GRAT: GROUPWARE SUPPORTED REQUIREMENTS ANALYSIS TOOL

VELESWARAN A/L NALLAIAH

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

MARCH 2004

GRAT: GROUPWARE SUPPORTED REQUIREMENTS ANALYSIS TOOL

VELESWARAN A/L NALLAIAH

DISSERTATION SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SOFTWARE ENGINEERING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY, UNIVERSITY OF MALAYA

MARCH 2004

Abstract

This research provides an environment for requirements analysis process to be conducted utilizing the Internet without jeopardizing on the collaborative involvement of the stakeholders. The result of the research was a web-based groupware supported requirements analysis tool, named Groupware Supported Requirements Analysis Tool or GRAT for short. GRAT supports the requirements analysis stage in a typical Software Development LifeCycle (SDLC) to review on what the proposed system will do and how it will fit into the target environment. The major activity of this stage is to review and confirm the documents that define the system, called the Functional Requirement Document or FRD. Being web-based, the constraint on time and space is very much reduced if not eliminated. In pursuit to reaching the objective, literature review was carried out focusing on requirements analysis methods and the tools that support the respective methodologies. Along the way, a very modular requirements analysis methodology was chosen to be used as the framework for the requirements analysis tool. This requirements analysis methodology was introduced by Ian Sommerville and has been the *de facto* standard in the commercial software development industry. The analysis and the design for GRAT were based on the object-oriented paradigm with the use of use-case diagrams, class diagrams and interaction diagrams. The implementation was carried out using Lotus Notes/Domino Release 5 as the back end and Microsoft Internet Explorer version 5 and above for the front end. The system was tested by two different groups of students who use the tool to cater for their class project and assignment. The result of the testing was captured through a number of questionnaires needed to be answered by the participants. The result were analyzed in order to understand the useful features of GRAT and to identify the areas for GRAT is to be improved or further developed in future.

Contents

List of Figures

List of Tables

CHAPTER 1: Introduction

1.1 Requirements Analysis

Software engineering, as defined by Ian Sommerville, is an engineering discipline where software engineers use methods and theories from computer science and apply this cost-effectively to solve difficult process (Sommerville, 1996).

Requirements play a vital role in the software development process. Before making a costly decision of "what to build", it is important that the requirements are understood completely (Sodhi, 1992). This process evolves from an initial statement of requirements for software engineering product to be completed. There is always a need to engineer system software that meets user requirements and expectations within available resources and to accommodate changes throughout the software life cycle.

1.2 Requirements Analysis Definitions

Institute of Electrical and Electronics Engineers (IEEE) Standard 610 (1990), defines software requirements as

- A condition or capacity need by a user to solve a problem or achieve an objective
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents
- A documented representation of a condition or capability stated earlier.

According to Kramer(1988), requirements analysis is the most important step in system development. It is consider as the most critical task in software engineering. Another general definition of requirements and requirements analysis as defined by Leite(1987), requirements analysis is a process of elicitation and modeling of "what is to be done". This process has to deal with different viewpoints, and it uses a combination of methods, tools, and actors.

In a typical Software Development Life Cycle (SDLC), the requirements analysis process takes place after initial feasibility studies. It is a major stage in requirements engineering. Sridar (1994) has divided requirements engineering into four specific processes:

- Requirements Elicitation

  "*The processes through which the customers, buyers or users of a software system discover, reveal, articulate and understand their requirements.*" (Sridar, 1994)

- Requirements Analysis

  "*The process of reasoning about the requirements that have been elicited; it involves activities such as examining requirements for conflicts or inconsistencies, combining related requirements and identifying missing requirements.*" (Sridar, 1994)

- Requirements Specification

  "*The process of recording the requirements in one or more form, including natural language and formal, symbolic or graphical representations; also, the product that is document produced by that process.*" (Sridar, 1994)

- Requirements Validation

  "*The process of confirming with the customer or user of the software that the specified requirements are valid, correct and complete.*" (Sridar, 1994)

Sommerville(1996) had defined requirements analysis as a process of deriving the system requirements through observations of existing systems, discussions with potential users and/or task analysis. He highlighted the important aspects of requirements analysis, which are:

- Requirements Analysis Sizes the Problem

  Requirements analysis enables the software or system engineers to specify software functions and performance. It also assists them to identify interfaces between software and other system elements. Moreover, it helps to establish software design constraints

- Requirements Analysis Is a Process of Discovery

  In addition to the previous mentioned characteristics, requirements analysis helps in recognizing problems, evaluating and synthesizing solution.

- Requirements Analysis Is a Process of Refinement

  In requirements analysis process, the software scope is refined in detail in which alternative solutions are analyzed and allocated to software elements. A complete specification of software requirements is developed, as it is essential to a project's success

Generally, all software requirements analysis irrespective of methods or architecture practiced encompasses of four areas, which are (1) problem recognition, (2) evaluation and synthesis, (3) specification and (4) review.

After running through the different definitions and explanations of requirements analysis by various experts in the field, one cannot help but to conclude that if the requirements are done well, the software design flows logically and smoothly. Conversely, if the requirements are done poorly, the resulting design is awkward and the coding is more difficult. Usually, errors identified in the requirements stage are the fastest and least expensive to correct, while those found in later stages are increasingly more time-consuming and expensive to correct.

Consequently, requirements analysis is a process of developing clear, complete, agreed upon and feasible requirements for a product generally. The acceptability of the system after it has been delivered depends on how well it meets the customer needs and supports the business process of the client's organization. If the analyst does not discover the customer's real requirements, the delivered system is unlikely to meet the expectations.

1.3 Requirements Analysis Objective

The tangible result of a requirements analysis is a set of requirements that can be used by the software development team. Therefore, the objective of requirements analysis is to produce a well-defined set of requirements through involvement of various parties or participants in this process.

1.4 Requirements Analysis Benefits

It is a norm for clients or users of a software system that is to be developed to have a vague idea of what they really need and with little idea of what software technology might offer. This leads the developers to many assumptions. A good analysis process helps them to explore and fully understand their requirements.

Looking into the economic and financial importance of the requirements analysis, Boehm and Pappacio (1988) pointed out that high number of faults attributed at the design stage could have derived from requirements error. Earlier, Basili and Periccone (1984) reported that 48% of the faults observed in a medium scale software project could be avoided if not for the incorrect or misinterpreted functional specifications or requirements. Beizer (1990) in his publication cited that slightly over 8% of the faults in his samples could have been minimized if not for the incorrect, illogical, unreasonable, ambiguous, over specified, unverifiable, poorly presented and changed requirements. He also added that it is not unusual for a faulty requirement to get through all development testing, beta testing and initial field use, only to be caught after hundred of sites have been installed

Thus, by active participation of both developer and end users, the requirements analysis process allows them to have a good understanding of the implications of the decisions they have made in developing the requirements. This results in fewer surprises when the system is built and delivered. At the end of the day, participants of a good requirements analysis process feels a sense of ownership of the product, satisfied with the process, feel informed, educated and believe their risk is minimize therefore commitment towards the success of the project is relatively higher (Sridhar, March 1994).

1.5 Requirements Analysis Difficulties

The requirements phase is not considered an easy process. There are several reasons for this. Among them are requirements volatility, requirements elicitation problems, language problems and requirements traceability problems. Sommerville (1996) have further elaborated the problems. According to him,

- Stakeholders often do not really know what they want from the computer system except in the most general terms. Even if they have a clear idea of what they want the system to do, they may find this difficult to articulate. They make unrealistic demands because they are unaware of the costs of their requests.

- Stakeholders in a system naturally express requirements in their own terms with implicit knowledge of their own work. Engineers, without much experience in the customer's domain, must understand these requirements and translate them to an agreed form.

- Different stakeholders have different requirements and they may express these in quite different ways. Engineers must discover all potential sources of requirements and discover commonalities and conflict

- Analysis takes place in an organizational context. Political factors may influence the requirements of the system. These factors may not be obvious to the system end-users. They may come from higher management influencing the system procurement in ways that satisfy their personal agenda.

- The economic and business environment in which the analysis takes place is dynamic. It inevitably changes during the analysis process. Hence, the importance of particular requirements may change. New requirements may emerge from new stakeholders from new stakeholders who were not originally consulted.

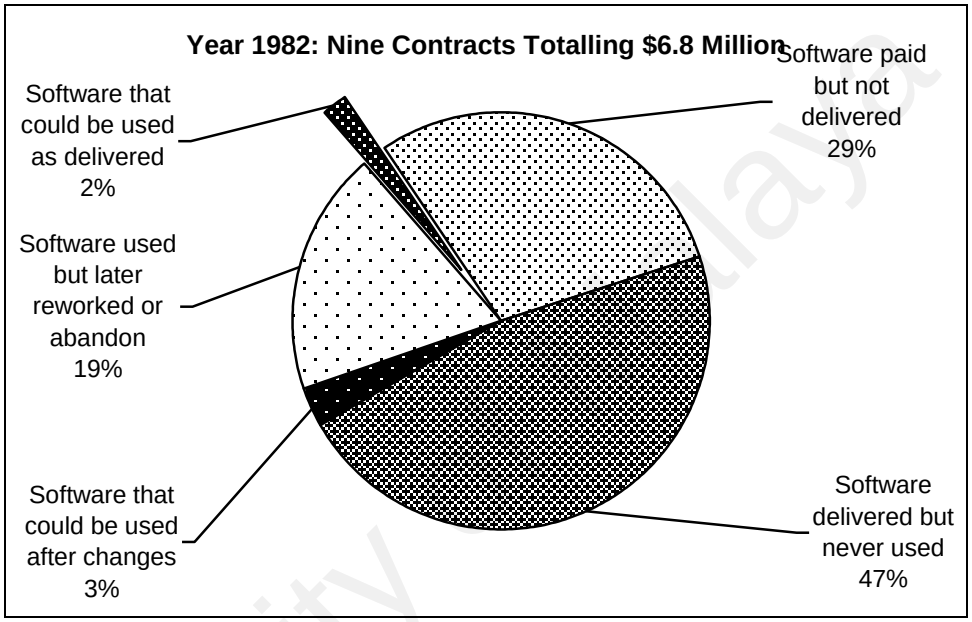


Figure 1.1: Results of GAO survey of software contracts (Sridhar, March 1994)

Figure 1.1 indicates the importance of understanding the user's requirements. Generally, almost none of the software purchased under these contracts could satisfy the user's need.

A survey of 23 development organizations showed that projects in many applications have similar problems with requirements (Lubars, 1993). Among the problems are,

- Organizational solutions are more pertinent than technology solutions.

- Organizations under invest in support and education so that analysts cannot effectively us the tools they have.

- Many requirements are invented during product design. There is no definite source of customer requirements.

Table 1.1: Possible causes of system failure (Lyytinen, 1987)

		Possible Cause				
		Lack of Systematic Process	Poor communication between people	Lack of appropriate knowledge and understanding	Inappropriate, incomplete or inaccurate documentation	Poor management of people and resources
Type of Failure	Process	✓	✓			✓
	Interaction		✓	✓	✓	
	Expectation		✓	✓	✓	

It is important for the requirements analyst to be aware of the possible causes of failure and must use techniques that help to avoid failure. Table 1.1 briefly outlines the link between possible causes with types of failure.

1.6 Easing Requirements Analysis Difficulties

Davis (1993) concludes that a perfect software requirements analysis does not exist and a perfect requirements analysis produces a document very large that it loses its conciseness. However, the requirements analysis methods must be able to generate, at

least, an acceptable set of requirements. Otherwise, developers do not know what to construct, customers do not know what to expect and there is no means to validate the system (Hsia, 1993).

Requirements analysis methods should be independent and provide structuring facilities to obtain more modifiable, traceable, annotated and organized requirements. Additionally, they should have a notation that favors the communication within the stakeholders.

According to Vanwelkenhuysen (1996), a quality or highly effective requirements analysis group decision-making must be implemented. Essential to group decision-making is that each participant has beliefs about (1) why a group decision is relevant to their concerns (2) how their actions influence the rest of the group and (3) the extent her concerns are addressed by the decisions.

In summary, requirements analysis is a total teamwork process. It requires active involvement of the stakeholders. In order to support their active involvement, groupware application or defined as a platform that provides the tools to support team activities is needed and the science that concentrates on building such tools is called Computer Supported Collaborative Work or CSCW for short.

1.7 Groupware

In a competitive environment that is global, intense and dynamic, the development of new products and processes is increasingly becoming a focal point of competition (Clark, 1993). Organizations earn more profit by getting to the market faster and more efficiently with products that are well matched to the needs and expectations of the end user. In order to cope with this competitive environment, many organizations are attempting to transform their structures and processes through teamwork, global integration and networking (Ciborra, 1993; Orlikowski et al., 1995).

This is where groupware comes into action. Groupware is a technology designed to facilitate the work of groups. The technology is used for communicate, cooperate, coordinate, solve problems, compete and/or negotiate in attending or completing a task. While traditional technologies like the telephone qualify as groupware, the term is ordinarily used to refer to a specific class of technologies relying on modern computer networks, such as email, newsgroups, videophones, or chat (Brinck, 1998).

CSCW refers to the field of study, which examines the design, adoption, and use of groupware. Groupware applications are computer-based system that supports teamwork in a common task or goal thus providing an interface to a shared environment (Ellis, 1991). Despite the name, this field of study is not restricted to issues of "cooperation" or "work" but also examines competition, socialization, and play. The field typically attracts those interested in software design, social and organizational behavior, including business people, computer scientists, organizational psychologists, communications researchers, and anthropologists, among other specialties.

One might ask how groupware design is different from traditional design. Groupware design involves the understanding of teamwork and how people behave in groups. It involves having a good understanding of networking technology and how aspects of that technology, for instance, delays in synchronizing views, affect a user's experience.

Many aspects of groups require special consideration. In other words, not only do million-person groups behave differently from 5-person groups, but also the performance parameters of the technologies to support different groups vary. Ease-of-use must be better for groupware than for single-user systems because the pace of a conversation often drives the pace of use of an application. System responsiveness and reliability become issues that are more significant. Designers must have an

understanding of the degree of homogeneity of users, of the possible roles people play in cooperative work and of who key decision-makers are and what influences them.

1.8 Research Motivation

Requirements analysis is a process involving stakeholders, from developer to end-user. A common platform that supports the activities is required to execute the processes. In other words, stakeholders involved in a requirements analysis process need a ground to get them involved, provide ideas, receive feedbacks, share opinions, review and make decisions. Therefore, a research on groupware supported requirements analysis tool was carried out. Being web-based, it fulfills the tools characteristic of being in a common ground and groupware supported to fulfill tasks on working together.

1.9 Research Objective

This research provided an environment for conducting requirements analysis process utilizing the web technology and platform. The objective of the project is to build a web-based requirements analysis tool that supports teamwork activities. The system enables requirements analysis process activities conducted via their terminals through the Internet. With these characteristics, time and space constraint is reduced if not avoided. The two main aims of this research are:

1- To build a Groupware Supported Requirements Analysis Tool that distributes the requirements analysis process among the team members. In the same time, it should also provide flexibility on time and place. This tool provides

- A repository for projects as this would help to keep track of the projects

- Requirements traceability to see the owner of the requirements and the changes made on the requirements.
- Collaborative activities like e-mailing, sharing ideas and discussions
- Ability to express disagreements or raise issues
- Negotiating to resolve issues
- Traceability of requirements decisions
- Checking the completeness and consistency
- Sharing information
- Supports brainstorming session
- Supports classifications of the requirements

2- To evaluate the implemented system. The system was tested against the above objectives and against its features that was identified from this research.

1.10 Research Importance

This research carries its importance in providing to the needs of the software engineers to support the phases of requirements analysis. It is important for software engineers to interact with other stakeholders, achieving traceability as from the first document presented and managing more efficiently the problem of changing user requirements.

