International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 9, Sept.-2014, pp: (257-267), Impact Factor: 1.252, Available online at: www.erpublications.com

A review on significant enhancements and the decade applications of Software Engineering

Anuj Rastogi[1]
Systems Engineer
Infosys India Limited, Hyderabad, India

Keywords: Applications, Database, Enhancements, Reusability, Software Engineering, SDLC, Tools.

INTRODUCTION

Ever since software has emerged there has always been strive to organize it. This organization was the fertile ground for the development of software engineering. The application of software engineering in software has provided software with the recognition of an engineering trade. Ever since the germinal stage of Software Engineering there has been a persistent effort for improving it, for it indirectly improves the way the software organization is done. Moreover, with the continuous improvements, the software engineering capabilities have been so bolstered, that it has transgressed to perform to an extent that it has made such projects possible which otherwise would have been just thoughts. Hence the document here is an attempt to edify some of the distinct enhancements in the software engineering and underscore the finest applications of software engineering.

ENHANCEMENTS:

The software engineering is relatively a novel idea for software development. Since its infancy around thirty years back, in such a short span, it has undergone some ground-breaking modifications and enhancements. Here are some of the significant enhancements that have changed the way the Software Engineering has evolved.

In the current context all software engineering projects require development models prior to the pragmatic execution. This is to assist in developing and modelling the entire work plan of their development. Due to variegated needs of the consumers, all software are novel in their own way. This begets a problem. Since, the software engineering application technology is developed to determine the process of software development, so with the changing nature of one the other also needs to change. This challenges two basic motives of engineering. The reusability of software engineering technologies and the prior experiences. Also, the first problem emanates the second. The very purpose of software engineering, as to align the software development process with the principles of engineering is vitiated. Since engineering supports reusability, organized process and development over prior experiences. Hence, there is a dire need to come up with an idea through which the same factors can be imbibed in software engineering. In [1]-[2], a model was propounded which can promote systematic reusability in application domains of software engineering technology. This was done through explicit models of application domains.

The elaborate model for application domains of software engineering technologies consisted four components: head, intuitive domain model, operationalization, operational domain model. The head related the software engineering technology domain to its task or goal and viewpoint. Intuitive and operational domain model have discreet schemas. They allow for knowledge regeneration with a descent structure and intuitive knowledge representation. The intuitive domain model proposes the model intuitively, on the basis of knowledge used in software program. And the final decision on the selection of pre-existing technology be used, that best suits the current context, can be taken with the help of operationalization domain model. This relates to the real world process that can be effectively determined. An elaborate structure of technology domain model is shown in the figure 1.

In the current context, the interface is responsible for creating a user perspective of the software and at the same is also responsible for explicating the basic functionality of software. Thus, it is a dire need to design it to be more familiar and interactive for the user.

In [3] a novel method for designing the user interface through the applications of Software Engineering principles was presented. A comparative account was drawn between the process of designing the user interface, for software and the design of software development procedure, for software development through Software Engineering. Maslow's Hierarchy theory provides a principle for art designers according to which the minimum level needs must be met, with

design, if the task to secure high level design needs to be attained. This is similar to the Software Engineering as well, wherein the first stage of requirement elicitation has to be met if further High-Level-Design needs to be proceeded with. Thus, the entire functionality of the software is decided based on this step. Hence, like in the interface we have low level work which is essential to pursue a high level task, similarly in software engineering we have the same scenario. Thus, it was supposed that the software engineering principles can be used in the interface design because of the analogy. Further, it was delineated that just designing the interface through art and beauty is not equivalent to its applicability and reasonable outcomes. The design of the interface should be done based on the requirements process and software demands. Hence, there is a need to interconnect Software Engineering with the Design Engineering, as Software Engineering helps in requirement gathering for the same. An interactive design development mode was proposed, wherein the Software Engineering development mode handled, the prominent technology and procedures with the prominent artistic patterns. This interactive design development mode was not just a mere mixing of two techniques but was ramification of the realization of the fact that, the interface of software was a rule developed by the user. Hence, design development mode is the optimization and combination of two original modes. The article thus, described the importance of user perspective in design domain. Also, the need of interconnecting software engineering with design engineering.

