Knowledge Representation for the tool used for Requirement Segregation in Software Development Process

Hema Gaikwad
Symbiosis International University
Symbiosis Institute of Computer Studies & Research
Pune, India
hemagaikwad2005@gmail.com

Abstract— Software development is the major activity of Information Technology. We can engineer the software with the help of various process models. System Development Life Cycle (SDLC) is the base for all process models. All the other process models (for example RAD, Spiral model, Component based model, Increment model and even Agile model) derived from SDLC. The Computer Aided software engineering (CASE) tools are software and we can use these tools to develop any application software. If we use these tools in the software development process we can reduce the manual work, efforts and save the time also. There are very low probability of failure the system. CASE tools are mostly associated with all phase of SDLC. Requirement engineering is the most important activity of software development life cycle. We have to assign more time on that phase because if we assign less priority and less time to that phase, we can't able to collect the proper requirement and this lead to unsatisfied deliverable. Unsatisfied and inefficient deliverables is not accepted by the client also. So there is a compulsory need to investigate some new CASE tools specially associated with Requirement Engineering which helps to optimize the software development process as well as increase the performance. As we know that Tool cannot work without providing knowledge. There are various ways to represent knowledge such as Inheritable knowledge, Relational Knowledge and Procedural Knowledge. This paper explores the CASE tool as well as various knowledge representation methods for Requirement engineering phase of Software Development Life Cycle and specially focused on requirement segregation.

Keywords- CASE, FR, NFR, RAD, SDLC

I. INTRODUCTION

Software is a set of instructions, software has many features like it is always engineered not manufactured and the most important feature is it doesn't wear out. Engineered means development. So software engineering is a systematic approach through which we can developed the disciplined application. Pressman stated that Software Engineering is a systematic ,disciplined quantifiable approach to the development, operation and maintenance of software; that is the application of engineering to the software[1]. The computer has become an invaluable tool throughout the corporate world and has greatly enhanced productivity. However, most applications have been targeted at making individuals more productive. With the development of networks, computers have expanded their role somewhat and now provide easy ways to communicate, share data, and share applications. Still, the environment provided by most computer systems is largely targeted at the individual. It is quite common, however, for groups of individuals to collaborate on a project. Computers support the individual work done by group members but provide only token support for group interaction. The details of this interaction is left for the users to organize. There is a need for new tools that provide an environment where a group can cooperate on a project. Most work done in this area has thus far been targeted toward office automation and communication. However, there has been very little done to provide a group tool that supports the engineering design process. The objective of the computer supported cooperative work (CSCW) project is to provide an environment that supports this process. In particular, this environment provides collaborative tools that support computer-aided design (CAD) and computer-aided software engineering (CASE)[2]. Software development is a complete process and has many phases like Requirement engineering, Planning, Analysis and Design, Development, Verification & Validation and Deployment, other phases of SDLC represent the sub processes. All the sub processes of SDLC will execute in sequential manner. Requirement Engineering is the most important and time consuming activity. It includes requirement gathering, requirement analysis and requirement segregation. Now a days we can automate each and every sub process of SDLC by using various CASE tools. CASE tools are known as CASE, Computer aided software engineering. CASE (Computer Aided Software Engineering) tools have played a critical role in improving software productivity and quality by assisting tasks in software development processes since the 1970's [3]. In recent years, computer-aided software engineering tools have emerged as one of the most important innovations in software development to manage the complexity of software development projects (1998). The benefits of using CASE tools include: increasing the speed with which software is developed and improving the quality of the developed system ,reducing the cost of software/system development and providing a uniform platform for software/system developers to present information and knowledge compactly for ease of communication[4]. CASE tools are a class of software that automate most of the software development activities, examples are Requirement tracing tools, Analysis and design tools, Interface design and development tools, Programming tools, Integration and testing tools and Test management tools etc. An important role of a CASE tool in software development is to serve as a methodology companion by providing methodology support. A tool that provides methodology support should generate an error or warning when the rules of the methodology are violated (Hatley, 1988)Vessey, Jarvenpaa & Tractinsky (1992)[4]. In this paper, we propose requirements for a platform that supports building CASE tools for software engineering research. The goal of such a platform is to provide core features, which are necessary for such tools such as diagram based visualization, versioning and collaborative modification of models. The core idea is that researchers spend a common effort in implementing these features rather than implementing the requirements over and over again[5]. Computer-Aided Software Engineering (CASE) tools are important for systems/software development and implementation. CASE tools can be used to reduce the cost and time of system/software development while improving the quality of the product developed. CASE tools help to save our time and efforts and indirectly increase the productivity and quality of the software. Tools offer many benefits to the project team to building the large scale system. In this paper we only focus on the requirement engineering CASE tools. To determine the values for the “tool use” factor, Checkpoint requires the input of tool effectiveness raging from 1 (Excellent) to 5 (Poor). Tools are categorized by the phases (Project Management, Requirements, Design, Coding, Testing and Debugging, Documentation, and Maintenance methods/tools) they support. Checkpoint captures the information about the degree of tools integration through the input of CASE Integration.