1.11 Research Scope

This research covers the application of web technologies and TCP/IP platform as a communication medium for the entire requirements analysis process, taking into

consideration for facilities for team cooperation and decision making. In addition, the flexibility of time and space is taken into account.

1.12 Methodology

A number of requirements analysis methods or architectures exists and being widely implemented. This research as mentioned earlier is to build a Groupware Supported Requirements Analysis Tool. The system was called as GRAT or Groupware Supported Requirements Analysis Tool. In pursuit of reaching this objective and scope, the following procedures are implemented.

- A literature review is carried out to evaluate current requirements analysis methods that are being implemented. This is done to get a proper understanding on importance of various processes in requirements analysis that varies from one to another. From here, it assists in selecting a proper requirements analysis method to be implemented in the tool. This step is considered the most crucial as it enables to determine positive and negative aspects of requirements analysis. An in depth study on groupware in general and how groupware is implemented in current requirements analysis tool was carried out too. The outcome of this literature review had assisted in successfully developing a platform that supports groupware activities in requirements analysis. Moreover, the technologies that are implemented are also reviewed in terms of capability and reliability.

- From the findings, the GRAT features are identified and the development process is executed. Analysis and design based on the requirements are done and documented.

- Finally, the GRAT system is implemented and tested. The results of the testing is also captured and documented in order to be able to identify useful features and areas to be improved in future.

Figure 1.2 shows the methodology applied in this research.

The limitation of this methodology is only one requirement analysis approach will be selected from the literature review. GRAT will be built to support this approach. The tool might be too rigid to support other approaches in future.

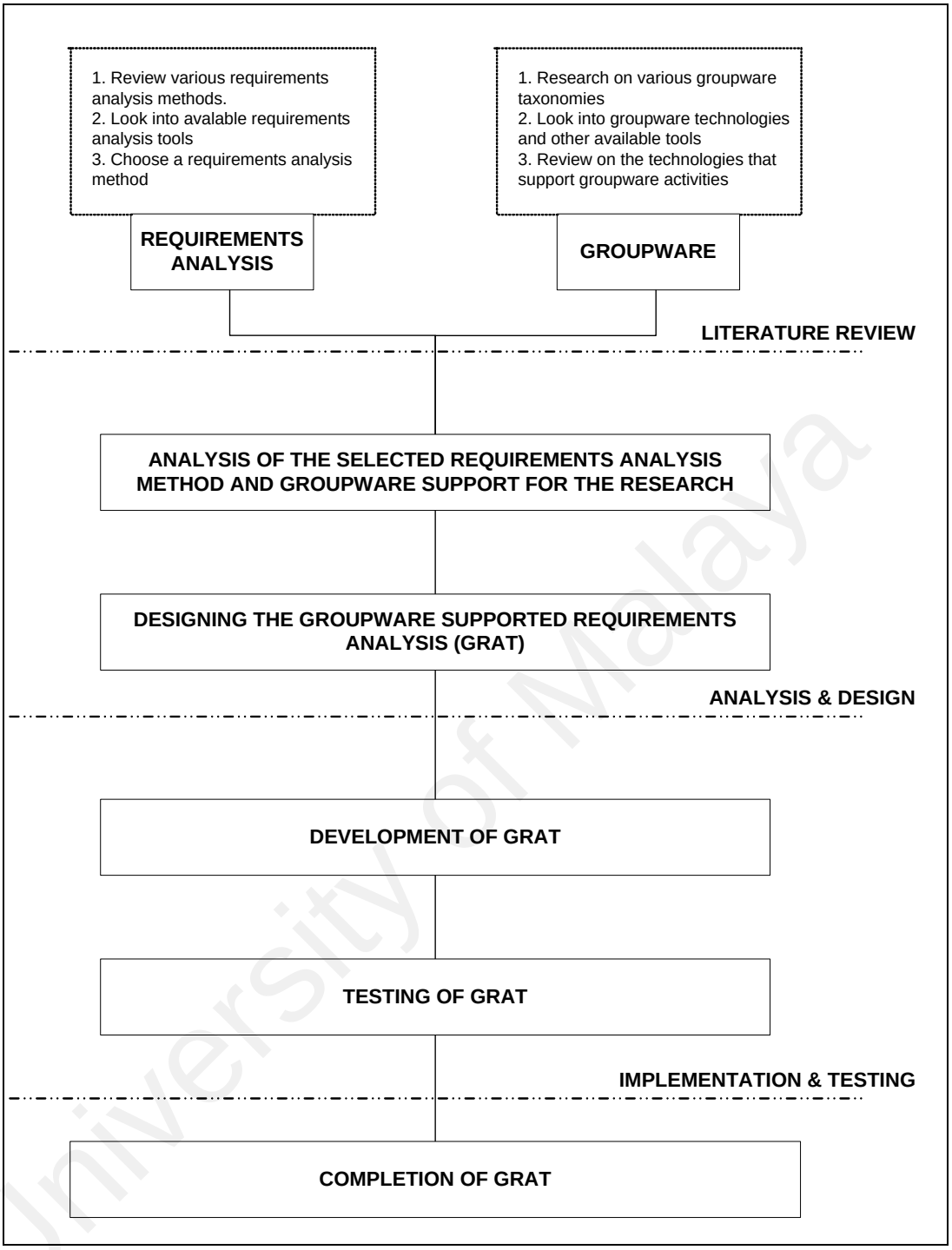


Figure 1.2: Methodology applied for GRAT

1.13 Thesis Organization

Chapter 2: Review On Requirements Analysis Tools And Groupware

This chapter discusses about the various requirements analysis methods and tools that are widely being used. Second part of the chapter looks into groupware and relates its support to some of the tools mentioned earlier. Finally a framework for GRAT is identified.

Chapter 3: Groupware Support For A Requirements Analysis Model

This chapter introduces the requirements analysis model. Looking into groupware strategy and support, a groupware blueprint for GRAT is defined. The groupware activities and users and their roles for GRAT are discussed.

Chapter 4: GRAT Analysis and Design

This chapter describes the analysis and design of GRAT. It includes the requirements analysis of GRAT and object-oriented analysis and design of GRAT. For the sake of simplicity, the detailed analysis and design of GRAT in shown in Appendix A.

Chapter 5: Implementation and Execution

This chapter presents the implementation and how GRAT executes the requirements analysis process. It is divided by the different phases of GRAT. The image capture of GRAT is presented in Appendix B.

Chapter 6: GRAT Evaluation and Results

This chapter looks into the evaluation process and the results recorded from the pilot study.

Chapter 7: Conclusion

This chapter summarizes the content and contribution of the research and the thesis and also identifying some of the aspects for future work.

CHAPTER 2: Review on Requirements Analysis and Groupware

2.1 Introduction

This chapter discusses various requirements analysis methods found in literature. The requirements analysis methods mentioned in this chapter are discussed based on its methodology and the tools that had been developed to support it. In addition, groupware functionalities and technologies are reviewed in order to shape the main features of GRAT. This chapter helped in finalizing the boundaries of GRAT.

2.2 Requirements Analysis

2.2.1 Goal Based Requirements Analysis

Goal based requirements analysis method (GBRAM) stresses the need to characterize, categorize, decompose and structure goals as requirements, but usually fail to offer strategies to identify goals, taking it for granted that the goals have already been documented (Antón, 1994; Dardenne, 1994; Yu, 1994). The method is useful in identifying, elaborating, refining and organizing goals for requirements specification.

There two important process in GBRAM, goal analysis and goal evolution. Goal analysis regards to the thorough studying on the documentation for organizing and classifying goals. Goal evolution meanwhile looks into goal changing from the moment it is first identified right up to the moment they are put in operation or implemented in a system specification. Further elaboration on the process,

- *Goal Analysis*. Goals are identified from process descriptions by searching for statements, which seem to guide design decisions at various levels within a system or organization. They are also sourced from various types of gathered information such as flow charts or Entity Relationship (ER) diagrams. However, process descriptions are insufficient for achieving thoroughness and completeness. This is due to the stakeholders tending to express their requirements in terms of operations and action rather than goals. Searching for action words is a useful way to extract goals from stakeholder descriptions. In addition to the goals, agents, stakeholders and constraints must also be identified. Identify the responsible agents as early as possible by determining what agents are ultimately responsible for the achievement or maintenance of a goal. Constraints are useful because they provide additional information regarding requirements that must be met in order for a given goal to be completed. Identify constraints by searching for temporal connectives, such as during, before and after, or any variants thereof. Figure 2.1 shows an example of an identified goal.

Maintenance Goals	Agent	Stakeholders
G_1: Career tracks provided	AFB	AFB
G_2: Tax payers money spent efficiently	AFB	AFB, DoD, empl
G_3: Training coordinated	AFB	AFB, empl

Figure 2.1: Example of the identified Goal (Anton, 1996)

- *Goal Evolution*. It is a known fact that during the development of software, stakeholders change their minds and refine and operationalize the goals into behavioral requirements. A stakeholder's goals may change or, at a minimum,

their goal priorities are likely to change. Goal evolution is thus affected via goal elaboration and refinement. Useful techniques for goal elaboration are, (1) identifying goal obstacles, (2) analyzing scenarios [8] and constraints and (3) operationalizing goals.

In order to one to anticipate exception cases, goal obstacles need to be identified. Meaning, possible ways for goals to fail need to be considered. Goal refinement occurs when synonymous goals are reconciled, when goals are merged into a sub-goal categorization, when constraints are identified, and when goals are operationalized. Further consideration of goal and agent dependency relations yields deeper insights for conflict resolution. In other words, goals are refined by eliminating redundancies and reconciling with synonymous goals. Goals are also refined via elaboration. The operationalized goals, responsible agents, stakeholders, constraints and scenarios are ultimately consolidated into a set of goal schemas that can be easily translated into a requirements specification. The resulting artifact, while not formal in the strict sense, provides a textual representation of system requirements organized according to system goals. Figure 2.2 illustrates an example of a goal schema.

Goal:	Available course slots announced
Type:	Achievement
Description:	HRD must announce the courses and the number of slots available so that qualified personnel can be identified, matched and notified.
Action:	Announce course slots
Agent:	HRD
Stakeholders:	HRD, employee, TSD
Obstacles:	1. No slots available 2. Employee prefs not available
Scenarios:	1.1 All courses closed (max capacity) 1.2 Courses cancelled (no slots avail)
Preconditions:	1. Employee prefs ready 2. IPs made available
Postconditions:	Employee and course slot matched
Subgoals:	1. Qualifying training slots announced 2. Position training slots announced

Figure 2.2: Example of a goal schema (Antón, 1996)

Goal Based Requirements Analysis Tool or GBRAT supports the goal-based requirement analysis. The platform serves as a medium for project team members from different locations involve in the decision-making processes, which permeate requirements engineering. Ability to collaborative on new ideas, discusses, and decisions about goals despite their location.

GBRAT user is expected to be experienced requirements engineer in the goal-based method and web based applications. However the GBRAT developers have made certain assumption.

> *"We assume that GBRAT users will work from existing diagrams, textual statements of need and/or additional sources of information, such as transcripts of interviews with stakeholders to identify and specify the goals of the desired system. After the analyst/elicitor has gathered all available information about*

the desired system he or she can then extract goals from these information sources and specify them using GBRAT as described below." (Antón, 1996)

Features of GBRAT enable users to create project repositories, create goals, trace goals, view goals from several perspectives and order goals. Among the features are

- **Project Repositories**. Goals concerning a project are kept or stored in a project repository. It is unique based on the specified project name and description as well as the name of the analysts working on a given project as shown in Figure 2.3. In a specific project repository, new goals can be created or previously specified goals can be viewed using three filters, (1) the maintenance and achievement goal filter, (2) the agent filter and (3) the total order filter.

Figure 2.3: Project Repositories (Antón, 1996)

- **Creating Goals.** Users need to submit a form to create a goal, as illustrated in Figure 2.4, to specify the goal name, classification and responsible agents(s). The naming convention for goals is in a standardized subset of natural language. The first word is a verb that describes the kind of goal being named. The goals are classified either as an achievement or as a maintenance goal. Achievement goals are objectives of the system and are named by the verbs "MAKE" and "KNOW" where else, maintenance goals are those goals that are satisfied while their target condition remains true and are therefore named using the verbs "MAINTAIN", "KEEP", "AVOID" and "ENSURE".

Figure 2.4: GBRAT Form to Create Goals (Antón, 1996)

- **Goal Traceability**. The traceability is enabled through hypertext links. In creation of goal, the author must specify the name of the information source from which each goal was identified to ensure that each goal can be traced back to its author, place of origin. In addition, analysts are able to identify goals that have been extracted from more than one information source in order to reconcile any similarities and differences.

- **Viewing Goals**. GBRAT provides users viewing services such that various filters can be used to view goals. The goals are listed alphabetically and displayed in a tabular format, as shown in Figure 2.5. GBRAT allows users to view either achievement or maintenance goals by name. The goals in Figure 2.x are achievement goals.

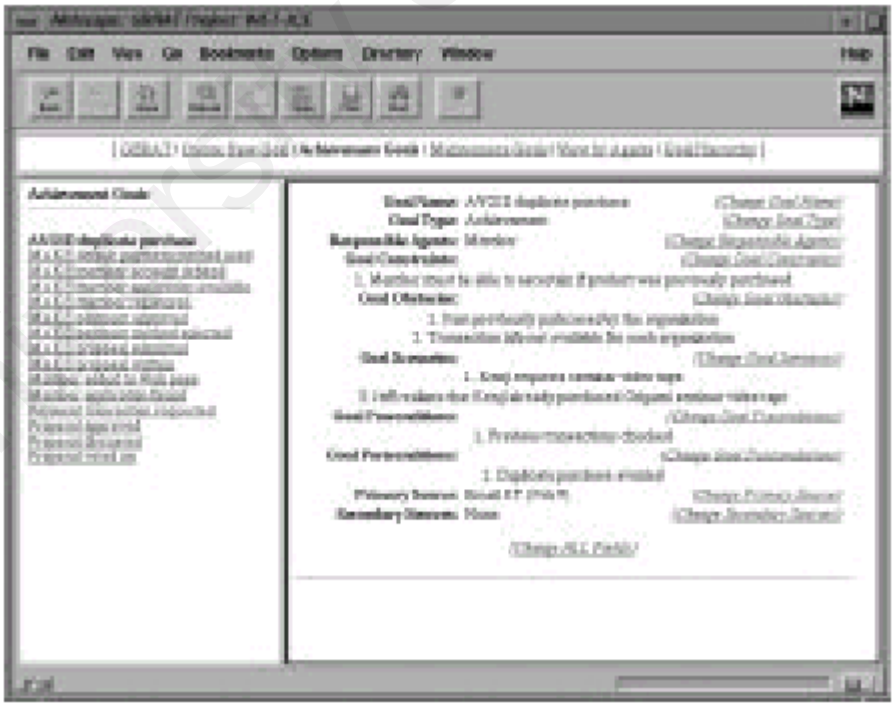


Figure 2.5: Viewing Goals by Name (Antón, 1996)

In summary, GBRAT supports the two stages of the goal-based approach, (1) goal analysis and (2) goal refinement and decomposition. Looking into the facilities provided by GBRAT, it supports the collaborative nature of requirements engineering by implementing Internet technologies. It is done by simply allowing project members regardless of the place they are to work collaboratively. However, Annie I. Antón and her fellow colleagues (1996) of Georgia Institute of Technology highlighted that the use of the Internet introduces problems linked to cooperation, information sharing, different viewpoints and networked enterprises.