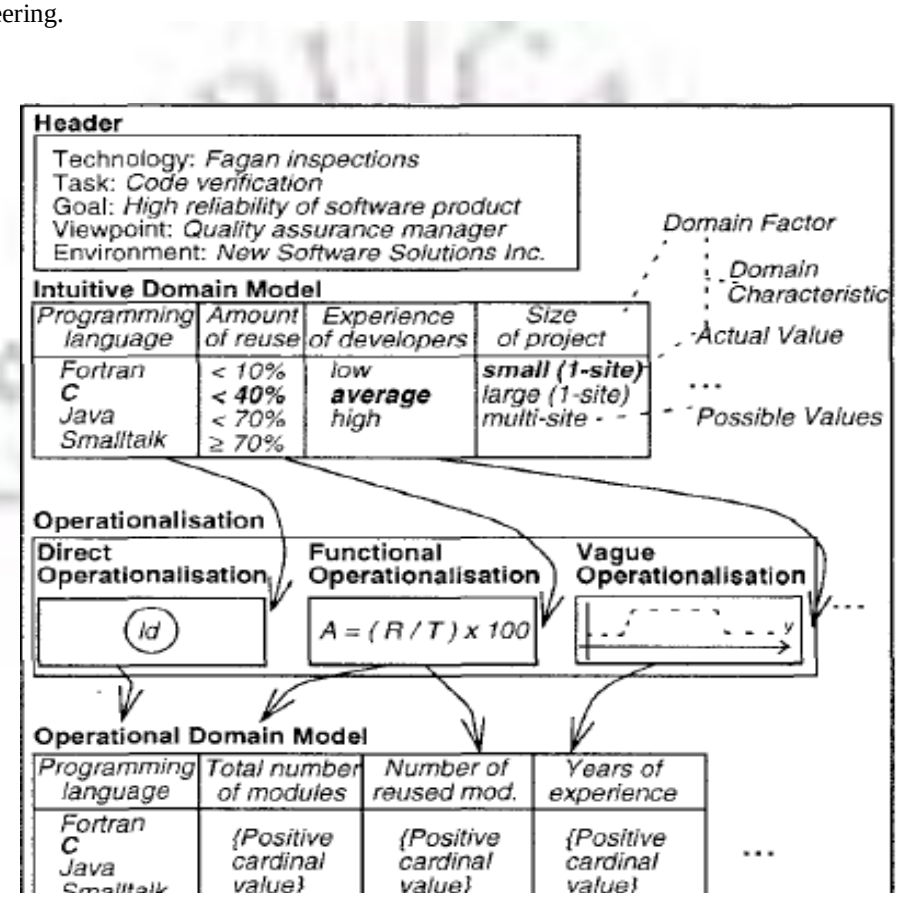

Figure 1. Technology Domain Model-Structure

In very small software enterprise (VSSE), due to small employee strength and lack of management, only a small amount of time can be spent on Requirement Engineering Process. Thus, for the sake of their performance and considering their low success rate, there is a need to bring a radical change in the requirement engineering process. In [4]-[8], an approach was proposed to manage such tasks in VSSE. The fundamental basis is to automate the Software Engineering requirement practices and present a collaborative effort. The software name was SRM (Software Requirements Manager). The ramifications of its implementation show that the application significantly improved the project's Requirement Engineering practice by the help of collaborative effort. The continuous implementation of this tool since 2007 has improved it and have largely focused upon co-ordination and communication techniques among the workers. In VSSE it's a requirement to keep the process cheap in terms of wealth as well as efforts. This demanded automation of the process. The important features of SRM are: SRM is capable of managing multiple projects and since software requirements, design artifacts are the rudimentary needs of the software project so the tools web interface has evolved.

Tool	Semester	Project	% Accompl.	# of Alarms	# of Errors x Proj.	% Errors x Req.	% Lost User Req.
Without SRM	2006-1	Portal of DCC news	71%	---	---	---	0%
	2006-1	Portal for the Continuing Education Area - DCC	67%	---	---	---	4%
	2006-1	Files Sharing System for P2P Networks	54%	---	---	---	15%
	2006-1	Repository and Management System for XML Schema files	56%	---	---	---	0%
	2006-2	Software Repository to Support Educational Activities	80%	---	---	---	0%
	2006-1/2	PASSIR - Collaborative System for P2P Mobile Networks	70%	---	---	---	10%
With SRM and Correspondence Matrix	2007-1	Publications Management System for the DCC	50%	16	16	25%	0%
	2007-1	Mobile Workspace for Firefighters	64%	0	0	0%	0%
	2007-1	System to Share Files and Applications in MANETs	50%	8	8	57%	0%
	2007-1/2	Graduate Programs Management System for the DCC	72%	18	16	25%	0%
	2007-2	Portal de la Intranet del DCC	74%	5	4	11%	0%

Figure 2. Table shows the projects carried out with and without the use of SRM-1st Version

Also, SRM supports input and output data flow. A person can modify the data depending upon the role and thus, the corresponding accessibilities. Project Manager begets projects, Analysts manages the requirements of the project and user, Designer supervises the design artifacts and maps these with the defined software requirements. The other roles are programmer and tester. Also, SRM has a 3 layered architecture. In the lower and middle strata manager represents independent sub-systems except the design and test manager. They are dependent on the services of the requirements manager. The upper layer extends services developed by manager i.e. CRUD (Create, Read, Update, Delete). As, each person in the VSSE have sporadic intervals of work, the lack of co-ordination results in vague information exchange. With this tool, a person can update the data corresponding to his responsibilities. Thus a co-ordination is secured for the information exchange.

The critical success factors (CSF) are properties of software engineering and determine the success factors for identifying potentially expert systems. There are various CSF's existing but 3 new CSFs were recently incorporated into this list, by the help of the research done by the graduates of University of Southern California (USC). In USC 10 Knowledge Based Software Engineering (KBSE) prototypes were developed. It was deduced that existing CSFs were unable to ensure the ubiquitous utility of these application prototypes and were inefficient in explaining the differences on their utility scale. In [9]-[13] this scenario was discussed. In the upshot it was realized that three additional CSFs were needed to explain these differences. The prototype application can be grouped into three software engineering task ramifications: Process Assistance, Software Architecture and Reuse Assistance and Cost/schedule/risk Assessment.