Table 1 Case Integration Criteria

Poor	Below Average	Average	Good	Excellent
No use of CASE tools in the company/organization	Company/organization is using CASE tools, but not on this project	Limited amount of CASE on this project	Some use of CASE tools in multiple phases	Integration of CASE across all phases of life cycle

The criteria for the CASE Integration is shown in Table 1. Even if Checkpoint evaluates Evaluation Team capability multipliers Tool capability multipliers CASE tools by quality and the degree of integration across phases/projects, it does not account for the effect of tool maturity and user support [3]. We know that we collect the requirement using various tools such as Interview, Questionnaire, Observation and Review of documents. These methods are known as Fact finding methods or Fact finding tools. Questionnaire may be either Open questions or close questions. Open question means Question is define and the space is given for writing the answer. Fill in the blanks, Dichotomous questions, Ranking scale, Multiple choice and rating scale questions are example of close questions. Another category of questions are W questions such as WHAT, HOW, WHICH, WHO,WHERE, WHEN, WHILE etc. Again W questions may be either open or close questions. Some of the more typical functional requirements include Business requirement documents, Business rules, Transaction corrections, adjustments and cancellations, Administrative functions, Authentication, Authorization levels Audit Tracking, External Interfaces, Certification Requirements, Reporting Requirements ,Historical Data Legal and Regulatory Requirements. Some of the more typical non functional requirements include Performance, Response, Time,Throughput,Scalability,Capacity,Availability,Reliability,Recoverability,Maintainability,Serviceability,Security,Regulatory,Manageability,Environmental,Data integrity, Usability and Interoperability. Current object-oriented CASE tools are useful for recording and gaining insight into OO models. They offer extensive support for especially the analysis and design of object-oriented software. The possibility to generate skeleton code motivates development teams to construct a good design before coding. Developers subsequently add the remaining code; CASE tools then offer support for keeping model and code consistent. Extensive possibilities for maintaining models and producing documentation are also provided. Contemporary CASE tools thus provide for more efficient communication within development teams and therefore facilitate overall maintenance[6].Within the context of CASE tools, a constraint is a rule that defines the range of options available in performance of a task. In the very simplest sense of the word, most tools constrain the user to the application of a particular modeling method, by simply not supporting alternative modeling mechanisms. Constraints can, however, be more sophisticated. For example, constraints on the structure of models might include a rule that every process in a data-flow model must have at least one input data flow and at least one output data flow. Or, constraints on the sequence of design activities might specify that a parent process must be specified before its child processes. These kinds of constraints, referred to here as methodological constraints since they are specified by design methodologies, are implemented in CASE tools to guide the user toward a ‘good’ or ‘correct’ design. By instantiating particular methodologies, the builders of CASE tools attempt to control the behavior of tool users -

by communicating (through the tools) the necessary and appropriate processes in software development[7]. Knowledge representation is a combination of data structures and interpretive procedures that leads to knowledgeable behavior. Therefore, it is required to investigate such knowledge representation technique in which knowledge can be easily and efficiently represented in computer[8]. Knowledge representation (KR) and inference mechanism are most desirable thing to make the system intelligent. System is known to an intelligent if its intelligence is equivalent to the intelligence of human being for a particular domain or general. Because of incomplete ambiguous and uncertain information the task of making intelligent system is very difficult. There are various knowledge representation schemes in AI. All KR techniques have their own semantics, structure as well as different control mechanism and power. Combination of two or more representation scheme may be used for making the system more efficient and improving the knowledge representation[9]. This paper states that Knowledge is a term and it represent progression that starts with data. Knowledge representation model progress from data through information and knowledge to wisdom[10]. This paper states that instead of using single knowledge representation method like Bayesian Network ,Facts and Production Rules, Semantic nets, Conceptual Dependency, CYC, Frames, Scripts, and Neural Networks. We can use the hybrid KR formalism consisting of two or more different sub formalisms [11].