2.2.2 Win-Win Requirements

At the last two International Conferences on Software Engineering, three of the six keynote addresses identified negotiation techniques as the most critical success factor in improving the outcome of software projects. Tom DeMarco stated that

> "*How the requirements were negotiated is far more important than how the requirements were specified*." (DeMarco, 1996)

Mark Weiser concluded that

> "*Problems with reaching agreement were more critical to the projects' success than such factors as tools, process maturity, and design methods.*" (Weiser, 1997)

The original spiral model (Boehm, 1988) uses a cyclic approach to develop increasingly detailed elaborations of a software system's definition, culminating in

incremental releases of the system's operational capability. Each cycle involves four main activities:

- Elaborate the system or subsystem's product and process objectives, constraints, and alternatives.
- Evaluate the alternatives with respect to the objectives and constraints.
- Identify and resolve major sources of product and process risk.
- Elaborate the definition of the product and process.
- Plan the next cycle, and update the life-cycle plan, including partition of the system into subsystems to be addressed in parallel cycles. This can include a plan to terminate the project if it is too risky or infeasible.
- Secure the management's commitment to proceed as planned.

After putting into practice, some difficulties came about and to avoid these problems, the spiral model was further reviewed and three activities were to the front of each spiral cycle, as illustrated in Figure 2.6 (Boehm, 1994). The activities are

- Identify the system or subsystem's key stakeholders.
- Identify the stakeholders' win conditions for the system or subsystem.
- Negotiate win-win reconciliation of the stakeholders' win conditions.

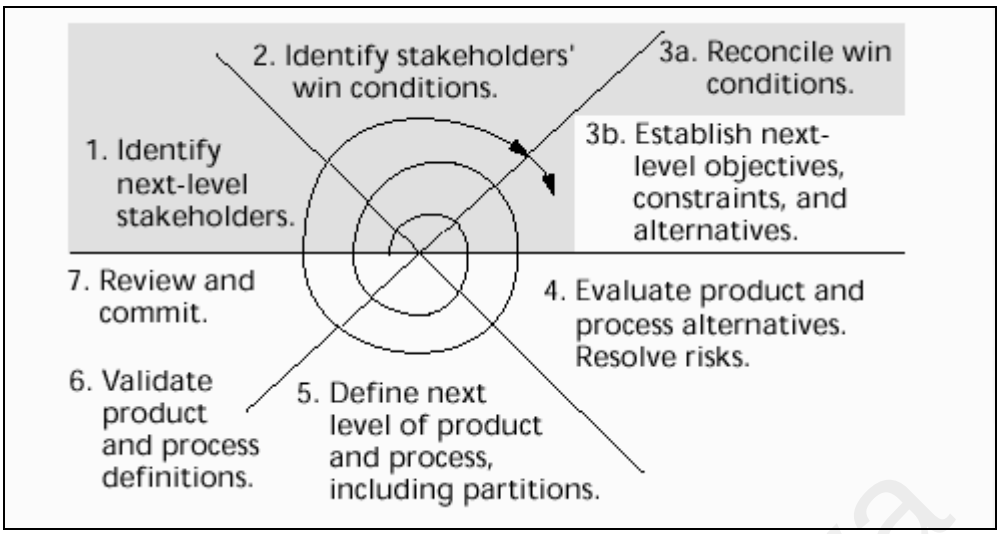


Figure 2.6: Reconciled Win-Win Spiral Model

In **Win-Win requirements** model the decision space that gets collaboratively explored in a WinWin process is modeled using four main conceptual objects:

- Win Condition - capturing the desired objective of the individual.
- Issue - capturing the conflict between win conditions and their associated risks and uncertainties.
- Option - capturing a strategy for resolving an issue.
- Agreement - capturing the agreed upon set of conditions which satisfy stakeholder win conditions and also define the system objectives. The ontology also defines a set of relations between these objects.

Figure 2.7 shows a typical abstract structure of the decision rationale in terms of the above entities and the link types denoting the relations between them. As shown in the figure, an issue (I) is related to one or more win conditions (W x and W y) through the involved relation. An option (O) for resolving an issue (I) is related to the issue through the addresses relation. An agreement (Ag i) based on an option choice (Op) is related to Op through the adopts relation. The agreement (Ag i) has a covers relation with a win condition and a replaces relation to any previous agreement it substitutes.

Figure 2.8 provides a schema-based representation of each of the artifacts. Each object type is defined by a set of tuples, where each tuple consists of the slot name denoting the relation name with a single or multiple cardinality and a typed value field that ranges over an object or enumerated value set. For example, in the win condition object W, the relation comment is an one-to-many function between win conditions and the set of all strings. Similarly the relation defined by the adopted by slot in an option is an one-to-one function between the options and agreements. The state slot of each object is an one-to-many function from the object to an enumerated set of unary predicate constants. For example, a win condition can be in both active and at issue state.

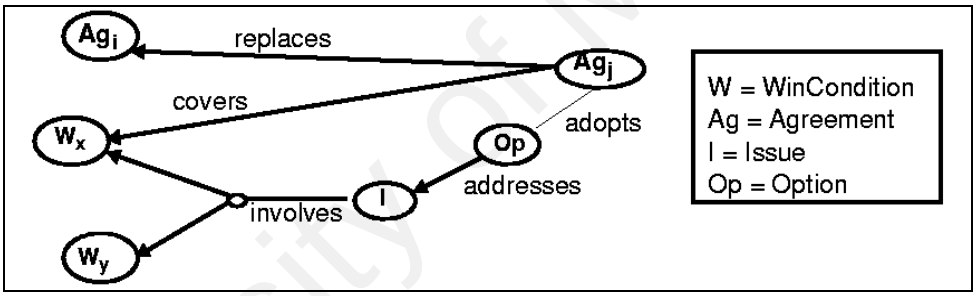
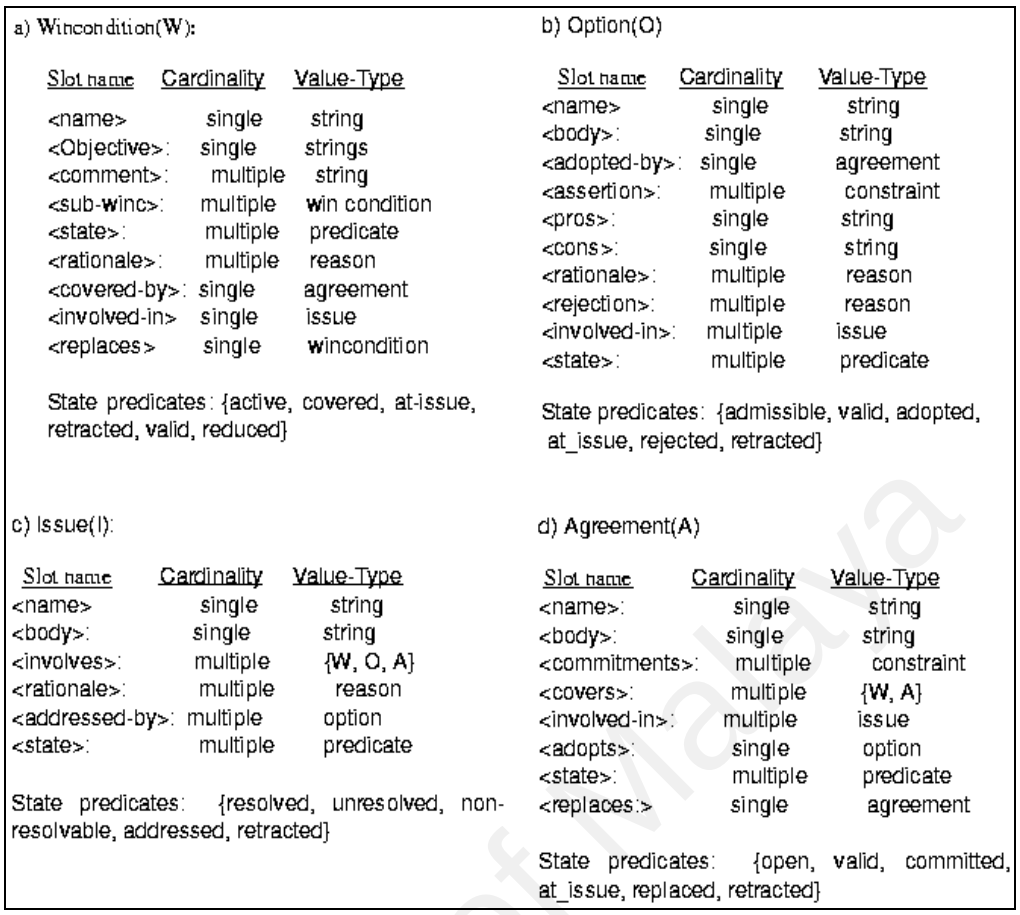


Figure 2.7: Win-Win decision objects and relation between them



a) Wincondition(W):

Slot name	Cardinality	Value-Type
<name>	single	string
<Objective>:	single	strings
<comment>:	multiple	string
<sub-winc>:	multiple	win condition
<state>:	multiple	predicate
<rationale>:	multiple	reason
<covered-by>:	single	agreement
<involved-in>	single	issue
<replaces>	single	wincondition

State predicates: {active, covered, at-issue, retracted, valid, reduced}

b) Option(O)

Slot name	Cardinality	Value-Type
<name>	single	string
<body>:	single	string
<adopted-by>:	single	agreement
<assertion>:	multiple	constraint
<pros>:	single	string
<cons>:	single	string
<rationale>:	multiple	reason
<rejection>:	multiple	reason
<involved-in>:	multiple	issue
<state>:	multiple	predicate

State predicates: {admissible, valid, adopted, at_issue, rejected, retracted}

c) Issue(I):

Slot name	Cardinality	Value-Type
<name>	single	string
<body>:	single	string
<involves>:	multiple	{W, O, A}
<rationale>:	multiple	reason
<addressed-by>:	multiple	option
<state>:	multiple	predicate

State predicates: {resolved, unresolved, non-resolvable, addressed, retracted}

d) Agreement(A)

Slot name	Cardinality	Value-Type
<name>:	single	string
<body>:	single	string
<commitments>:	multiple	constraint
<covers>:	multiple	{W, A}
<involved-in>:	multiple	issue
<adopts>:	single	option
<state>:	multiple	predicate
<replaces:>	single	agreement

State predicates: {open, valid, committed, at_issue, replaced, retracted}

Figure 2.8: A subset of the WinWin ontology for decision rationale

Given the above WinWin framework for collaboration, the elucidated Win-Win requirements model represents stakeholder oriented needs and constraints. Analyzing the Win-Win requirements model for understanding issues and guiding negotiation of tradeoffs requires a solution-oriented model at a level of abstraction relevant to understanding the Win-Win artifact interactions. Figure 2.9 shows an initial conceptualization of the extended decision rationale structure that builds on the viewpoint that the process of analysis is best facilitated by mapping the requirements model to an abstract design model. The extensions in the decision rationale structure involve use of a design model (D) that is motivated by the win conditions (W), considering design factors (Df) arising from D for explaining issues and introducing revisions (del-D and del-W) based on options.

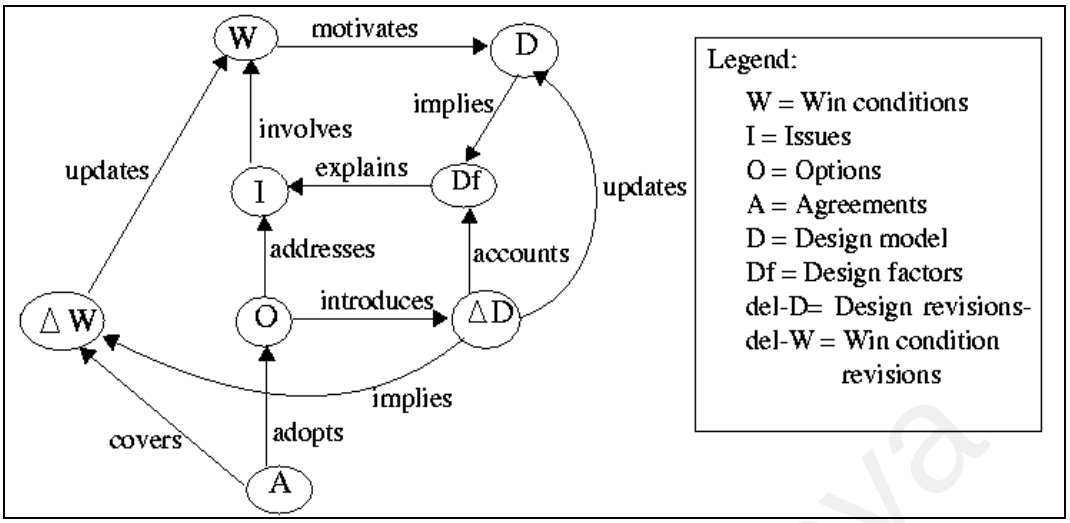


Figure 2.9: An initial conceptualization of the decision structure supporting analysis of Win-Win requirements model.

The above conceptualization of the abstract design model and its relationship to the requirements model can be refined based on a more detailed understanding of the needs that must be met by the design representation. The refinement points would be

- Win conditions expressing functional objectives need to be mapped to functional elements and their task assignment For instance, a representation, explicating the active information processing design elements and their task responsibility then serves as the basis for attributing constraints on design features of the function elements in order to achieve non-functional objectives.

- Win conditions expressing non-functional objectives need to be situated for relevant tasks and function elements and mapped to constraints on those elements - the constraints make explicit behavioral, communication, structural, and other architecture oriented characteristic features of function elements that strengthens quality properties

- Issues expressing win condition conflicts have their basis in design factors - the design factors make explicit the negative ramifications of existing design feature constraints on functional and non-functional needs.

- Options expressing strategies for resolving issues need to be mapped to revision/tradeoff constraints - the revision constraints make explicit the necessary changes required to resolve issues pertaining interfering ramifications of design choices.

The point to note is that the abstraction being used in the above mapping of the WinWin requirements model must be adequate to capture multiple views. Such representational adequacy is required to map and understand different stakeholder win conditions that bear on different aspects of the underlying design model elements and their relationships.

WinWin is a computer program that aids in the capture, negotiation, and coordination of requirements for a large system. It assumes that a group of people, called stakeholders, has signed on with the express purpose of discussing and refining the requirements of their proposed system (Horowitz, 1999). The system can be of any type. WinWin contains facilities for:

- Capturing the desires (win conditions) of the stakeholders
- Organizing the terminology so that stakeholders are using the same terms in the same way
- Expressing disagreements or issues needing resolution
- Offering options as potential solutions

- Negotiating agreements which resolve the issues
- Using third party tools to enlighten or resolve issues
- Producing a requirements document that summarizes the current state of the proposed system
- Creating documents that support multimedia and hyperlinks
- Tracing the ways by which requirements decisions were reached
- Checking the completeness and consistency of requirements

According to the Ellis Horowitz (1999) and his colleagues at University of Southern California (USC), users are advised to follow this simple scenario to begin their work, which has been illustrated in Figure 2.10.

i. Identify the owner of the project. He/she identifies other members.
ii. Owner starts WinWin and registers the project and members, whom are the stakeholders of the system.
iii. Input the defined or tailored existing set of terms of the proposed system into WinWin.
iv. Input the defined or tailored existing taxonomy of the proposed system into WinWin.
v. Review and iterate the terms and taxonomy.
vi. Begin negotiation process, (a) Create Win Conditions and/or (b) create Issues and/or (c) create proposed Agreements.
vii. Review entered artifacts, (a) new Issue is created from new conflict, (b) Options developed to address Issues and (c) new Agreements are created.