A process Assistance model is selected on abstract rationale without prior experience. The approach that follows entails application program, enabling the manager to identify the best process model, based on the needs. This is done with the help of a case study at the backend between the present challenge and the past experiences. The major comparison domains constitute-Process Aspects and Product Aspects. Limitation of such an approach is the fact that it can prove helpful only when there is a leeway and realistic top level diagnostic is to be done. Preliminary design is a complex process requiring co-ordination of many reviews. For any particular scenario a design is chosen based on the past experiences. Thisobtrudes, that the limitations cannot be transcended. Also, Software Architecture balance diagnosis helps the engineer, in designing and analyzing the design and analysis of complex software architecture, as in-apropos forces can create imbalance. In the approach followed, for the same, there is a schema to represent software systems connections and components. Imbalance is represented as set of production rules. The limitation of the approach is its insensitivity for dynamic systems. Again, in cost/risk management the forward chaining rules are added corresponding to the need-capability tables. It is assumed that, whether the present capabilities or technology, to say, are used to

achieve current needs. This assessment is highly scarce in the context. For transgressing limitations in the utilities 3-CSFs were proposed.

One was the core decision driver, making use of core abstractions: (1) abstracting less significant aspects which help take better software decisions and which defines the solution space and is a framework of all the rules. Second was that, CSF was Unambiguous and ease of mapping from facts to abstractions helped to map requirements directly to interface. It thus, helps in creating the better interface. In requirement engineering, complexity of inputs was a deterrence. The approach propounded in CSF was, to choose the solution description language same as problem description language. The third CSF was Critical Mass Option Representation. The more successful applications have a critical mass levels of software engineer's practical concerns. Some have modicum of rules, in the process model selection assistance. They succeeded because of well supported process models. Some succeeded by adding subtle areas, in critical mass way. Thus CSFs were found useful in explaining the difference in the utility of the prototypes
.

For any software development team, it is a dire need to resolve performance related issues at the time of designing. For doing this Software Performance Engineering (SPE) depends on various language, with the help of which we are able to translate the structural models to formal models.The issues while translation such as active and reactive objects, synchronous and asynchronous invocations are major impedances, deterring the exact analysis of performance. They are an even bigger challenge while dealing with distributed systems. To solve these performance issues and increasing complexity of the software poses a dire need to automate that process.

In the papers [14]-[17] one-to-one translation technique was presented to translate the software sequence and deployment model to the corresponding FSP Finite State Process model. The resulting model was then out under a test by the application of tools such as LTSA, for performance analysis. Although, a lot of models are used for the translation but all such models are fettered to limitations. FSPs are abstract machines for modelling the behavior of distributed system, which gets repeated. Till now no such algorithm exists, capable of direct mapping between the UML and simulation model. For the same, an algorithm was also presented for automatic transformation of software deployment and sequence diagram into FSP. With the corresponding FSP thus obtained performance factors such as response time, length of queues object population can be estimated. This also contains the performance tags. As we know that in a distributed system the only link is the synchronous activities linking the rest together. Thus, for transforming FSP transformation of objects, threads, CPU and communication links to FSP is required. The transformation process involves transformation of every object to FSP. The order of operations shown on the timeline help determine the order in the corresponding FSP process as the tasks. Also, the corresponding messages in timeline are translated to actions in FSP.

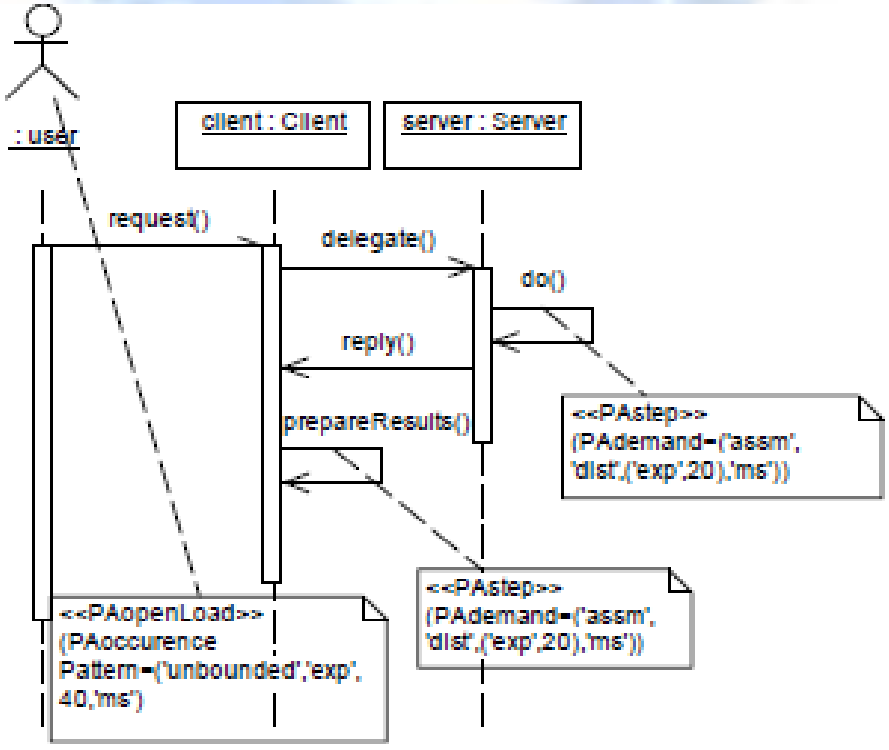

Figure 3. UML with Performance Tags

For the translation of multi-threaded objects a predefined number of instances are created. The incoming and the outgoing messages help determine the type of the object. Each node of a UML help determine the number of CPUs installed in that node. Based on all this an algorithm was presented as:

```
Algorithm FSPmodelUMLtoFSP(model m)
Begin
FSPmodel f=create FSPmodel();
For each SequenceDiagramsd in m
For each object o in sd
f.AddObject(o);
For each DeploymentDiagram d in m
For each Node n in d
f.AddNode(n);
For each Link l in d
f.AddLink(l);
f.Synchronize(m),
end
```