II. DISCUSSED PROBLEM

CASE tools helps in complete SDLC process, they are the software and try to reduce the human time and human efforts. Various CASE tools are available in the market for every phase of SDLC. We know that requirement engineering is the first and most important phase of SDLC. So we have to give maximum time to that phase because if we gather the requirement carefully we can deliver the efficient system otherwise the same SDLC process will repeat. So CASE tools should be present for requirement engineering phase, but the thing is that only tool is not sufficient we have to develop knowledge also.

III. SOLUTION

The aim of this research is to suggesting a model that we can use in requirement engineering phase as well as develop the knowledge by using various methods. A questionnaire instrument was selected in order to collect the large amount of data required to gain a comprehensive understanding of the Requirement Engineering process. Additionally, the questionnaire acted as a time saving tool. The questionnaire was based on a questionnaire developed by the Requirements Engineering. The questionnaire consists of three sections covering (1) background details of the participant, company and project, (2) the RE process, and (3) RE techniques used. Many closed-ended questions were used to minimize the length of the questionnaire, however participants were offered an “Other-please specify ”option to prevent forced answers from occurring. In the proposed model there are total 4-5 components such as stakeholders, Requirement bunch, W questions private cloud and expert system.

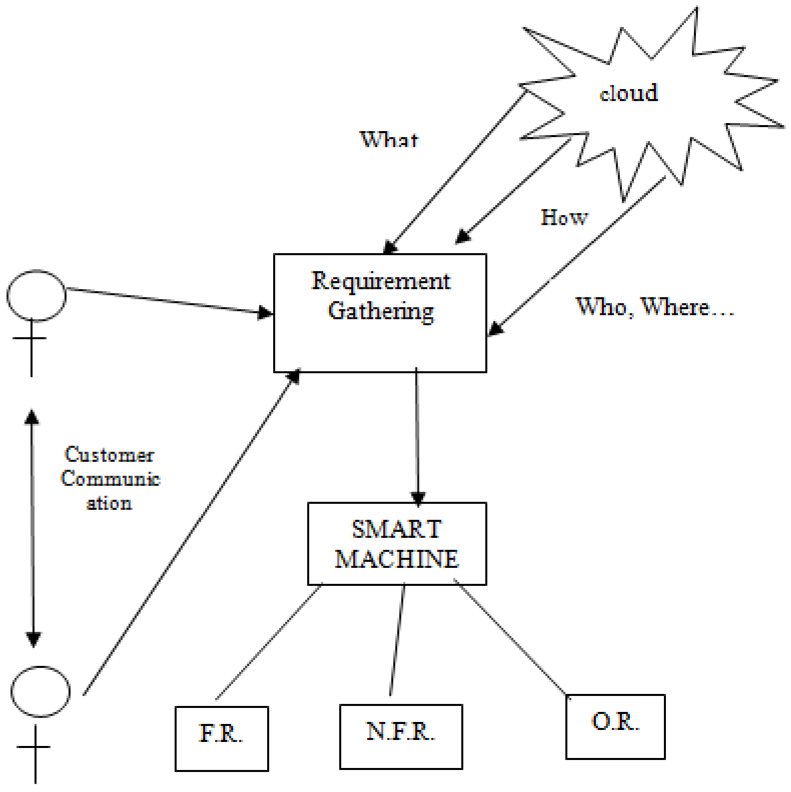

Fig 1 Expert System

Figure 1 shows that when stakeholders completed their communication we get the bulk of requirements. We scan all requirements one by one. The input of the expert system is scan requirements then the smart machine uses its artificial knowledge given by us and derive some conclusions. We can provide knowledge to the expert system in different ways. The Inheritable knowledge is as follows: Requirements bunch consist of multiple requirements, multiple requirements can be Functional, Non functional and other requirements. WHAT is Functional requirements, How is Non functional requirements and WHO,WHERE,WHEN, WHICH etc are other requirements. Business requirements specification and Business rules represent WHAT but Performance, Response,Time,Throughput,Scalability,Capacity,Availability,Reliability,Recoverability,Maintainability,Serviceability, Security, Regulatory, Manageability, Environmental, Dataintegrity, Usability, Interoperatibility represent HOW.