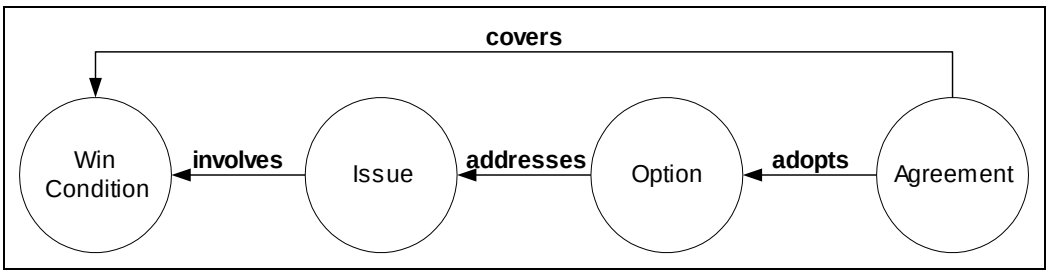


Figure 2.10: WinWin Scenario

Steps 6 and 7 are continued until all Win Conditions covered, all Issues are resolved and all Agreements are passed.

To support auxiliary tools that stakeholders desire to use during the course of negotiation, WinWin provides file attachments facility. For instance, the stakeholders need to use a spreadsheet to analyze the financial impacts of a given Option as shown in Figure 2.11.

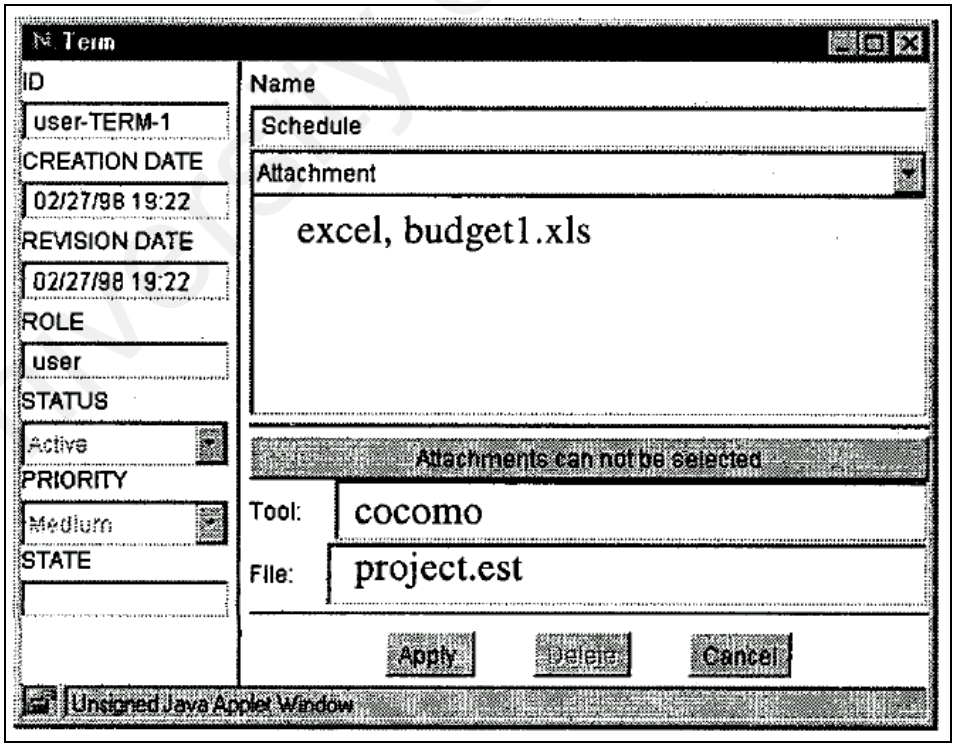


Figure 2.11: File attachment utility in WinWin

Since WinWin assumes that all the stakeholders are at different locations and working at different time, WinWin was made as a distributed and asynchronous mode of operation. In other words, a stakeholder can log into the system regardless of place or time.

2.2.3 Inquiry cycle

Results from several field studies and experiments that were conducted on work practice problems in requirements analysis clearly emphasizes the importance of three activities, namely (1) communication, (2) agreement and (3) traceability management (Curtis, 1988; Kaiya, 1993; Takashashi, 1994.). The problem is these activities are implemented ineffectively and are poorly supported.

Realizing this, Takahashi (1993) and his group of researches developed the **Inquiry Cycle Model**. It is basically a cyclic model of inquiry based requirements analysis. According to them, the model has been applied to several projects successfully. The Inquiry Cycle for requirements analysis consists of 3 activities

> "*(1) Expression is the proposing or preparing of requirements-related Information including not only requirements documents, but also domain specific information, scenarios and enterprise goals. (2) Discussion includes discussing requirements in formal meetings and circulating of comments and individual requirements. (3) Commitment includes making decisions based on the discussions, such as charge requests, agreements about terminology and commitments seeking missing information.*" (Takahashi, 1993)

The 3 activities identified as expression, discussion and commitment is done in loops under control to refine requirements until they become deliverables. Figure 2.12

briefly shows the Inquiry Cycle Model. Take note that the number and nature of the stages are controlled by the strategy. There are three possible views of the Inquiry Cycle to contribute to requirements analysis. It could act as rhetoric for explaining ideas, traceability and as a process model that coordinates and guides participants toward an agreed specification.

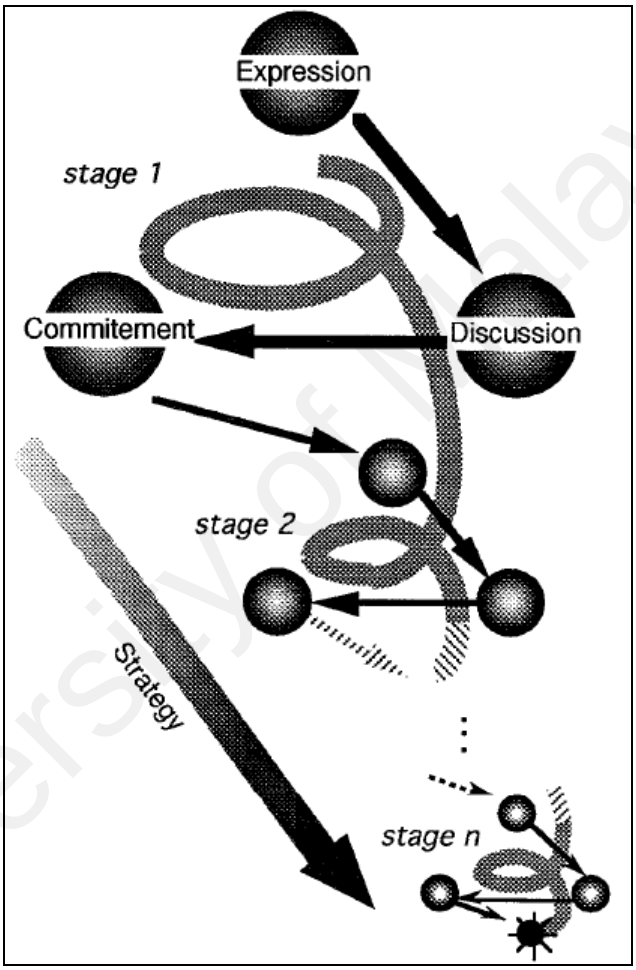


Figure 2.12: Inquiry Cycle Model.

The advantages of Inquiry Cycle being an artifact-based model for collaborative work are the traceable changes and assisting participants share awareness by making the artifact being discussed visible and explicit.

EColabor is an active hypermedia for collaboration requirement analysis. It is designed to address problems in communication, agreement and change management in

requirements analysis. Based on the Inquiry Cycle model, EColabor supports analysts in systematically managing the requirements. EColabor records all the processes of elaborating requirements in shared hypermedia and provides comprehensive support for utilizing these records. According to Kenji Takahashi and his fellow Ecolabor developers,

> "*Ecolabor has client-server architecture that provides participant, even if distributed or working asynchronously, with transport access to the Ecolabor hypermedia information*" (Takahashi, 1996).

Basic features of Ecolabor are,

- Assigning an URL to each element of the hypermedia information stored.
- Extending HTTP to handle check in/out of documents
- Sharing information
- Multicasting audio/video stream
- Shared electronic whiteboard
- Application sharing

Ecolabor provides traceability support through the assist of multimedia. It records an entire session in audio/video and provides support for segmenting the records on the fly. It requires participant to select elements of Ecolabor hypermedia information that they are currently discussing as specifically as possible on their computer displays to set a high level of awareness among other team members. It also supports taking snapshots of sketches and text on the shared whiteboard.

To support coordination, Ecolabor identifies and reports the status of requirements analysis process. It also lets the participant to control the access to the

working version and define a new version. It manages the status of discussions over revisions. In other word, Ecolabor provides status management and version controlling. However, many have opted out due to its complexity in customization. In addition, it requires a decision to be made by one person based on his/her justification without providing flexibility like voting to close a problem or issue.

In summary, Ecolabor manages the pieces of information generated and referred to during requirements analysis. It transmits information with the help of multimedia application. It keeps track of the evolving requirements documents along with the informal information regarding it. Finally, it monitors and navigates the requirements analysis process.

2.2.4 KJ

Repeatedly it has been emphasized that the most important step in system development is to analyze, understand and record the problem that the user, sponsor or client is trying to solve (Takeda, 1992). The functions, goals and constrains on the proposed system must be precisely specified and both the parties must agree on the specification, as they form the basis for the design of the system.

For such problem recognition stage, the **KJ method** is very effective. It was formalized by Kawakita (1982) for generating ideas in his ethno geographical works. The method has been well structured such that it is widely accepted by Japanese business community for usefulness of consensus making among participants of idea generation.

As mentioned earlier, KJ method was developed for new idea generation and claims to establish an orderly system, from chaos. It consists of 4 steps.

The initial stage is to write down on a card what has come to mind on the subject to be discussed. Only one thing must be written down on a card. No judgment

should be made on the importance of what has been written. According to Kawakita, the importance can only be valued after the stage is completed.

Next, is to associate several cards into one group. All the cards must be shuffled and spread on the table or floor. Each card is read several times. The classification or grouping is done subjectively rather than objectively or rationally by examining contents of the card. As quoted by Kawakita, the subject may reveal real desire hidden by the rationality. The grouped card may be labeled in a new card and further grouped in a hierarchical manner.

Following that, the cards are grouped on a large piece of paper and to enclose each group by an outline. Spatial relationships of arranged cards and groups must reflect semantic relationship and extreme prudence is required for this step. It is also necessary to make clear the mutual relationships between the cards and groups by drawing special lines. The relationships include opposition, causality and equality. At this stage, the internal structure of the matters written on the cards that is invisible in the first step becomes visible. The result is called A-type chart.

The final step would be to write down an essay on the subject according to the A-type chart. This step is called B-type writing. It should be noted that the A-type chart represents the subject spatially and that the B-type writing represents the same information in a sequential order. Because of the difference in representation, while doing B-type writing, oversight in the A-type chart may be found and some revision must then be made. However at this point if the cards are pasted on a piece of paper, this is not easy. This fact often makes one hesitant in revision the diagram. Figure 2.13 shows a brief outline of the KJ method while Figure 2.14 depicts the results of KJ Method in requirements analysis.

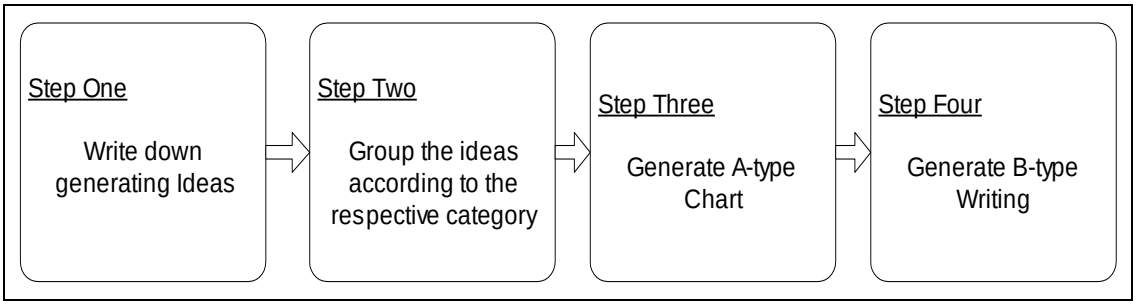


Figure 2.13: Summary of KJ Method.

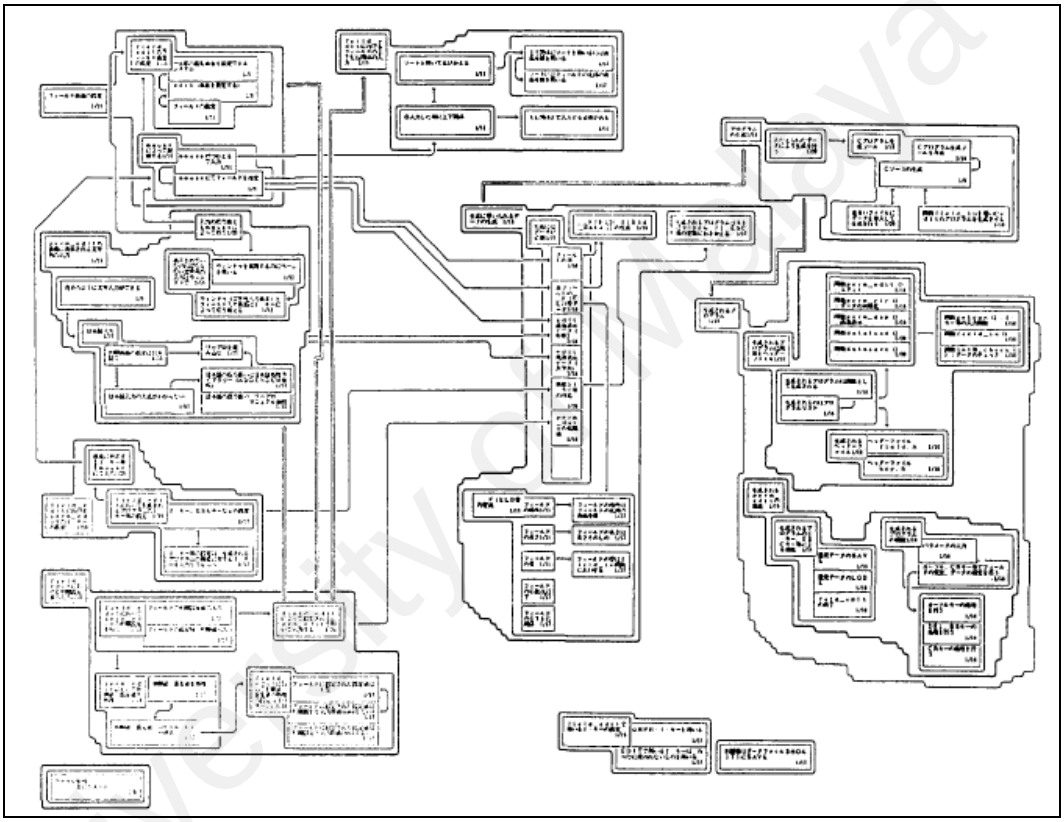


Figure 2.14: Results of Requirements Analysis using KJ Method. (Takeda, 1992)

The system developed to support KJ method was called **KJ Editor**. KJ Editor stimulates index card arrangement on a desk is based on the KJ method that is widely used in the Japanese business community (Takeda, 1992). It is a new tool for getting panoramic view of the whole card arrangement on a computer display.

According to the KJ Editor developers, KJ Editor is a very good tool to record the thinking process of the designer. The thinking process is directly reflected on the

editing action and reviewing the whole process may easily be speeded up, because the record is machine readable and processable.

As the system is developed in Japan, most of the information regarding the functionality of the system is in Japanese. Therefore, there is not much a paper or manuals in English regarding the system that can be referred to. From what the developers have summarized, the KJ Editor is not just limited to KJ method but it is open-ended as it supports the development of flow chart, DFD and state transition diagram. Adding, they also quoted that the Editor ability of recording the thinking process makes the resultant chart produced from KJ editor a very good communication tool between the designer and the client.

