) A function is severely broken, cannot be used and there is no workaround.**

Major: Defects that could (or did) cause significant consequences for the system in question - A defect that needs to be fixed but there is a workaround.

Minor: Defects that could (or did) cause small or negligible consequences for the system in question. Easy to recover or workaround.

E.g.1) Error messages misleading.

E.g.2) Displaying output in a font or format other than what the customer desired.

No Effect: Trivial defects that can cause no negative consequences for the system in question. Such defects normally produce no erroneous outputs.

E.g.1) simple typos in documentation.

E.g.2) bad layout or mis-spelling on screen.

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

6.0 FINDINGS ON OCCURRED ERRORS

For this research five software development project adapted and observed its errors in all developmental phases. The errors are commonly occurs in Keywords, variable declaration and utilization, syntax of the statements, data base connectivity, function specification, DLL_OLE, object connectivity, MFC and file errors. The following table shows the occurrence of the error

Error Occurrence at	SDE_ JAVA	SDE_ ASP	SDE_ Script	SDE_ System	SDE_ Network
Keyword	544		139		50
Variables	31	240	45	52	344
Syntax	283	688	1420		312
Database connectivity		62	30		
Functions	151	402	20	545	227
DLL_OLE	51	182	154	22	48
Object Connectivity	274	899	204	482	295
MFC			42	10	69
File	104	298	125	593	310
	1438	2771	2179	1704	1655

Table: Error occurrence at

Graph for total errors occurred level such as keyword, variables, syntax, database connectivity, functions, DLL_OLE, object connectivity, MFC and File. Most of the error occurs in the syntax level. Next to that object connectivity followed by file processing. In data base connectivity and MFC leased level error occurs. The occurred error level and its number of occurrence shown in the following graph.

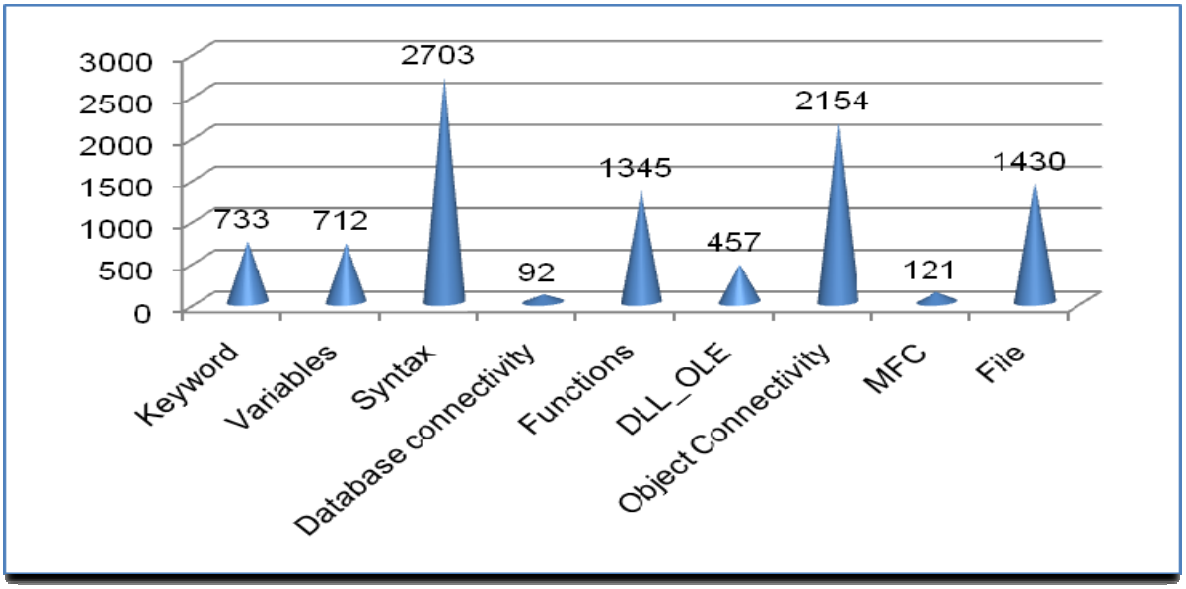


Figure: Error occurrence at

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

The errors are classified design, coding, build, run and implementation levels. Each phase level errors are observed in different type of project with its total life cycle. The error occurs in each project and its level of occurrence is listed below as a table format

Error Occurrence level	SDE_ JAVA	SDE_ ASP	SDE_ Script	SDE_ System	SDE_ Network	Total
Design	20	18		32	75	145
Coding		290				290
Built	236	579	1241	114	312	2482
Run	952	788	620	776	482	3618
Integration	18	71	87	144	289	609
Implementation	212	1025	231	638	497	2603