The expert system is a smart machine and has the following features like- it should requires knowledge database and with the help of knowledge database the expert system is able to take decisions. There are various methods are used to develop knowledge database such as Inheritable, Relational and Procedural. The Inheritable knowledge for expert system is as follows-

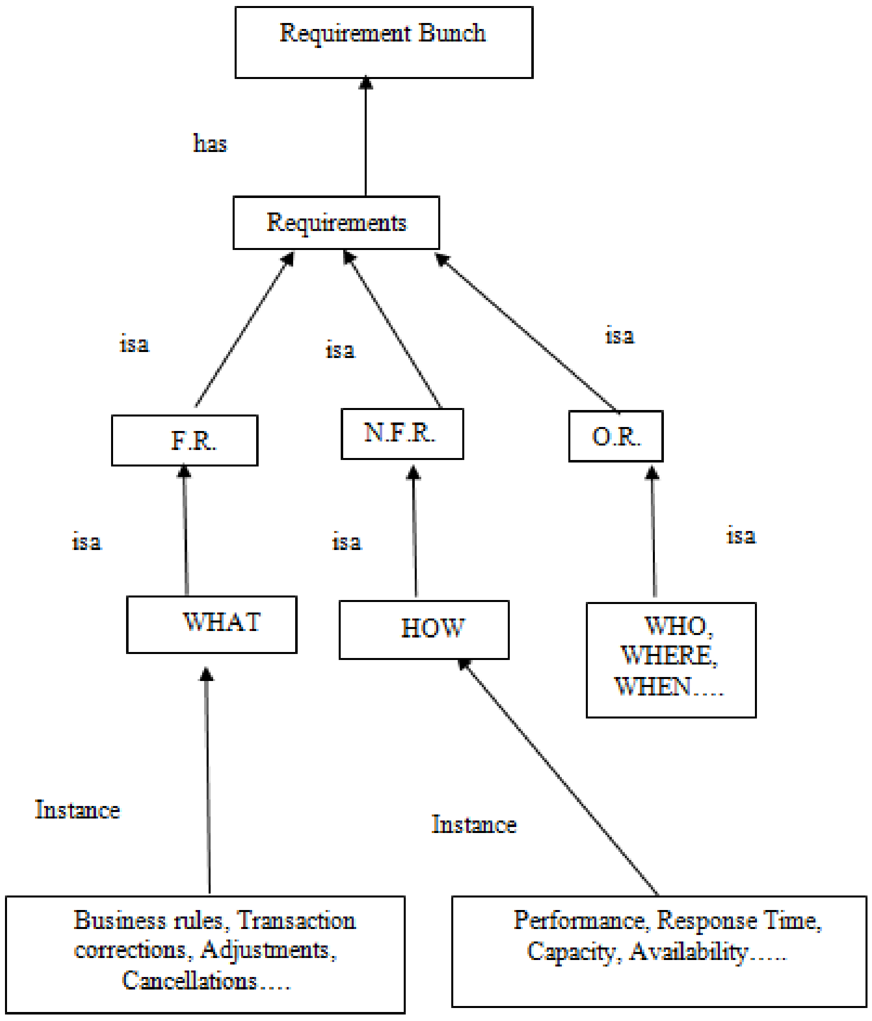
Requirement Bunch
has
Requirements
isa
isa
isa
F.R.
N.F.R.
O.R.
isa
isa
isa
WHAT
HOW
WHO, WHERE, WHEN….
Instance
Instance
Business rules, Transaction corrections, Adjustments, Cancellations….
Performance, Response Time, Capacity, Availability…..

Fig. 2.Inheritable Knowledge

Figure 2 shows the Inheritable knowledge. It shows that Requirement bunch has multiple requirements, FR, NFR and OR are type of requirements. WHAT is Functional Requirement, HOW is Non functional Requirement and WHO,WHERE,WHEN,WHICH are Other Requirements. WHAT instances are Business requirement documents, Business rules, Transaction corrections, adjustments and cancellations, Administrative functions, Authentication, Authorization levels Audit Tracking, External Interfaces, Certification Requirements, Reporting Requirements, Historical Data Legal and Regulatory Requirements. HOW instances are Performance,Response,Time,Throughput,Scalability,Capacity,Availability,Reliability,Recoverability,Maintainability,Serviceability,Security,Regulatory,Manageability,Environmental,Data integrity, Usability and Interoperability.The Relational Knowledge for the Expert system is as follows—

Table 2 Relational Knowledge

Player	Task/Activity	Output
Project Manager and Stakeholder	Communication	Set of Requirements
Project Manager and Stakeholder	Uses Fact finding tools	Set of Requirements
Project Manager	Uses the Expert System	Different types of Requirements
Project Manager	Checks the Different types of Requirements	Allocate each requirement to its specific category.