2.2.5 Summary of the reviewed methodologies and tools

The above review reveals the characteristics of some of the widely used tools currently available in market. The two tables below summarize the basic characteristics of the tools and the advantages and disadvantages of the tools.

Table 2.1: Comparison of the reviewed tools

Tool	Requirements Analysis Methods	Web-Based	Distributed	Stand Alone
GBRAT	Goal-based Requirements Analysis	Yes	Yes	No
WinWin	Win-Win	No	Yes	No
Ecolabor	Inquiry Cycle	Yes	Yes	No
KJ Editor	KJ	No	No	Yes

Table 2.2: Advantage and disadvantage of the reviewed tools

Tool	Advantage	Disadvantage
GBRAT	Creates repository for projects, which enables the building of knowledge network or future reference within an organization. Traceability supported through hypertext links. Supports collaborative nature by implementing Internet Technologies	The users are required to know about the goals, which are documented elsewhere before using the system. No room for discussion or information sharing.
WinWin	Proper project kick-off with the selection of team members. Capturing the win conditions of the stakeholders Ability to express disagreements or issues Negotiating to resolve issues Producing a requirements document Traceability of requirements decisions Checking the completeness and consistency	Abstraction for Win-Win mapping must be adequate to capture multiple views.
Ecolabor	Sharing information Multicasting audio/video stream Semi-structured email Shared electronic whiteboard Application sharing	Complexity in customization Decision-making is based on one's own justification.
KJ Editor	Supports brainstorming session Supports classifications of the requirements	Stand alone system Does not support collaboration related activities.

2.3 Groupware

Groupware as a collaborative tool provides an easily accessible, widespread platform for gathering and sharing information and for capturing ideas. Groupware is an umbrella term describing the electronic technologies that support person-to-person collaboration. Technologies that support collaboration are in greater demand today than ever before, and, in recognition of that fact, vendors are integrating collaboration technologies into their products. Distributed workforces, information overload, and

getting products to market as quickly as possible are just a few of the motivations pushing collaboration technology development. Additional, as strongly supported by Herd Krasner (1991) and his colleagues, the development of collaborative related technologies depends on the identification of specific human enterprises where the new technologies might prove useful.

2.3.1 Definition of Groupware

Groupware is a relatively new term or concept. Peter and Trudy Johnson-Lenz originally used it in between 1978 to 1980 (Baecker, 1993) to refer to,

> "*Intentional group processes plus software to support them*"

Industry leaders present the following definitions, the most commonly used,

> "*Computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.*" (Ellis, 1991)

> "*Groupware is computer-based technology that (1) actively facilitates two or more users working on a common task, possibly simultaneously, using a shared environment and (2) provides synergistic mechanisms for coordinating each user's actions with respect to the rest of the group and system.*" (Krasner, 1991).

> "*Computer-mediated collaboration that increases the productivity or functionality of person-to-person processes.*" (Coleman, 1996)

"*Computer mediated culture.*" (Baecker, 1993)

As shown in Figure 2.15, groupware lies on a network infrastructure that includes PCs, cabling, network operating systems and administration utilities, or phone lines for a wide area network (WAN). Although groupware is part of the networked applications environment, not all networked applications constitute groupware. For example, access to a corporate database through a network is not necessarily groupware (Coleman, 1997).

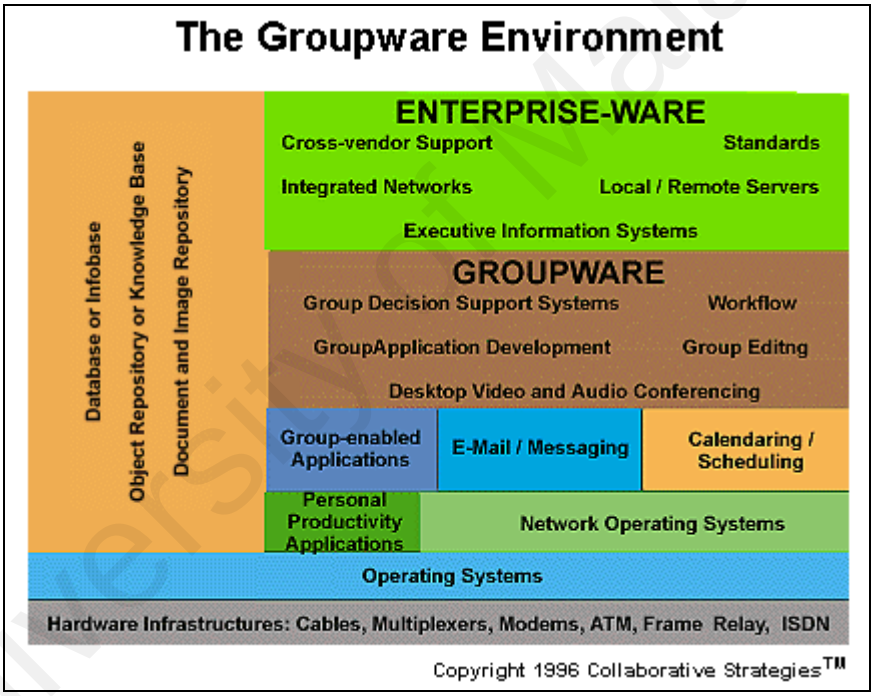


Figure 2.15: Groupware's position in IT architecture (Collaborative Strategies, 1996)

In software process the terms like teamwork, cooperation, coordination and communication that occur within and among groups is synchronous through out the life of software project currently. As mentioned by Alan Kay (cited McLaughlin, 1989) in the final plenary session of the 11th International Conference on Software Engineering, the current examples of automated tools aimed at collaborating groups from the notion of groupware, which is oriented toward the "Tomorrow" column in Table 2.3. The

systems of today as reported by Kenneth L. Kraemer and John Leslie King that support working groups such as automated scheduling applications and electronic mail do not support group as such but rather individuals working in a group (Kramer, 1988).

Table 2.3: Scenario for Software of Yesterday, Today and Tomorrow

Function	Yesterday	Today	Tomorrow
Where	Machine Room	Desktop	Wherever you are
Who	Expert	Individuals	Collaborative Tools
What	Edit	Layout	Orchestrate
How	Remember and Type	See and tell	Ask and tell
Style	Data/procedures	Object Oriented	Rules

To summarize, groupware is computer technology that actively facilitates groups of collaborating users. The goal of groupware is to improve group process, bringing about more effective performance of the group task.

2.3.2 Groupware Parameters

Systems that support groupware activities can be characterized by the way they deal with the set of key parameters. The parameters that were taken into considerations and its definitions are listed below as guided by Krasner (1984).

- Time is the period over which task-related interactions may occur, either synchronous or asynchronous. The occurrences may be for a short or a long period of time.

- Place basically refers to the physical locations of the members of a group, which could be face to face or from one corner to another corner of earth.

- Task meanwhile refers the group's objective and nature of the tasks involved. For instance making decision or voting.

- Human-machine allocation of work refers to the extent to which both the task to be performed and the coordination of the work are performed by humans versus the extent to which they are automated.

Other parameters as directly mention by Herb Krasner and his colleagues are

> "*Group context includes the groups environment, the incentives and the social conventions present in the cultural infrastructure of the working group, including the group's physical surroundings, its adopted style (pecking order), the level of formality in its communication, its willingness to try new things and so forth.*
>
> *Group composition includes the characteristics of the individuals who make up the group and the relationship among those characteristics, for example, the distribution of task-related expertise among group members.*
>
> *Artifact and process focus is the extent to which the groupware focuses on products of a process versus the extent to which it focuses on processes.*" (Kramer, 1991)

Table 2.4 refers to the examples based on the discussion and the conclusion by Johansen (1988) on the role of time and place observing that most groupware supports either one of the four conceptual quadrants according to the needs of the group. He also

asserted that the future groupware to overcome barriers of time and place in order to provide smooth transitions between synchronous and asynchronous work and between collocated and dispersed work.

Table 2.4: Time and Place Dimensions of Groupware Examples.

Dimension	Same time	Different time
Same place	Meeting Room	Hospital nurses' records
Different Place	Teleconferencing	Wide area computer conferencing

2.3.3 Groupware Taxonomy

According to David Coleman (1997), there are twelve functional categories, which form a logical taxonomy. He added 3 new categories from the original, which is a separate category for groupware services, a new category for groupware applications and a special category for the emerging Internet-based collaborative applications and products. The twelve categories are,

- Electronic Mail and Messaging

  Email is by far the most common groupware application. While the basic technology is designed to pass simple messages between 2 people, even relatively basic email systems today typically include interesting features for forwarding messages, filing messages, creating mailing groups, and attaching files with a message. Other features that have been explored include automatic sorting and processing of messages, automatic routing, and structured communication.

- Group Calendaring and Scheduling

  Allows scheduling, project management, and coordination among many people, and may provide support for scheduling equipment as well. Typical features detect when schedules conflict or find meeting times that works for everyone. Group calendars also help to locate people. Typical concerns are privacy, completeness and accuracy.

- Electronic Meeting Systems

  This is a real-time conferencing system as well as collaborative presentation systems. It integrates with calendaring/scheduling systems. It also allows post-meeting follow through the posted information on action items, goals and commitments. Affordability of desktop videoconferencing and Availability of multi-point conferencing make it a suitable groupware application. Addition, they provide tools for brainstorming, critiquing ideas, putting weights and probabilities on events and alternatives, and voting. Such systems enable presumably more rational and even-handed decisions they can encourage equal participation by, for instance, providing anonymity or enforcing turn taking.

- Desktop Video and Real-time Data Conferencing (Synchronous)

  Video communications systems allow two-way or multi-way calling with live video, essentially a telephone system with an additional visual component. Cost and compatibility issues limited early use of video systems to scheduled videoconference meeting rooms. Video is advantageous when visual information is being discussed, but may not provide substantial benefit in most cases where conventional audio telephones are adequate. In addition to

supporting conversations, video may also be used in less direct collaborative situations, such as by providing a view of activities at a remote location.

- Non Real-time Data Conferencing (Asynchronous)

  Contrary to desktop video and real-time data conferencing, this is an asynchronous conferencing. It is mostly like bulletin boards, where one can carry on a conversation over time, leave a message for someone and they answer it, and your can respond back to them later. These messages can be left public or private basis. A clear example would be newsgroups and mailing lists. In practice the main difference between newsgroups and mailing lists is that newsgroups only show messages to a user when they are explicitly requested, sort of an on-demand service, while mailing lists deliver messages as they become available, an interrupt-driven service.

- Group Document Handling

  At a wider definition, group editing, shared screen editing work, group document/image management and document databases falls into this category. A famous feature of group document handling would be hypertext. Hypertext is a system for linking text documents to each other, with the Web being an obvious example. Whenever multiple people author and link documents, the system becomes group work, constantly evolving and responding to others' work. Some hypertext systems include capabilities for seeing who else has visited a certain page or link, or at least seeing how often a link has been followed, thus giving users a basic awareness of what other people are doing in the system.

- Workflow

Workflow systems allow documents to be routed through organizations through a relatively fixed process. A simple example of a workflow application is an expense report in an organization: an employee enters an expense report and submits it, a copy is archived then routed to the employee's manager for approval, the manager receives the document, electronically approves it and sends it on and the expense is registered to the group's account and forwarded to the accounting department for payment. Workflow systems may provide features such as routing, development of forms, and support for differing roles and privileges.

- Collaborative - Internet-based Applications and Products

  Many collaborative functions are moving to the WWW and use the Internet as the input and output while still using traditional groupware on the LAN. It is concentrated on application customization for seamless collaboration on the WWW, data or information storage and balance between security and collaboration. However, there is a very significant limitation of Web applications relative to traditional groupware.

2.3.4 Importance of Groupware

Collaborative Strategies (1996) was commissioned to conduct research in examining how large companies are using IP networks to support electronic collaboration. Their findings were

- 56% felt collaboration was necessary to support a distributed workforce by increasing communication, coordination, facilitation and planning. Sales force

automation and distributed project management were two key functions where collaboration was identified as critical.

- 13% use collaboration strategies to reduce cycle times in product development, thereby realizing an increase in competitive advantage.

- 6% believe that collaboration technologies increase productivity and coordinate or facilitate complex processes

2.3.5 Groupware Design

Those who practice traditional software design have a particular way of thinking about computer. At the core of their mindset is a notion that the computer is a mechanism for manipulating and exchanging data. Meanwhile effective groupware design sees the computer as a medium for people to collaborate (Bock, 1995). Table 2.5 looks at different mindsets, how they differ, and discover how each influences the design of the computer briefly.

Table 2.5: How mindsets influence group design

Design Mindset	What is collaboration	How is groupware designed	What does the design represent
From Understanding	Complex and integrated human interactions	Detached study. Note complexity of interactions.	Purpose, goals, behavior, equilibrium
According to form	Stereotyped work. Tasks that can be labeled and ordered.	Detached study. Focus on form. Identify patterns.	Types of objects, object relationships, object states.
Using rules	Autonomous tasks coordinated by a set of rules.	Detached study. Analyzed according to a model/ theory.	Input, output, control, workflows, functions, rules, procedures.

Within a context	Work that occurs within existing human relationships	Study only-within a given context. Note uniqueness.	Viewpoint, context, experience, action, conversations.

2.3.6 Performance and Acceptance of Groupware Application

One form or other sort of electronic communities were existence for almost as long there has been reliable electronic communication, at least since the mid 1970s (Hiltz, 1984, Hiltz, 1978 and Valee, 1978). Mainly used for proprietary conferencing system. The evolution of Internet and internal network spawned email, listservs and usenet newsgroup to came into action. Their advantage was the universality characteristics compared to the proprietary conferencing systems. Since conferencing systems, email, listservs and newsgroup are generally categorized as different forms of groupware, groupware was defined as a set of hardware and software designed to help groups work together regardless of time and place (Nunamaker, 1991).

The web presents a new opportunity. Rather than relying on proprietary software and architecture, which by their very nature are limiting, new groupware systems and architectures that take advantage of the web, a widely accepted open standard. Due to that, more than 75 groupware systems were made available in 1996, the same year such system made a debut (Woolley, 1996).

Based on the survey conducted on understanding the ways web based groupware was being used by Alan R.Dennis and Bradley C.Wheeler (1997), the major categories are supporting project teams, educational use and special interest group. A further breakdown on the result of the survey is shown in Figure 2.16.

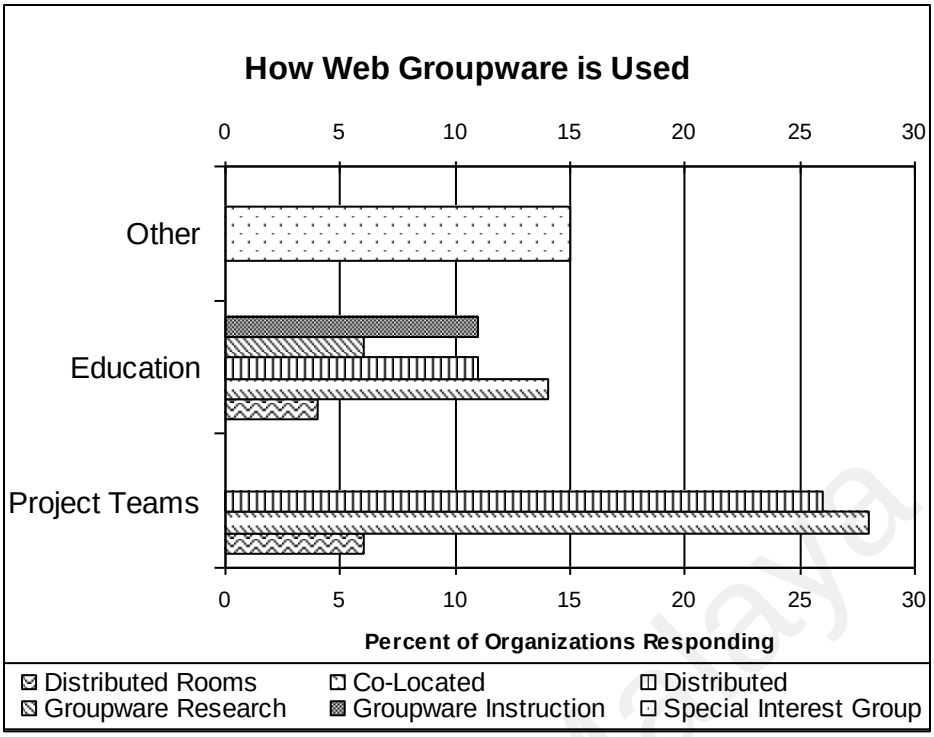


Figure 2.16: Survey on web based groupware

The most mentioned advantage of web groupware was the use of open network standard of Internet and the use of open client standard to enable any-place-any-time interaction. The other advantages are the ability of transformation such as to structure, analyze and sort discussion, specific features, common tools, inexpensive setup costs and minimal learning. The advantages of and its influence are shown in Figure 2.17.