Here O, L, M, C denotes Objects, Links, Sets of Messages and CPUs respectively. For each o on O, there is Name, type, First Message, Last Message, Message set. For each l in L, there is Name and Number. For each m in M, there is Name, Sender, Receiver type, communication delay, computational delay. For each n in N, there is Name and Number. Thus, the new approach proved highly effective for the distributed systems.
For each software, maximum effort needs to be put in to assure smooth Software Development Life Cycle (SDLC). A number of tasks need to be carried out at the process level and hence equal importance need to be given to each task, big or small. There needs to be an even distribution of time due to which for harder tasks there is less time and for simpler tasks there is a wastage of time. This effects the overall efficiency of the project.
.
In [18]-[25], a technique to overcome this fallacy was addressed and the utility of the Pareto-chart or 80/20 rule in software process was bolstered. According to this, 20% efforts beget 80% results. Hence, one should first identify those 20% factors ignoring the rest. This promotes the application of this rule in the work task part, to optimize the effort, to increase efficiency. The research was carried out on the software firms. Then 80/20 was applied to identify 20% tasks yielding maximum efficiency. After research and analysis of 144 tasks, 45 were found out to be of less utility. The list of such tasks was published in [23]. Also a comparative result was shown for waterfall model with and without the application of 80/20 rule in figure 4. Here, FPA is Total functional points of the phases of the software waterfall process model. N is total environment i.e. non-functional influence factor. CAF is the complexity adjustment factor. AFP is adjusted functional points, KLOC is kilo lines of code, E is the Effort, TDEV is the project time duration, SS average staff and P is productivity.

APPLICATIONS

The development in software engineering have always promoted the researchers to transgress the limits of the subjects and the basics of the subject to the fullest. This has yielded through a passage of time such significant applications, of software engineering, which otherwise would not have been a possibility. Some of the decade applications of software engineering are discussed below.

The space shuttle Primary Avionics Software Subsystem (PASS) is one of the most important applications of software in real time projects. Developing perpetually through 1970 its one among the most intricate soft wares, divulging the fact regarding the finest applications of software engineering. It has thus proved the capabilities of software engineering in handling perplex projects. In the NASA space shuttle the use of software reliability engineering (SRE) strengthened the fact, as to how it can be used in prediction support verification, validation of software and developing test strategies. The success has buttressed the Engineering regarding the extraordinary capabilities of software engineering. The SRE is currently used by Lockheed Martin Space Information Systems.

In the paper [26]-[28], an idea of how a software tool, for developing internet applications with e-business process, can be developed by integrating it with IDEFO tool. It was deciphered how the adaptability can be assured by the use of

both tools, simultaneously. E-business can be judged by accessing the supply chain and interactions of internal applications. To make e-business connections adaptable it is a dire need to develop them using software engineering, for using it imbibes in them organization and technical infrastructure. The systems are more adaptable when the following are created in order: (1) business architecture (2) business design (3) process design (4) technical architecture (5) technical design. In the e-business IDEFO is a prominent tool, for it can distinguish between dataflow. Thus IDEFO are used for business architecture and design. Further open process tool is used to bring such designs into pragmatic forms via internet browser. Here, event and responses are identified as parameters of business interface and it also defines how responses will be made. The rules that it defines include SLA, service level, response time etc. Each component can perform the process in itself and its design is an IDEFO model.

CAF[3]	Before application of 80/20 rule in Waterfall Model	1.138	1.138	1.138	1.138	1.138	1.138	1.138	1.138	1.138	1.138
	After application of 80/20 rule in Waterfall Model	1.138	1.138	1.138	1.138	1.138	1.138	1.138	1.138	1.138	1.138
AFP[4]	Before application of 80/20 rule in Waterfall Model	584.932	584.932	584.932	584.932	584.932	584.932	584.932	584.932	584.932	584.932
	After application of 80/20 rule in Waterfall Model	415.37	415.37	415.37	415.37	415.37	415.37	415.37	415.37	415.37	415.37
KLOC[5]	Before application of 80/20 rule in Waterfall Model	74.871	31.001	34.5109	25.1521	35.096	22.274	26.907	20.473	29.247	40.360
	After application of 80/20 rule in Waterfall Model	53.167	22.015	24.51	17.86	24.922	15.784	19.107	14.537	20.768	28.661
E[6]	Before application of 80/20 rule in Waterfall Model	222.97	88.34	98.87	70.93	100.63	62.294	76.132	57.14	83.097	116.54
	After application of 80/20 rule in Waterfall Model	155.646	61.667	69.02	49.512	70.246	43.485	53.145	39.887	58.007	81.35
TDEV[7]	Before application of 80/20 rule in Waterfall Model	19.51	13.72	14.32	12.63	14.42	12.02	12.97	11.63	13.41	15.25
	After application of 80/20 rule in Waterfall Model	17.02	11.972	12.495	11.014	12.579	10.484	11.314	10.145	11.696	13.3
SS[8]	Before application of 80/20 rule in Waterfall Model	11.428	6.437	6.902	5.617	6.978	5.183	5.869	4.913	6.197	7.643
	After application of 80/20 rule in Waterfall Model	9.145	5.151	5.524	4.495	5.584	4.148	4.697	3.932	4.959	6.116
FP[1]	Before application of 80/20 rule in Waterfall Model	514	514	514	514	514	514	514	514	514	514
	After application of 80/20 rule in Waterfall Model	365	365	365	365	365	365	365	365	365	365
Languages to be used for software development using the organic mode of COCOMO Model		**C**	**C++**	**C#**	**HTML**	**JAVA**	**ORACLE**	**PL/SQL**	**SQL**	**VB**	**ASP**