Table 2 shows the Relational knowledge. This is the simplest way to represent declarative facts is as a set of relations of the same sort used in database systems. The reason that this representation is simple is that standing alone it provides very weak inferential capabilities but knowledge represented in this form may serve as the input to more powerful inference engines. The important thing about Relational knowledge is that it always represent in tabular form. This means that it consist of rows and columns. We always read relational knowledge row wise

The procedural knowledge as Rules for the Expert System is as follows--

```
If:     Customer communication, and
        Uses the fact finding tools, and
        Uses the Expert system, and
        Get different types of requirements,
Then:   Allocate requirements to its specific category.
```

Fig 3 Procedural knowledge as Rules

Figure 3 shows the procedural knowledge. The most commonly used technique for representing procedural knowledge is the use of production rules. In this method we write rules by using two keywords IF and THEN. With IF we can write multiple conditions with AND operator and with THEN we can write only response.

When the communication between project manager and stakeholder is complete we get the set of requirements, after that then project manager uses the expert system and find the different types of requirements such as Functional, Non functional and Other Requirements.

IV. CONCLUSION

There are basically two conclusions first is The proposed model avoids or minimize the manual work because 80% of the work (Requirement segregation) is done by smart machine. The advantage of this model is it can handle millions of requirements at a time also increase the speed and efficiency. The advantage of Requirement Segregation CASE tool is, it is not limited to any software but this is the generalized tool. We can embedded this tool to any Information Technology project. The second conclusion is that there is not a single method to represent knowledge, but there are various methods are available. We can select any one from our convenience.

ACKNOWLEDGMENT

Our most cordial thanks are due to the SICSR for the use of its facilities which made this study possible. I thanks Dr. Yogesh Patil for their inspirational quotes and valuable suggestion. I thanks Lalit Kathapalia and Harshad Gune for their support.

REFERENCES

[1] McGraw-Hill International Edition (2010). Software Engineering :A Practitioner's Approach(7th ed.), Roger S. Pressman.

[2] Fouss, J. D. (1996).Computer-aided Software Engineering in a Computer Supported Cooperative Work Environment(Report). Auburn, USA.

[3] Jongmoon Baik (2000).The effects of Case tools on software development effort. Dissertation for Faculty of Graduate School, University of South California.

[4] Andoh-Baidoo, Francis Kofi, K. Niki Kunene, and Ross E. Walker. An Evaluation of CASE Tools as Pedagogical Aids in software development courses.

[5] Helming, Jonas, Maximilian Koegel, Hoda Naguib, Miriam Schmidberger, Florian Schneider, & Bernd Brugge. (2010). An analysis of tool-based research in software engineering. In Computer Software and Applications Conference (COMPSAC), IEEE 34th Annual, (pp. 53-61). Munich, Germany.

[6] Greenhorst, Danny, Maat, Matthijs, & Maijers, Rob. (1998). Evaluating OO-CASE tools: OO research meets practice. In Object-Oriented Technology: ECOOP'98 Workshop Reader,(pp. 486-488). Springer Berlin Heidelberg.

[7] Offen, R. (2000). CASE Tools and Constraints. North Ryde: Macquarie University Joint Research Centre for Advanced Systems Engineering. Sydney, Australia.

[8] Kesarwani, P., & Misra, A. K., et al. (2013). Selecting Integrated Approach for Knowledge Representation by Comparative Study of Knowledge Representation Schemes. International Journal of Scientific and Research Publications, Volume 3, Issue 2, ISSN 2250-3153.

[9] Tanwar, Poonam., Prasad, T., Datta, Kamlesh., et al. (2012). Hybrid Technique for Knowledge Representation & Comparative Study. International Journal of Computer Science & Engineering Survey (IJCSES), Vol.3, No.4.

[10] Gupta, Sita., Mohta, Deepak Kumar., Todwal, Vinod., Singh, Sushma., et al. (2011). Knowledge Representation. International Journal of Soft Computing and Engineering (IJSCE),Volume-1,Issue-NCAI2011, ISSN:2231-2307.

[11] Rajeswari, P., Prasad, T. (2012). Hybrid Systems for Knowledge Representation in Artificial Intelligence. International Journal of Advanced Research in Artificial Intelligence,(IJARAI), Vol. 1, No. 8.