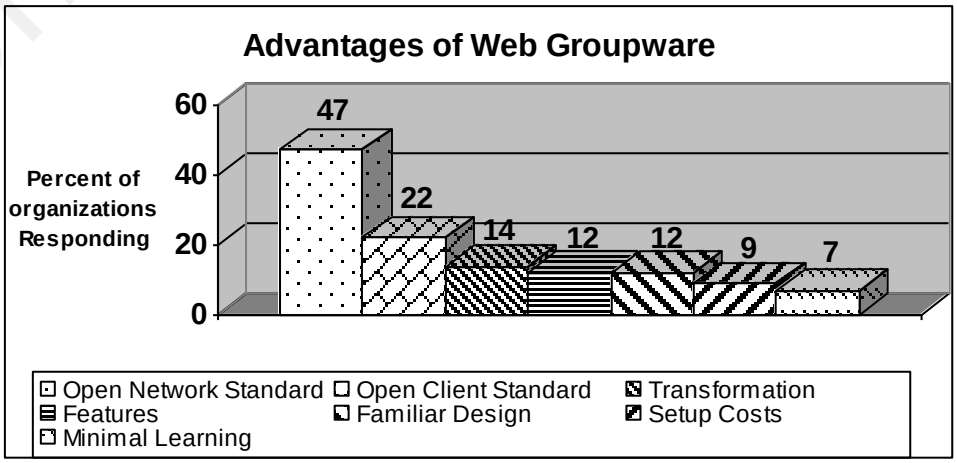


Figure 2.17: Survey results on Groupware Advantages

However, it is a common nature that when there are advantages, disadvantages tends to appear. For this web groupware, the disadvantages can be categorized into three groups. The first and most prominent set of disadvantage dealt with network technologies. The second group is centered on the features of current web group and final group includes operating costs, changing workgroup skills and others. Figure 2.18 depicts the result of the survey conducted on disadvantages of web groupware (Dennis, 1997).

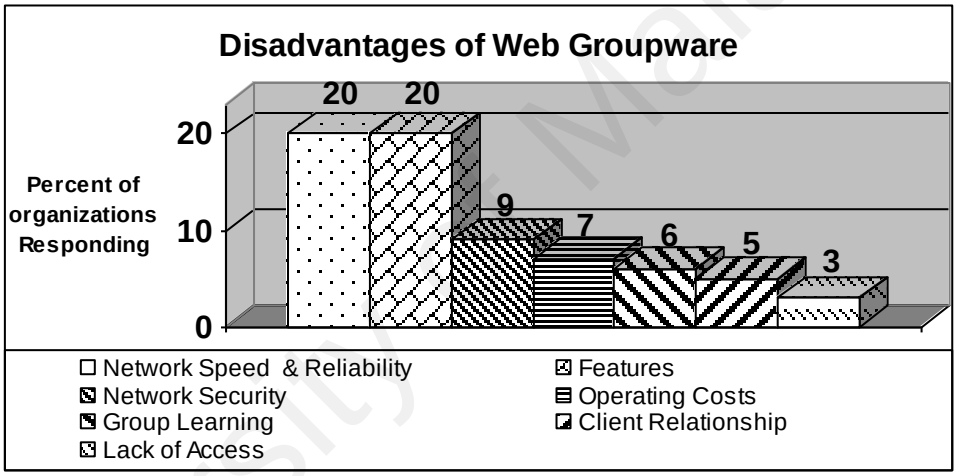


Figure 2.18: Survey results on Groupware Disadvantages

2.3.7 Groupware for Requirements Analysis

Groupware use collaborative technologies to either enhance processes or support collaboration in a specific work environment. With the development of web in terms of architecture and infrastructure has definitely created a boom in groupware applications. The web revolutionizes groupware and the way in which organizations choose to use it. It was expected in 1997 that the web and groupware has transformed from electronic publishing to building and maintaining virtual organizations and enhancing the work of project teams within two years (Dennis, 1997). Overall, groupware can be an important

productivity tool, but the current state of the art is better suited for small team environments rather than large groups. However, active, intelligent, adaptive, orchestrated groupware that manages multiple and possibly concurrent accesses to shared workspaces is the desirable goal for future technology (Krasner, 1991). It is a common believe now that the one who get to the web first, learn its intricacies and push its limits to have a distinct advantage.

Looking at the various requirements analysis frameworks and mentioned in section 2.2, all the requirements analysis methods require some level of teamwork if not at all level. However, the question is to what extend does the tools that supports the requirements analysis methods are supported by groupware technology. The table below looks into the tools and the various groupware technology implemented.

Table 2.6: Comparison of the requirements analysis too against the groupware support

	GBRAT	WinWin	Ecolabor	KJ Editor
Electronic Mail and Messaging	×	×	×	×
Group Calendaring and Scheduling	×	×	✓	×
Electronic Meeting Systems	×	×	✓	×
Desktop Video and Real-time Data Conferencing (Synchronous)	×	×	✓	×
Non Real-time Data Conferencing (Asynchronous)	×	×	×	×
Group Document Handling	✓	✓	✓	×
Workflow	×	✓	✓	×
Collaborative – Internet-based Applications and Products	✓	×	✓	×

Although requirements analysis is a process where teamwork or collaborative exercises are basically in high priority in successful projects, but the tools except for Ecolabor does not seem to be able to perform those task as clearly shown in Table 2.6.

2.4 Research Framework

After reviewing various requirement analysis methods and tools that complements them, in addition to the groupware reviews, a framework for this research has been defined and described in the following sub-sections. The tool to be developed was named as Groupware Supported Requirement Analysis Tool (GRAT)

2.4.1 Requirements Analysis Method

According to Ian Sommerville (1996), requirements analysis or elicitation is the major stage of a generic requirements engineering process as shown in Figure 2.19.

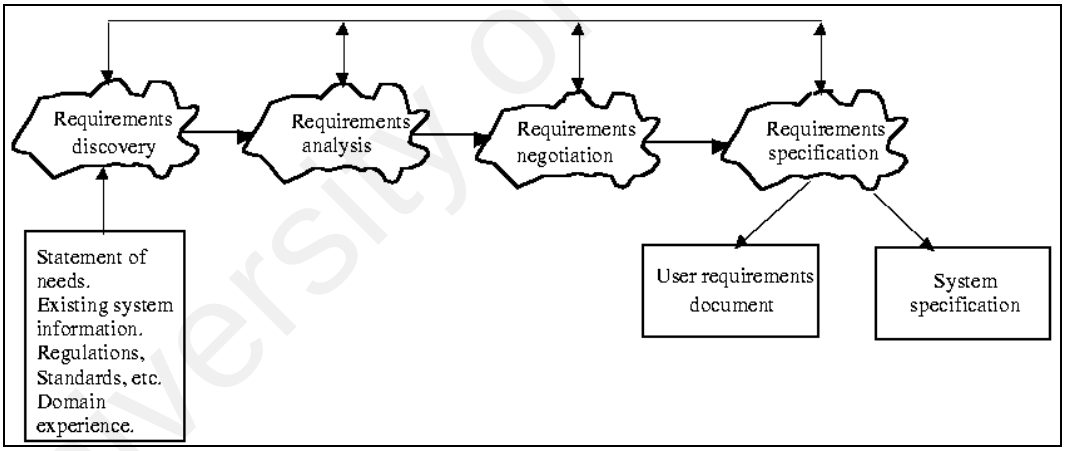


Figure 2.19: Generic Requirements Engineering Process

In order to execute requirements analysis, the problem domain must be well understood by the analysts, as the process used is often domain dependent. Most of the models of the process are inevitably simplifications. The process model shown in Figure 2.20 illustrates a number of important requirements analysis activities.

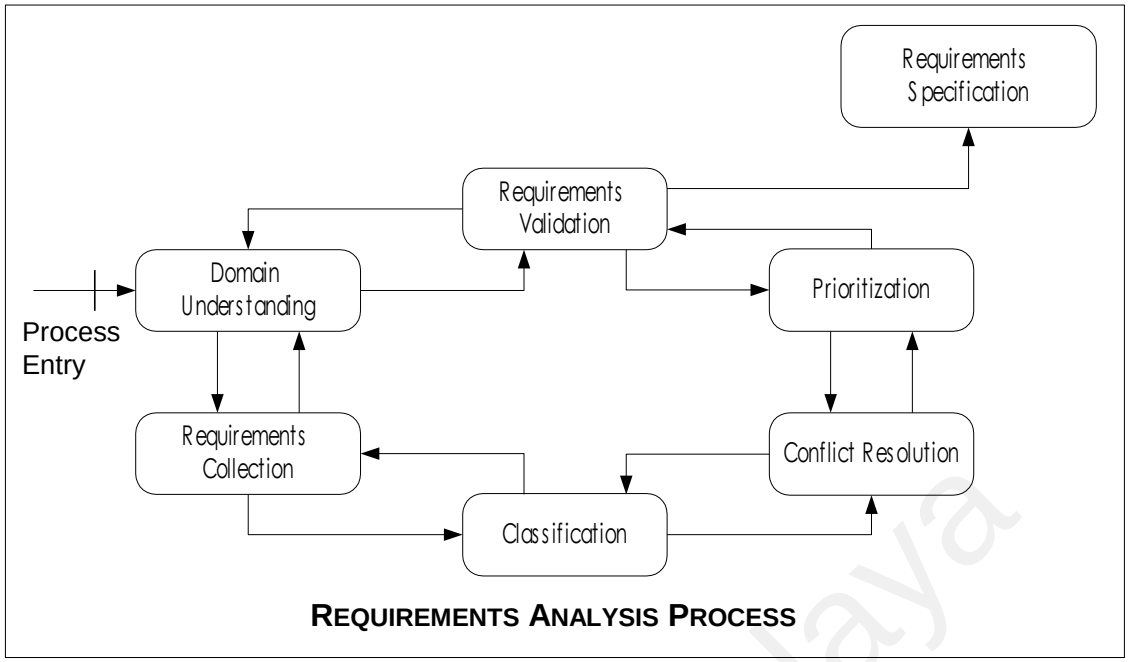


Figure 2.20: Requirements Analysis Process

As depict in Figure 2.20, the processes in requirements analysis are highly iterative with feedback from each activity to other activities. As Ian Sommerville quoted in his book,

> "*The process can be viewed as a cycle starting with domain understanding and ending with requirements validation. The analyst's understanding of the requirements improves with each round of cycle.*" (Sommerville 1996)

The process activities are:

- ***Domain Understanding***. In this process, the domain must be clearly understood by the analyst. For instance if a clinic management system is being developed, participants need to know as much as possible on the operational of a clinic management.

- ***Requirements Collection.*** All stakeholders interact and the requirements for a system are identified or discovered.

- ***Classifications.*** As the name of the process states, it is basically grouping of requirements into the relevant classes or categories.

- ***Conflict Resolution***. In any requirements analysis, requirements conflict especially when it involves more than one participant. So, at this stage, the conflicts are identified and resolved.

- ***Prioritization***. Approaching this stage, a set of requirements and its relevant categories or classes is prepared. Some requirements have higher precedence than others. Here is where the important requirements are identified.

- ***Requirements Validation***. Finally, the requirements are checked to see if they are complete, consistent and in parallel of users need.

As a final result of this stage would be the functional and non-functional requirements of the system being developed. This is a part of the software requirements specification in which could be used as an agreement between user and the developer before proceeding to the later stages of the implementation.

For this research, requirement analysis introduced by Ian Sommerville is selected. Among the reason implementing Ian Sommerville's requirement analysis, as deemed by DeMarco (1996) as foundations for a good requirements analysis methodology,

- Supports collaborative. As can be seen in the diagram above, almost all the activities require some level of co-operation and collaboration.

- Ability to express disagreements or issues and negotiating to resolve issues as done in conflict resolution.

- Producing a requirements document as a result of a successful requirement validation.

- Checking the completeness and consistency as done in requirements validation.

- Supports classifications of the requirements.

Table 2.7 compares the characteristics of the Ian Sommerviles method against the ones reviewed earlier

Table 2.7: Comparison of Ian Sommerville's methodology based on the advantage of the GBRAM, Win-Win, Inquiry Cycle and KJ

Characteristics	GBRAM	Win-Win	Inquiry Cycle	KJ	ISRA*
Repository for projects	✓	×	×	×	✓
Traceability.	✓	×	×	×	✓
Supports collaborative	×	✓	✓	×	✓
Ability to express disagreements or issues	×	✓	×	×	✓
Negotiating to resolve issues	×	✓	×	×	✓

Traceability of requirements decisions	×	✓	×	×	✓
Checking the completeness and consistency	×	✓	×	×	✓
Sharing information	×	×	✓	×	✓
Supports brainstorming session	×	×	×	✓	✓
Supports classifications of the requirements	×	×	×	✓	✓

**ISRA – Ian Sommervilles Requirements Analysis*

2.4.2 Groupware Supported

Based on the methods mentioned briefly earlier, a tool with groupware-supported facilities was developed. In other words, the aim is to develop groupware supported requirements analysis tool and at the same time reduce constrains on time and space. The model is discussed in detail later in Chapter 3. Tables 2.7 shows how GRAT fares when compared to other tools in terms of groupware applications.

Table 2.8: Comparison of the groupware features of GRAT against other tools

	GBRAT	WinWin	Ecolabor	KJ Editor	GRAT
Electronic Mail and Messaging	×	×	×	×	✓
Group Calendaring and Scheduling	×	×	✓	×	✓
Electronic Meeting Systems	×	×	✓	×	✓
Desktop Video and Real-time Data Conferencing (Synchronous)	×	×	✓	×	×

Non Real-time Data Conferencing (Asynchronous)	✗	✗	✗	✗	✓
Group Document Handling	✓	✓	✓	✗	✓
Workflow	✗	✓	✓	✗	✓
Collaborative – Internet-based Applications and Products	✓	✗	✓	✗	✓

As can be analyzed in the table above, GRAT provides a total groupware support for the requirement analysis activities. It provides almost all the integrated groupware facilities of the other 4 tools reviewed earlier except for desktop video conferencing. The reason for not being able to use desktop video conferencing is due to the development environment or platform does not support synchronous activities and also due to limitation on bandwidth.

2.4.3 Implementing a web-based tool

As mentioned earlier, a tool named GRAT or Groupware Supported Requirements Analysis Tool was developed. Making the tool web-based has given the tool an competitive and usability edge compared to other non-web based tools in the market. The advantages are:

- Open Network and Client Standard

  GRAT is running on the web. Meaning, anyone with an Internet browser and valid login information are able to use and benefit from GRAT.