Figure 4. Impact of application of 80/20 Rule in waterfall model

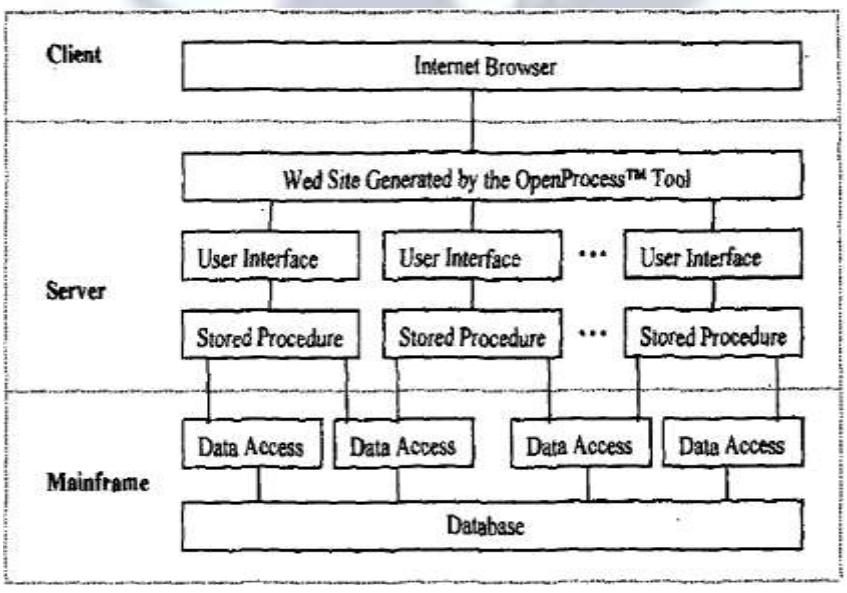

Figure 5. Open Process Model Technical Architecture

The process modelling is done by loading the IDEFO model into Software Engineering tools, with the entire description document. A hierarchy is created with business at the top and process at the lowest end. Also, IDEFO models are mutated into a card describing each process. Each card has a tab and also, "parts per days" metric is encoded as a hyperlink to permit drill down. These hyper link appearance can be customized using browser. Thus an integration of IDEFO with software engineering tools makes the e-business adaptable as well as organizes the process.

Currently, the design and SDLC for all software applications are done using software engineering tools. But in multimedia applications it is entirely different as the requirements from the user in the beginning are indistinct, which makes planning and coding difficult. In the papers [29]-[31] traditional software engineering approach was analyzed and the possibilities of its implementation to the multimedia applications were mulled over, with the scope of further enhancements in the same. For the same an approach was developed. In this approach, a prototype of the system is first developed based on minimum requirements. Once the system is operational, based on requirement clarification the enhancements can be done. For the initial phase a formal set of requirement is made, specifying the behavior of components. Also, prototyping is at times supported by specialized 4GL language. In the beginning model is made just to clarify the requirements. To further enhance prototyping MICE environment is developed. MICE has powerful abstractions like TAO (Tele Action Object), integrated toolkit based n abstraction and swift prototyping. TAO is a powerful object having a powerful set of knowledge to adapt to vacillating environment. Thus, TAO knowledge base is graphically build with the help of IC build and TAO builds help to make the graphical UI of TAO. TAO builder makes TAO abstractions even powerful. Hence TAO becomes an important part of MICE for GUI is important for prototyping. It bolsters the user and whets him for further requirements. This helps in enhancing the software.