Table : Error Occurrence level

The most of the errors occurs in the run level next to the built process level. Most of the errors are occurs in the java based project during the runtime followed by the ASP project. But most of the design level errors are occurs in the script based project. In occurred level of errors in different phases represented as a diagram below

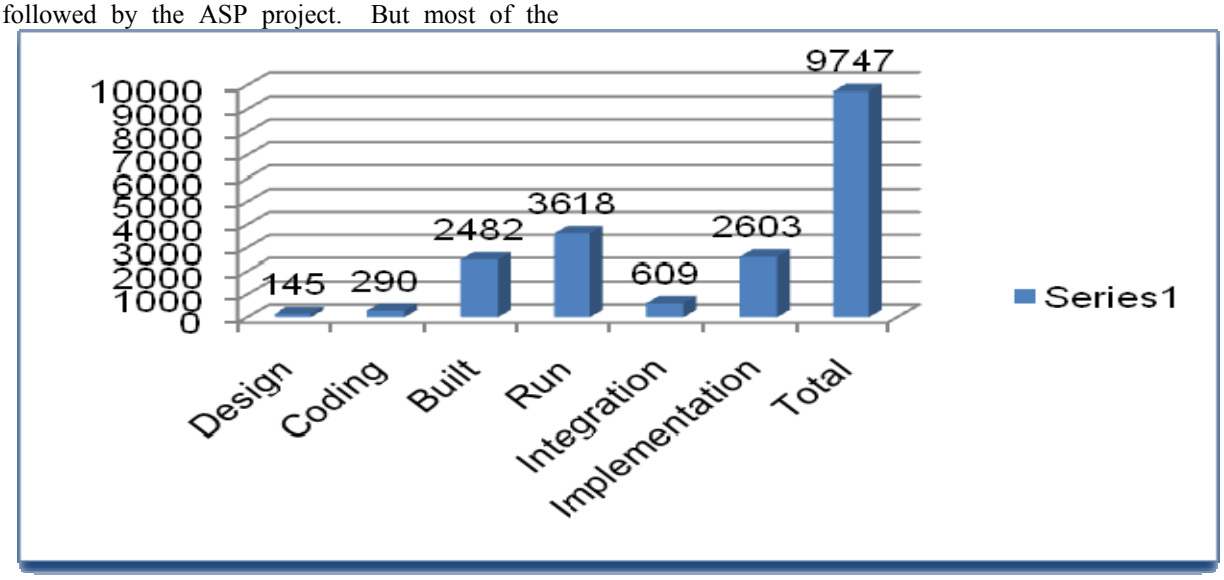


Figure: Error occurrence level

Findings on Available solution

As per the observation of available solution most of the solutions are adapted from the manual which is not able to supported by the built in process. Out of 9747 errors except 670 errors contains the solution from the built-in, manual, help or web links.

Ratio of Learning

Determination of unavailable solutions in different levels and its occurrence is gaining the new knowledge about the errors. These errors are identified machine processing in compare with the existing error list. If the errors are not able to be identified than the errors solutions are not available in the above specified formats. Therefore the learning ratio is specified the unavailable solution error which is occurs out of total errors. It means that the error which identified but the solution not available in the prebuilt-in, help, manual and expert solution. Therefore these identified errors are new knowledge in the error detection and correction process of IDE.

Knowledge Gaining Ratio = Number of New Errors / Number of Total Errors * 100

= (670 / 9747)*100

= 6.87 %

CONCLUSION

The identified unavailable solution errors to be distributed to the experts for their valuable suggestion and solutions. How they solved these errors are added into the knowledge base which is

ISSN: 2076-3328 www.ijric.org E-ISSN: 2076-3336

the input of the next IDE project. Therefore the learning ratio will be reduced over the period of time.

REFERENCES:

[1]. Rajkishore Prasad, Dr A.K Sinha, Dr.Rajeev Ranjan KISAN*: An Expert System for Soil Nutrient Management,AFITA 2002: Asian agricultural information technology & management. Proceedings of the Third Asian Conference for Information Technology in Agriculture, Beijing, China, 26-28 October, 2002

[2]. Clancy W.J.: The Epistemology of a Rule Based Expert System; A Framework of Explanation, AI, Vol20 pp 215-225, (1983).

[3]. Durkin, J. : Expert System: Design and Development, Prentice Hall, New York, NY, (1994)

[4]. Peter S.: Expert system development in prolog and turbo-prolog,GT publication, 1995, New Delhi

[5]. Weis, S.M. and Kulikowasaki,C.A.,A practical guide to designing expert system:, Rowman and Allanheld NJ, USA (1984).