- Setup Costs

  GRAT has significantly reduced setup costs as the installation of the system is only done at the server and does not require any installation on client. Purely representing thin client with some applets running is some of the process.

- Awareness

  GRAT provides the facility where email notification can be executed for meetings as well as discussions.

- Repository

  GRAT provides a repository for all the projects that were executed using that system which could be used for future reference, which indirectly feeds to the improvement of the process.

Other features mentioned in 2.4.2 that are included in GRAT are electronic mail and messaging, group calendaring and scheduling, group document handling, workflow, traceability of requirements, discussion forum and information sharing which are totally accessible from the web.

Table 2.9 summarizes the features of GRAT when compared to other tools mentioned earlier in the chapter.

Table 2.9: Comparison of GRAT against other tools based on the requirements analysis methods and supported architecture

Tool	Requirements Analysis Methods	Web-Based	Distributed	Stand Alone
GBRAT	Goal-based Requirements Analysis	Yes	Yes	No

WinWin	Win-Win	No	Yes	No
Ecolabor	Inquiry Cycle	Yes	Yes	No
KJ Editor	KJ	No	No	Yes
GRAT	Ian Sommerville's requirements analysis	Yes	Yes	No

2.4.4 Evaluating GRAT

GRAT was evaluated based on experimental materials that are mentioned in the later chapters. The evaluation is done to validate GRAT and not the issues on requirements analysis method.

2.5 Summary

This chapter reviewed some of the widely used requirements analysis methods and tools in the first section of this chapter. Next, it looked at the groupware from a general point of view and narrowed the view into requirements analysis by comparing groupware features in the requirements analysis tools. Finally, from the understanding on requirement analysis and how groupware complements the process, a general framework and features for GRAT was derived and identified.

CHAPTER 3: Groupware Support for a Requirements Analysis Model

3.1 Introduction

This chapter introduces the processes and features of GRAT. It discusses how Ian Sommerville's requirements analysis model, discussed in the previous chapter, is supported by the architecture of GRAT

3.2 GRAT Architecture

This section outlines groupware architecture to support the requirements analysis model described earlier. It exemplifies a conceptual design found in a generic requirements analysis process as proposed by Ian Sommerville (1996). This architecture was developed using Lotus Notes.

Figure 3.1 shows the requirements analysis process and Table 3.1 describes the phases of the requirements analysis process as defined by the Ian Sommerville (1996). Later, the architecture of GRAT is discussed based on the different phases identified.

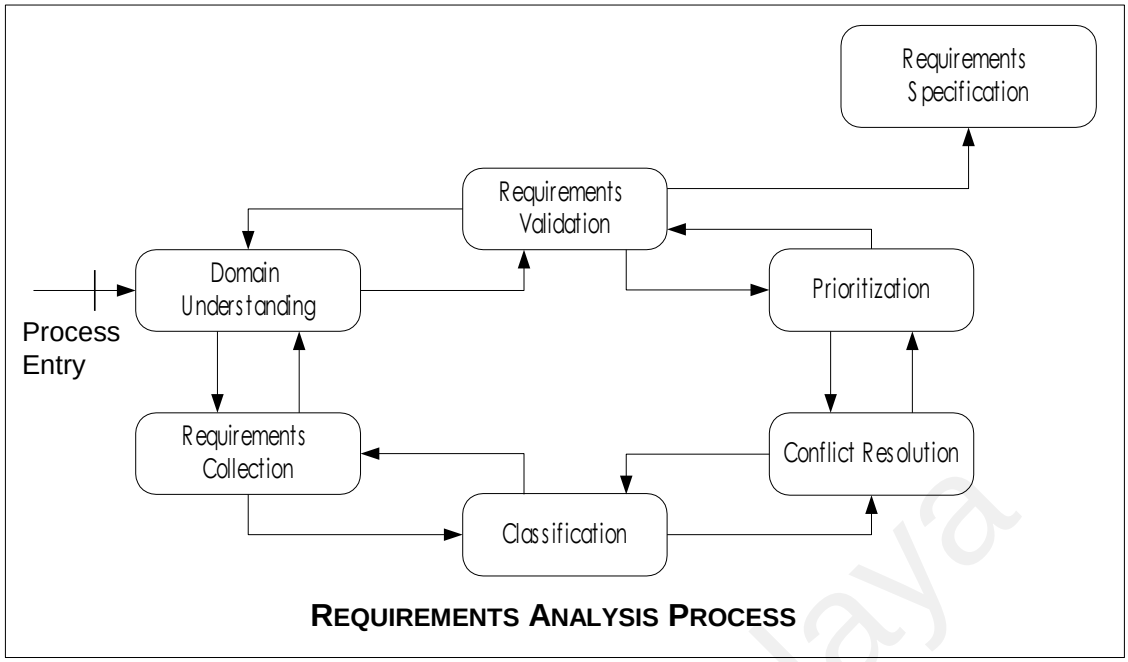


Figure 3.1: Requirements Analysis Process (Sommerville, 1996)

Table 3.1: Phases in Requirements Analysis.

Phase	Description
Domain Understanding	Analyst must develop their understanding of the application domain. Therefore if a system for a supermarket is required, the analyst must find out as much as possible about supermarkets
Requirements Collection	This is a process of interacting with stakeholders in the system to discover their requirements. Obviously domain understanding also develops during this activity.
Classification	This activity takes unstructured collection of requirements and organizes them into coherent clusters
Conflict Resolution	Inevitably, where multiple stakeholders are involved, requirements conflict. This activity is concerned with finding and resolving these conflicts.
Prioritization	In any set of requirements, some are more important than others. This stage involves interaction with stakeholders to discover the most important requirements
Requirements Validation	The identified requirements are checked to discover if they are complete, consistent and in accordance with what stakeholders really want from the system.

The phases mentioned in Table 3.1 are further described in terms of functionality and architecture in the following sections.

3.2.1 Domain Understanding

When analysts have some information to be shared with other team members, they submit the forms via web. They can submit the forms text basis, attach a file, or create a hyperlink. Once they have submitted the form, other users can look at the information in the View. They can choose to read the information by double clicking on the view or choose to add more information regarding the domain. This is shown in Figure 3.2.

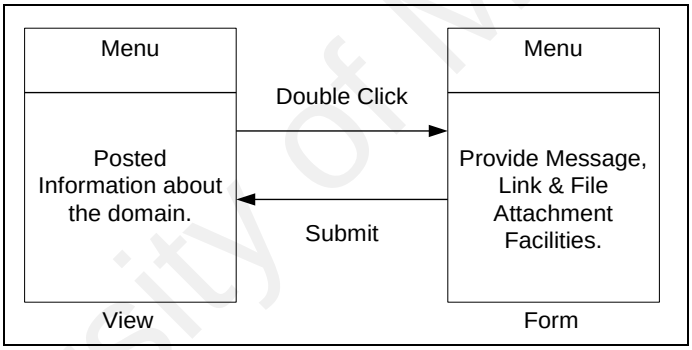


Figure 3.2: Domain Understanding in GRAT

3.2.2 Requirements Collection

Rapid generation of a single pool of ideas by individual group members with evaluation and criticism is being discouraged at this stage (Macaulay, 1997). Therefore the interface for brainstorming needs to permit the members of a distributed team with a means of inputting a suggested object at their terminal and sending it to update a common area which is seen by all members as depict in Figure 3.3.

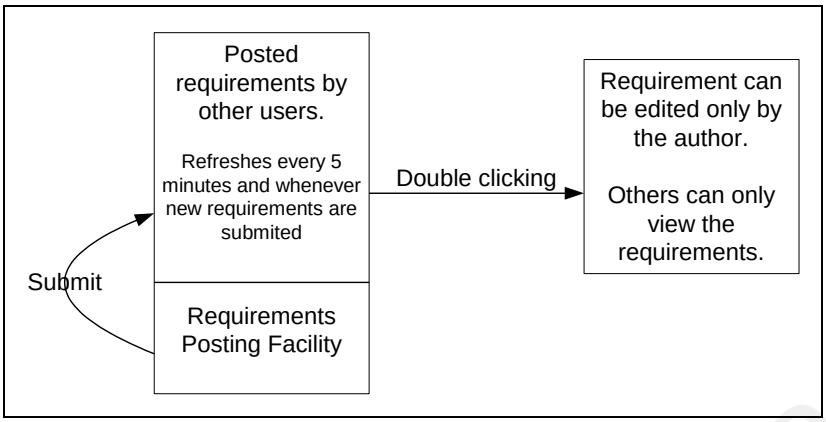


Figure 3.3: Requirements Collection in GRAT

Team members post the requirements that are feasible for the domain. All the team members are able to see the whole set of requirements submitted to the server. The list of posted requirement refreshes every 5 seconds or whenever a new requirement is posted. There will not be any communication facilities provided between team members.

3.2.3 Classification

Classify with lists according to functional, nonfunctional, environment and also design constrains; and also according to partitions defined by domain models and development paradigm (Christel, 1992). For this phase, it is divided into two sub-processes. One is to collect the list of categories and the other to classify the requirements into appropriate categories.

3.2.3.1 Categories Collection

While looking and browsing into the list of requirements, participants can start posting the appropriate categories they think suits for the requirements. The list of

categories refreshes every 15 seconds or when the participant proposes a new category. This is pictured in Figure 3.4.

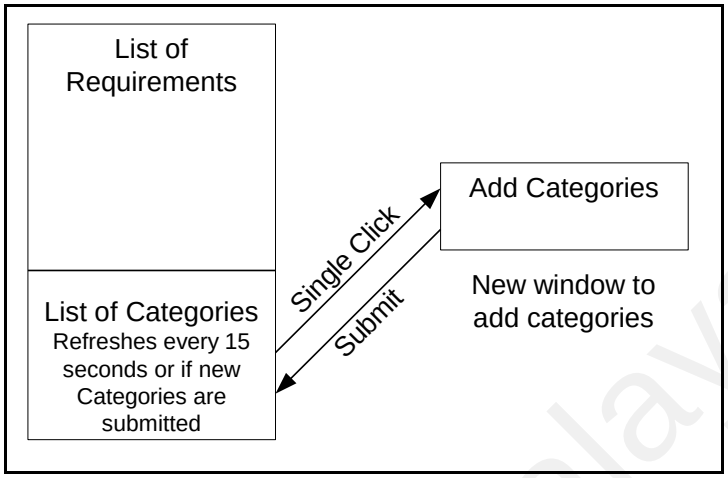


Figure 3.4: Categories Collection in GRAT

3.2.3.2 Classification

Participants have to match each requirement with appropriate category. The list of categories is provided in the list box. The System checks if all the requirements have been matched with an appropriate Category. Only then participants are allowed to preview their classification based on the categories. If they are not satisfied they can re-categorize the requirement. If they are satisfied, then they can submit to the server for further actions. Figure 3.5 shows the classification process in GRAT system.

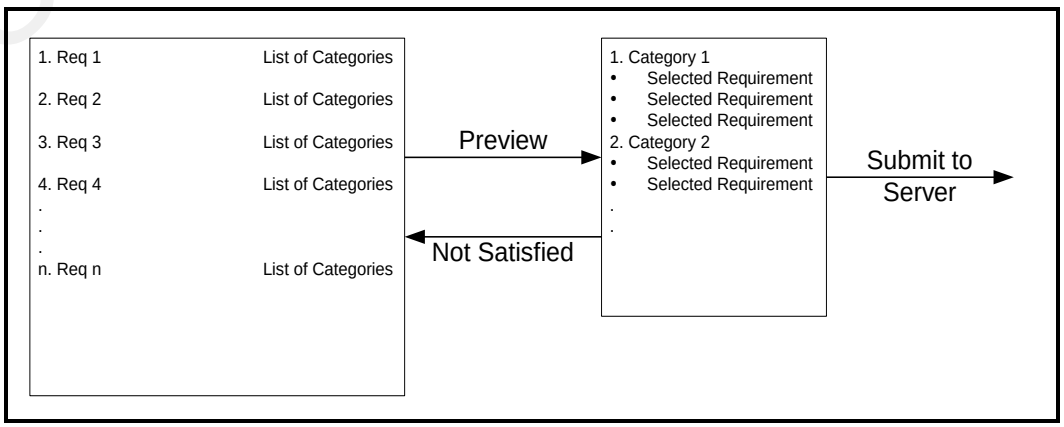


Figure 3.5: Classification in GRAT

After users have submitted their respective classification result, the result are processed and calculated. The calculation produces a summary of all the requirements based on the respective category. Figure 3.6 shows how GRAT handles the situation.

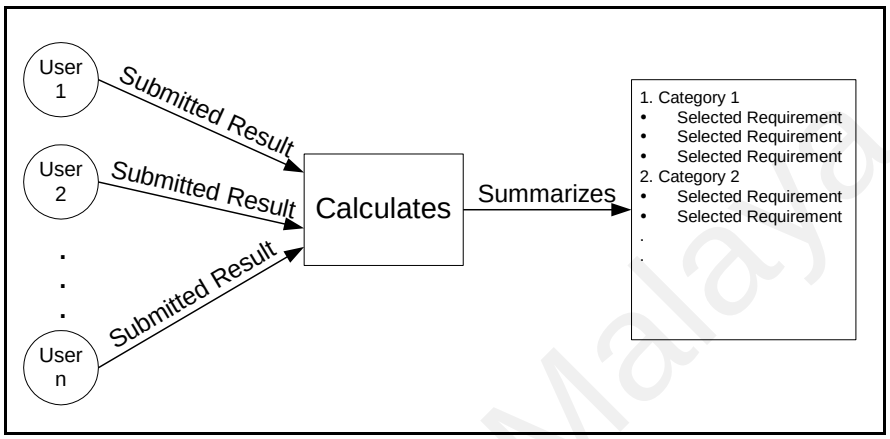


Figure 3.6: Computation of Classification Results in GRAT

3.2.4 Conflict Resolution

Perform abstraction to answer "Why do you need X?". This in effect moves statements of "how" to statements of "what" (Christel, 1992). Capture rationale to support future requirement evolution. The rationale behind the information collected in previous stages should be examined to determine whether the true requirements are hidden instead being explicitly expressed as depicted in Figure 3.7. After the classification result has been submitted to the server, project team members are able to see the requirements sorted out according to the categories. From here, if the participants feel there is a problem with the requirements, they can double click the row of the requirements and get to see the form. If they choose to post a question, then just by clicking an option, the web page directs it to another form where they can post their conflicts. Other members can view the conflicts just by double clicking them. They can

respond to the conflict by simply clicking a button. The conflicts and the responds to the conflicts are arranged in a hierarchy form.