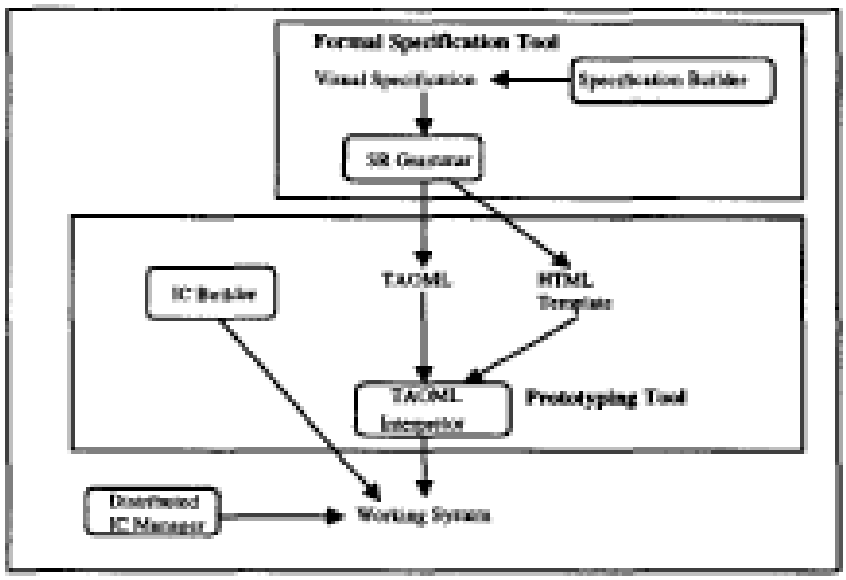

Figure 6. MICE Model

Electromagnetic transient program (EMTP) is a software for managing the computational parts of PSA (Power System Analysis). EMTP was earlier written in FORTRAN-77. This language is best suited for computational works, but has become obsolete. The continuous up-gradations in EMTP has yielded a more perplex code, difficult to maintain. Thus, the further modifications in code will make it even more intricate. Since, the GUI of FORTRAN-77 is very rudimentary, it is difficult to manage the inputs of user. The application at the same time cannot be migrated to OOPs, for FORTRAN supports computational ability which is the dire need of EMTP. Hence upgrading language needs to be chosen by proper Software Engineering Technique. In [32] it was delineated as, how the software engineering basics help choose the language for upgrading the application. By the results obtained, FORTRAN-95 was identified as the correct substitute. Although, more advanced languages such as JAVA\C\C++ are present but FORTRAN-95 was chosen due to several reasons. There are still compilers in FORTRAN-95 which support some of the features of FORTRAN-77. It is also suitable to computational tasks. Since matrix etc. are important for the formulation of electrical networks so all of them are supported by FORTRAN-95. In FORTRAN-95 the concept of function is overridden by "Scope". A scope is defined in a program and can use other scope in that structure. Also, the private keyword is defined.The file keyword transfers information value into the matrix.

```
CASE('put_in_Yn_ss') !* Numeric contributions to Yaug_c at w <A name="put_in_Yn_ss">
   RLC%gz=RLC%R+jz*( w*RLC%L- RLC%C/w )
   RLC%g=RLC%R+(2/Dt)*RLC%L+(Dt/2)*RLC%C
   RLC%gz=inv_vector(RLC%gz); !1/( R+j*(wL-1/(wC) )
   DO k=1,SIZE(RLC)
      CALL fill(RLC(k)%knode,RLC(k)%knode,RLC(k)%gz       )    !knode, knode
      CALL fill(RLC(k)%knode,RLC(k)%mnode,c_mone*RLC(k)%gz )   !knode, mnode
      CALL fill(RLC(k)%mnode,RLC(k)%knode,c_mone*RLC(k)%gz )   !mnode, knode
      CALL fill(RLC(k)%mnode,RLC(k)%mnode,RLC(k)%gz       )    !mnode, mnode
   END DO
.....
CASE('put_in_Yn')  !*contribute to Yn in td, <A name="put_in_Yn">
   DO k=1,SIZE(RLC)
      CALL fill(RLC(k)%knode,RLC(k)%knode,RLC(k)%g)
      CALL fill(RLC(k)%knode,RLC(k)%mnode,mone*RLC(k)%g)
      CALL fill(RLC(k)%mnode,RLC(k)%knode,mone*RLC(k)%g)
      CALL fill(RLC(k)%mnode,RLC(k)%mnode,RLC(k)%g)
   END DO
```

Figure 7. CASE Sections used by RLC component

Here the environment used was Microsoft visual studio. This allows linking of GUIs in C\C++ with the FORTRAN and hence for every component there is a GUI. The strong presence of software engineering helps in justifying the requirements of FORTRAN-95 as well as documenting the code. The predefined variable are used for controlling the precision. It was also delineated that there should be certain functions that should communicate to the core code. The CASE section used by RLC component for sending its equation into the core system of equation.

The applications written on the database (DB) are tightly coupled to it, and they provide a lot of tools to develop applications on 4GL. Thus, there is less possibility of migrating 4GL (Generation Language) application to other DB. As this consumes a lot of time and expenditure. In spite of this there is still a dire need to migrate 4GL to latest DB. In [33]-[34] a Knowledge Based Software Engineering (KBSE) tool based application was proposed to convert 4 GL applications into the code that DB supports. There were two approaches proposed. They were the outcome of the ITOC oracle project. The tool developed in the project, was to transfer the ABF applications to be used with oracle developer designer 2000. The design issues of the tool-set and its implementation was supported by Reasoning System Software Refining, KBSE development environment. The approaches were: (1) emulation (2) design recovery. In emulation approach each statement of the source has a corresponding translation in the target. In design recovery information is the key. This is extracted from source and fed into CAISE tool which further generates the target code. For the same, inference tool in CAISE are developed and remaining code can be developed in the target.

To apply both approach in the integrated environment a KBSE tool is used. The source code is first fed into this and then in the later stage it is decided as to what inference level has to be applied. The KBSE tool also has a Software Refinery. This is a KBSE development environment with the re-engineering tool production. The tool contains prior experiences to transform source to target. In KBSE tool, knowledge base is maintained by the earlier conversion experience. Since the transformation rules are applied on abstract level, from translating each node in the source to each node in the target, thus, translation is done using refine language. The inference rules of transformation are: (1) for emulation first order inference is used, to transform only base information (2) in design recovery the base information as well as the combined information derived from two rules is used. It uses second order inferences, also used with (a) fix query columns (b) to identify foreign key. The analysis of both approaches project emulation to be complete and efficient while recovery to be maintainable.

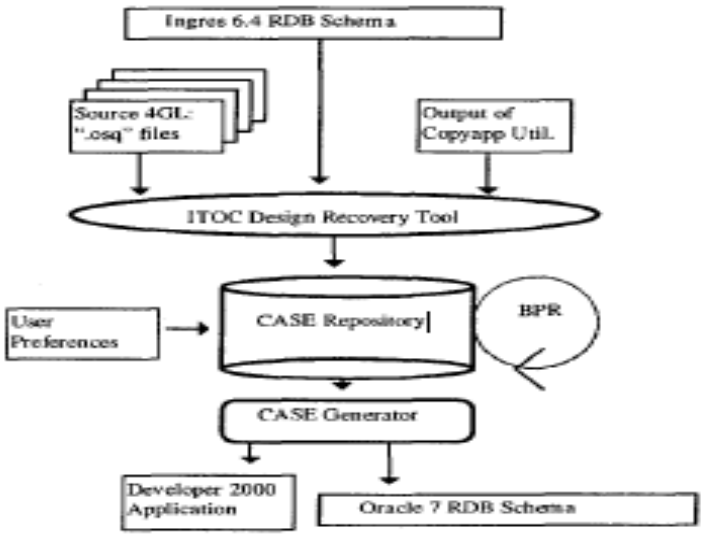

Figure 8. Recovery Approach Design

Real time system is one where the time output is produced is significant. The input to the system is the real world object. There are two stringent constraints the output has to suffice: (1) the output should be accurate (2) the time to process the output should be less for producing the output instantly. Hence the designing and requirement elicitation is perplex. In [35] a review was presented on the complex application of software engineering applications in the design and requirement elicitation. The requirement in such a system are not just limited to user but there are concealed requirements like rate of sensor response, the control system, actuator command etc. For such a complex elicitation process the requirements are transformed into tasks in the SRS so that the time can be adjusted accordingly. In real time there are 3 types of tasks. (1) Periodic (2) Aperiodic (3) Server. The architecture of a real time system is characterized by the presence of an executive that executes the tasks. There are 2 types of tasks: Fixed Computational Tasks and Variable Computational Tasks. In Fixed Computational every task is allocated the CPU time. Each task is assigned a time with a pocket for further growth. Aperiodic tasks are also scheduled as periodic tasks with high priority. In variable Computation cycle there is a high flexibility. The high priority task if arrive in the middle are immediately set. If the low priority tasks occur then they are kept in a queue. In the designing of such system there is a necessity that the scheduling time is kept separate from function time. The former decided by the scheduler while the later by the function module. Periodic, aperiodic and server all tasks should be scheduled. While scheduling an important criteria of timeline should always be kept in mind. The programming language for designing such system should manage the system clock properly and allow the execution of corresponding tasks. Thus with the help of Software Engineering it becomes easy to identify the demands of such intricate systems and develop them. Also reliability testing is easy as the requirement are illustrated distinctly in SRS.

In the present scenario, the software development and maintenance are separated by a very fine line. The rapid creation of software have brought an age of software where novel creation has been ousted. All software developed are on existing software or an enhancement. Thus the SDLC planning can be done using earlier experiences that can help automate the design stages of a new software. Thus, the domain engineering comes into scene. In domain such systems are developed as having the features of earlier software that can be used iteratively to develop target systems. These components have similarities from the earlier systems. The target system can then be generated by tailoring the components. In [36] the technique for framing the Evolutionary design life cycle (EDLC) was proposed. Also, a software engineering environment was discussed to implement in the target system. The EDLC system constitution and the configuration of distributed system was discussed. EDLC exist when there is an existing set of soft wares on which the future family of software need to be developed. Thus taking the earlier experience SDLC can be automated. EDLC has two contents: Domain Engineering and target system configuration. In Domain engineering domain analysis, especially, of the earlier software forming a domain and reusable specifications for the family of systems is designed. In target system configuration depending upon the target needs and specification a set of components is selected from domain to assign it. Each application domain is presented in multiple views depending on the perspective. For distributed systems 2 views are important. Object Communication view and Feature/Object dependency view.

Figure 9. Object Communication Model

In the former, concurrent real world objects are used to represent and the communication between them is represented through messages, loosely/tightly coupled. Each such model has Kernel-"Present in every target", optional- "present in target optionally", variant-"object to meet specific needs". In the later, the objects to support the necessary features are viewed. The domain requirements are classified as Kernel, optional, prerequisites. In architecture each object belongs to component and each feature belongs to design fragments. Architecture description language (ADLs) support the architecture of components and their interconnections. In the paper [42] the design fragments are described using

Darwin. Each component in the design is the Darwin component. Having interface requiring another Darwin component. For each feature in the domain architecture there is design fragment containing information of component type and their interconnections. For communication among component they should be instantiated using "inst" statement.

A software environment to automate the process was also presented. This can generate target systems and can do mappings of their specification, with the existing reusable component. Thus an automation approach can be followed for SDLC. The information in the multiple views of the domain is centralized and drawing the consistency it is stored in the reuse library. Also, generating reusable architectural domain is automated. For every Kernel or optional a Darwin component is there in the reuse library. The design fragments are generated using Design Fragment Generator. According to the specifications of the target system, at the backend, the components are checked for consistency and a target system is generated. The detailed design of reusable component is done and they are stored in reuse library.

References

[1]. Andreas Birk, "Modelling the Application Domains of Software Engineering Technologies", 0-8186-7961-1197 $10.00 0 1997 IEEE.
[2]. R. Prieto-Diaz and P. Freeman. "Classifying software for reusability", IEEE Software, 4(1):6-16,January 1987.
[3]. Xinyu Wei, Zhanfeng Shi and Yiran Wei, "Application or Engineering Methods In Art Design or Software Interface", 978-1-4244-7974-0/10/$26.00 ©2010 IEEE.
[4]. Sergio F. Ochoa, AlcidesQuispe, Andrés Vergara, José A. Pino, "Improving Requirements Engineering Processes in Very Small Software Enterprises Through the Use of a Collaborative Application", 978-1-4244-6763-1/10/$26.00 ©2010 IEEE.
[5]. A. I. Anton, "Successful Software Projects Need Requirements Planning". IEEE Software, Vol. 20, No 3, pp. 44 – 46, May/Jun. 2003.
[6]. J. Aranda, S. Easterbrook, G. Wilson. "Requirements in the Wild: How Small Companies Do It", Proc. of the 15th IEEE Requirements Engineering Conference, pp. 39 – 48, Oct. 15–19, 2007.
[7]. R. Berntsson-Svensson, A. Aurum. "Successful Software Project and Products: An Empirical Investigation", Proceedings of the 5th ACM/IEEE International Symposium on Empirical Software Engineering, pp. 144 – 153, Sep. 21–22, 2006.
[8]. R. N. Charette. "Why Software Fails". IEEE Spectrum, Vol. 49, No 9, pp. 42-49, Sept. 2005.
[9]. Barry Boehm and PrasantaBose, "Critical Success Factors for Knowledge-Based Software Engineering Applications", 1068-3062194 $4.00 0 1994 IEEE.
[10]. Boehm, 19891. B. W. Boehm, 2bftware Risk ManaeementIEFE Computer Society Press, Los Alamitos. CA 1989.
[11]. Bolcer, 19941. G. Bolcer, "User Interface Design Assistance For Scale Software Development",submitted to KBSE 94.
[12]. Green et. al., 19831. C. Green, D. Luckham, R. Baker T. Cheatham, and C. Rich, "Report on a Knowledge Based Software Assistant", RADC Technical Report, June 1983.
[13]. Jackson, 19901. P. Jackson, Introduction to Expal&&ms (2nd. ed.), Addison Wesley, 1990.
[14]. OmidBushehrian, Hassan Ghaedi, "The Application of FSP Models in Software Performance Engineering: A Multi-Threaded Case-Study", 978-1-61284-691-0/11/$26.00 ©2011 IEEE.
[15]. M. Woodside, G. Franks, D. Petriu, "The Future of Software Performance Engineering", IEEE, 2007 .
[16]. S. Balasmo, A. Dimarco, P. Inverardi, A. Simeoni, "Model-based performance prediction in software development", IEEE Trans. On Software Engineering, 30(1): 295-310, 2004.
[17]. S. Balsamo, M. Marzolla, "Performance evaluation of UML software architectures with multi-class queuing network models", Proc. 5th Int workshop on Software and Performance (WOSP), ACM SIGSOFT ,2005.
[18]. Muzaffar Iqbal and Muhammad Rizwan, "APPLICATION OF 80/20 RULE IN SOFTWARE ENGINEERING WATERFALL MODEL", 978-1-4244-4609-4/09/$25.00 ©2009 IEEE.
[19]. Koch, R., 2004, The 80/20 Principle. In: 'Living the 80/20 Way'. pp1-6. Nicholas Brealey Publication, Melbourne Australia.
[20]. Ultsch, A., 2002, Proof of Pareto's 80/20 Law and Precise Limits for ABC-Analysis, Data Bionics Research Group University of Marburg/Lahn, Germany. pp. 1-11.
[21]. Pressman, R., S., 1998, Software Project Management. In: Software Engineering – A Practitioner's Approach, Fourth Edition, pp. 28-38. McGraw- Hill Companies Publication, Inc. USA.
[22]. Mendez, Y., B., 2004, Project Management for Software Process Improvement. In: Proceedings of PMI Global Congress', Prague, Czech Republic.
[23]. Parida, P., 2006, "Essence of Waterfall Model".
[24]. David A. Marca and Beth A. Perdue, "A Software Engineering Approach and Tool Set for Developing Internet Applications", ICSE 2000 Limerick Ireland Copyright ACM 2000 1-581 13-206-9/00/6 ... $5.00.
[25]. Gomes-Cassares, Benjamin, The Alliance Revolution, Harvard University Press; 1998.
[26]. Moore, G., Crossing the Chasm, Harper Business, 1999.
[27]. Pokier, Charles C., Advanced SumlvChain Management, Publishers' Group West,1999.
[28]. Marca, D., Perdue, B., "Business-to-Business Connections", Soft Eng Tech Council, Vol15, No 3; 1997.
[29]. Timothy Arndt, "The Evolving Role of Software Engineering in the Production of Multimedia Applications", 0-7695-0253-9/9$91 0.00 0 1999 IEEE.
[30]. Adjeroh DA, Nwosu KC (1997) Multimedia database management - requirements and issues. IEEE MultiMedia 4:24-33.

[31]. Arndt T, Cafiero A, Guercio A (1997) Symbol Relation Grammars for Teleaction Objects. Technical Report, Dipartimento di InformaticaedApplicazioni, University of Salerno.
[32]. Jean Mahseredjian and Pierre Lacasse, "Software engineering environments for recoding large scale applications", 0-7803-6672-7/01/$10.00 © 2001 IEEE.
[33]. John V. Harrison, Anthony Berglas and Ian Peake, "Legacy 4GL Application Migration Via Knowledge-Based Software Engineering Technology: A Case Study", 0-8186-8081-4/97 $10.00 0 1997 'IEEE.
[34]. Elmasri, R. and Navathe, S., "Fundamentals of Database Systems", BenjaminKummings Publishing, 1994.
[35]. Ali Behforooz Frederick Hudson, "Software Engineering for Real-Time High Reliability Applications", O-7695-0028-5/99 $10.00 0 1999 IEEE.
[36]. H. Gomaa and G.A. Farmkh, "A Software Engineering Environment for Configuring Distributed Applications from Reusable Software Architectures", 0-8186-7840-2/97 $10.00 0 1997 IEEE.