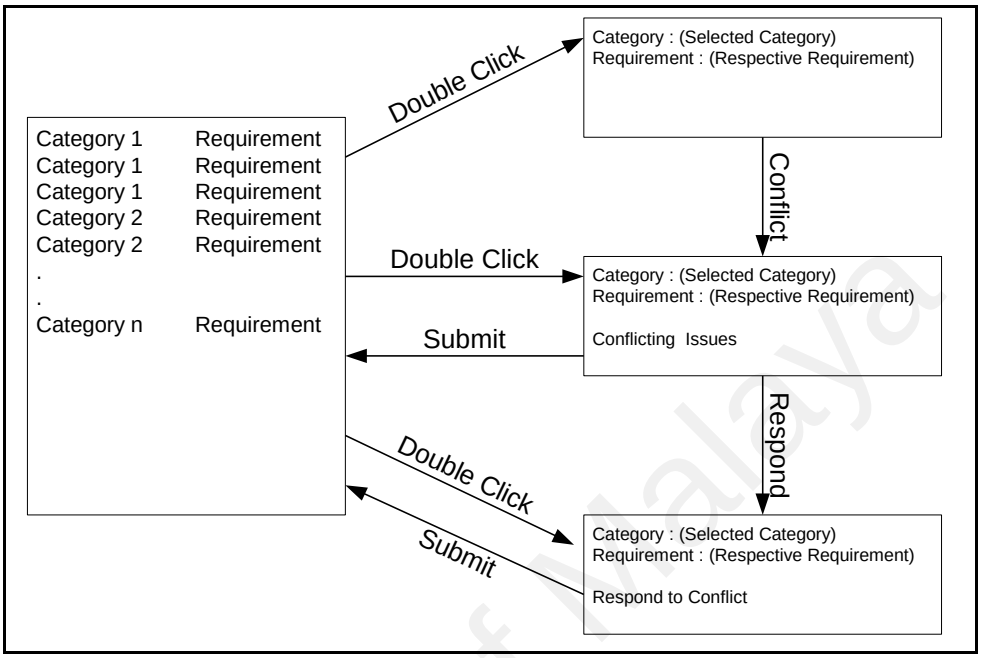


Figure 3.7: Conflict Resolution in GRAT

3.2.5 Prioritization

Here is where the criticality of the requirement is determined. The requirements are prioritize based on importance, cost and dependency (Christel, 1992). Participants have to prioritize each requirement in their respective categories. The values of prioritize is numbered. The range depends on the number of requirement in each category. For example, if a category contains 6 requirements, then the range would be 1 to 6. The values are provided in list box. 1 represents most prioritize and 6 represents the least prioritize. The System checks if all the requirements in a category have a unique value. Only then participants are allowed to preview their prioritization. The requirements are arranged according to the priority given by participant. If they are not

satisfied they can re-prioritize the requirement. If they are satisfied, then they can submit to the server. The process is clearly shown in Figure 3.8.

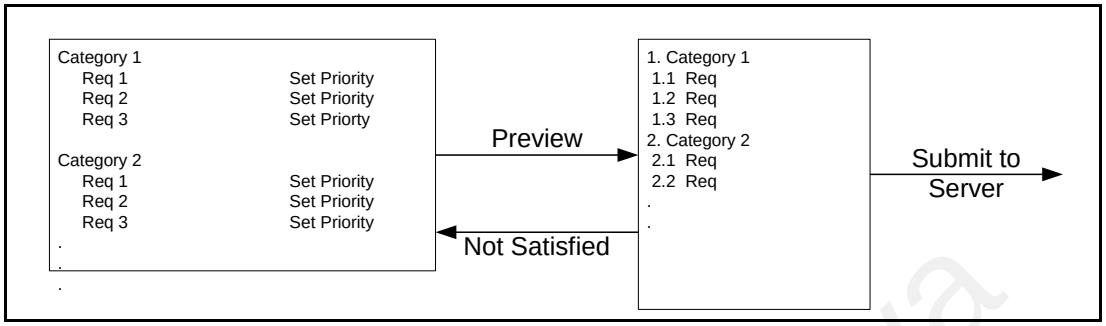


Figure 3.8: Prioritization in GRAT

After users have submitted their respective classification result, the result are processed and calculated. The calculation produces a summary of all the requirements based on the respective category as depict in Figure 3.9.

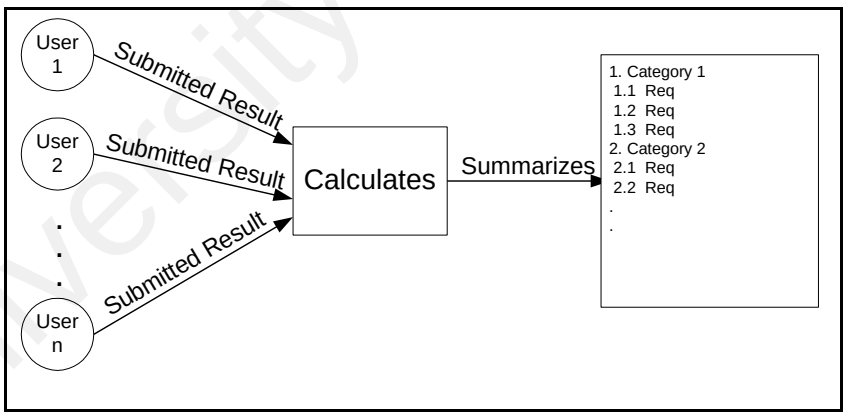


Figure 3.9: Prioritization in GRAT

3.2.6 Requirements Validation

Consistency checking, validating the requirements that are in agreement with originally stated goals and obtain authorization or verification to move to the next step of development is the routine conducted in this phase (Christel, 1992). Participants who

feel that particular requirement have some problem, double clicks on the requirement, and it shall prompt user to the requirement validation form. The validating options are based from Ms Ow's (1998) book. Once the form has been submitted, the submitted result is displayed below the respective requirement as shown in Figure 3.10.

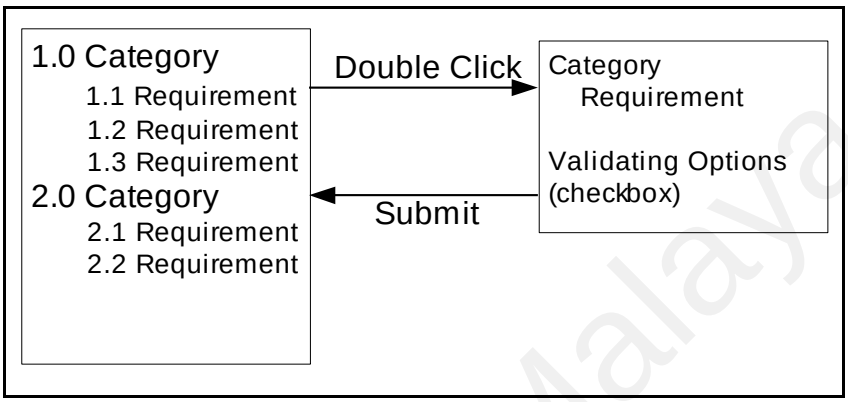


Figure 3.10: Requirements Validation in GRAT

3.3 The Role of Project Manager

In order for the requirements analysis to be conducted in a effective manner, there is a need for a person in the project team to initiates the project, orchestrate the activities, have a final say when there is a conflict arises and unable to be solved by the team members. This role is mostly played by the Project Manager. Similarly, in GRAT the Project Manager is needed to ensure that the processes and product of the different phases mentioned above is as closely coupled with the objectives of the project.

The role of a facilitator or Project Manager is undeniable. He/she is the core for the success of the project (Viller, 1991). The Shorter Oxford English Dictionary clearly defines that facilitate is "To render easier; to promote, help forward". Therefore it is concerned with assisting the group members in performing their collective task as a group (Macaulay, 1997).

This research takes into account the role of the facilitator seriously as mentioned by Linda A.Macaulay (1997) in her PhD thesis. She mentioned that the function of the facilitator is very important at the initial stage. With the knowledge of the group process, the facilitator can utilize this position to improve group cohesion and for setting of group norms. However as the group process moves on to the middle stage the facilitator becomes less important where they are mere enabler and intervening when necessary. When the process reaches the final stages, the role becomes important. The precise role played by the facilitator at this stage depends upon the circumstances in which the group is breaking up. Meaning, whether or not the group has fully achieved its purpose (Douglas, 1970).

According to Marsh (1991) who introduced Quality Function Deployment, the facilitator has to be,

- Planner
  - Help team establish objectives.
  - Develop agenda.
  - Establish dates, times and places for meetings.
- Guide
  - Explain the process
  - Regulate the flow of the process
  - Monitor participation
- Cheerleader
- Coach
  - Develop the team.
  - Facilitate consensus decision-making.
- Arbitrator

- Helps settles disputes and conflicts- collaborative effort
- Keep the team spirit alive
- Ensure problems are being solved

For GRAT, the project manager will be the planner, the guide, the coach and the arbitrator.

3.4 Summary

This chapter has introduced the architecture to support GRAT. Each phase of the requirements analysis has been identified as modules of GRAT. The modules are discussed in detail on its architecture. The role of project manager for GRAT was also introduced in this chapter. The following chapters will discuss the analysis, design and implementation of GRAT.

CHAPTER 4: GRAT Analysis and Design

4.1 Introduction

This chapter looks into the analysis and design of GRAT. The former identifies the entire functional and non-functional requirements. An analysis using object-oriented analysis is presented. The design part also uses object-oriented design for the architecture of the system.

4.2 GRAT Analysis

4.2.1 Requirements Analysis

This section is broken into two sub divisions, functional and non-functional requirements. Functional requirements are functionality of GRAT system, and non-functional requirements describe other aspects such as usability, efficiency and other runtime properties.

4.2.1.1 Functional Requirements

There are nine modules that were identified for the project. Six are based on Ian Sommerville's (1996) requirements analysis method as mentioned in Chapter Two, one is to support his model, another as a project repository or database and the last one is too support scheduling. The modules are,

- Project Repository
- Domain Understanding
- Requirements Collection

- Categories Collection
- Classification
- Conflict Resolution
- Prioritization
- Validation
- Activity Scheduling

There system also supports three roles. One is a facilitator as mentioned at the last part of Chapter Three who will be called, as Project Manager, Domino Administrator and the others will be identified and Team Members.

GRAT is a web-based requirements analysis tool. Meaning, by default it is a client-server web application. Clients are required to have a web browser and the server is the host computer of the organization.

Project Repository

i. The system lists all the projects the organization is involved.
ii. The system lists the project title, the relevant PROJECT MANAGER, START DATE, END DATE and PROJECT WEB SITE.
iii. The system provides links for all the projects.
iv. The system opens the PROJECT DETAILS page of the double-clicked project on the list. The PROJECT DETAILS page displays the name of the project, start date, end date, team members, details of the project and a link to the project web site. It has a button to return to the PROJECT REPOSITORY page.
v. The system provides a button and function for NEW PROJECT and DELETE PROJECT.

vi. The system only allows ADMINISTRATOR to use DELETE PROJECT.

vii. The NEW PROJECT button opens up a form that requires the PROJECT MANAGER to fill in the information. The user who clicks the New Project form is the PROJECT MANAGER. The information needs to be filled are name of the project, start date, end date, team members, details of the project. The start date is auto-generated from the server and it should display the current date. It shall have three buttons that are TEAM MEMBERS to select the team members, REGISTER to register the project and ABORT to ignore the information and open up the Project REPOSITORY page. The System checks to see all the information has been filled in. The System submits the form to the ADMINISTRATOR when the PROJECT MANAGER clicks the REGISTER button.

viii. The ADMINISTRATOR approves the project and provides an automated URL for the project. The ADMINISTRATOR can relocate the URL to other than the generated URL. Approved project has the website listed in the PROJECT REPOSITORY page.

Domain Understanding

i. TEAM MEMBERS are able to share information with other TEAM MEMBERS and PROJECT MANAGER via web form.

ii. TEAM MEMBERS are able to submit the forms on text basis, attach a file, or create a hyperlink.

iii. The system allows all TEAM MEMBERS to look at the information in the page.

iv. TEAM MEMBERS are able to choose to read the information by double clicking on the selected list

v. The system provides a button for TEAM MEMBERS to submit the information they want to share.

Requirements Collection

i. TEAM MEMBER are able post the requirements that will be feasible for the domain.
ii. All TEAM MEMBER are able to see the whole set of requirements submitted to the server.
iii. The system refreshes the list of submitted requirements every 5 seconds or whenever a new requirement is posted.

Categories Collection

i. TEAM MEMBERS are able to post the appropriate categories while looking and browsing into the list of requirements.
ii. The system refreshes list of categories every 15 seconds or when the TEAM MEMBER posts a new category.

Classification

i. TEAM MEMBER matches each requirements with appropriate category.
ii. The list of categories is provided in the list box.
iii. The system checks if all the requirements have been matched with an appropriate Category.
iv. The System previews the classification result of each TEAM MEMBER based on the categories if all the requirements have a matching category. TEAM MEMBERS who is not satisfied with their choice are allowed to modify their selection.

v. The system allows TEAM MEMBERS to submit their classification result.

vi. The system calculates and computes the overall result from the submission of the TEAM MEMBER. The calculation and computation is based on voting process.

Conflict Resolution

i. The system allows TEAM MEMBER to see the requirements sorted out according to the categories.

ii. The system allows users voice opinion for a particular requirement.

iii. The system displays a form for TEAM MEMBER to voice out conflict regarding a requirement by double clicking.

iv. The system allows responds to that conflict by clicking a button. The conflicts and the responds to the conflicts are arranged in a hierarchy form.

Prioritization

i. TEAM MEMBERS are able to prioritize each requirement in their respective categories.

ii. The values of prioritize are numbered. The range depends on the number of requirements in each category.

iii. The system provides the values in list box. 1 represent highly prioritize and 6 will represent the least prioritize. The value is converted to score.

iv. The system checks if all the requirements in a category have a unique value.

v. The system previews the prioritization result of each TEAM MEMBER based on the categories if all the requirements have a unique priority. TEAM MEMBERS who is not satisfied with their choice are allowed to modify their selection.

vi. The system computes and totals the overall score from the submission of the TEAM MEMBER. The calculation and computation is based on voting process. The lowest scored requirement will be recorded as highest priority.

Validation

i. The system allows TEAM MEMBERS who feel that a particular requirement has some problem, to double click on the requirements, and prompt TEAM MEMBER to the requirements validation form.

ii. The system displays the submitted result displayed below the respective requirements.

Activity Scheduling

i. The system displays the scheduled activities for the current week and following week in the project web site.

ii. The system provides a page to submit new activity. TEAM MEMBER has to fill in the date of the activity is suppose to be held and activity description. The default date is the current date of the system and it can be edited. The system provides auto text generation for the requirements analysis activities, which are Domain Understanding, Requirements Collection, Categories Collection, Classification, Conflict Resolution, Prioritization and Validation.

Project Manager

i. PROJECT MANAGER is able to change the project status and set the current project status. The statuses are Domain Understanding, Requirements Collection, Categories Collection, Classification, Conflict Resolution, Prioritization and Validation.

ii. PROJECT MANAGER is able to delete the materials collected at all activities.

Domino Administrator

i. DOMINO ADMINISTRATOR is able to provide URLs for the respective project either by using default values or edit the values.

4.2.1.2 Non-Functional Requirements

The non-functional requirements are considered when building GRAT in addition to the above mentioned functional requirements. The requirements are stated in the following sub sections.

Usability

The system must be built taking into considering the groupware activities. It should take into consideration different level of users and thus provide the respective tools that support the respective users.

Maintainability and expandability

The system must be easily updated, maintained and expanded from time to time without affecting the current data and/or information.

Portability

In order to make the system as a web application, clients only require a web browser without worrying of the platform. The system must be aimed to be running on Microsoft Internet Explorer version 5.5 and above.

Efficiency

GRAT must be able to provide a good response time. Being a web-based application, GRAT should consider minimizing response, process and page reloading time.

Scalability

GRAT should support scalability, meaning that the services of the system must be provided regardless of the size of the system. There should not be a limitation or whatsoever on the number of requirements or categories, which mean the system, should support a small scale or a big scale project.

4.2.2 Object-Oriented Analysis

For this research, the GRAT was developed using object-oriented analysis and design with traditional implementation. The main benefit using this approach is being able to model the problem domain using OO analysis. OO analysis promises a more natural medium for communication between end user and the analysts than current structured analysis techniques provide (Wilkie, 1993).

The Unified Modeling Language or UML (UML, 2000) was used to represent the system's diagrams. The analysis and design steps presented in Appendix A are based on the reference made to S.R.Schach (Schach, 1999) The diagrams are presented in Appendix A.

4.3 GRAT Design

4.3.1 GRAT Architecture

GRAT is a client server application. The system is a two-tier system. Meaning that only two layers will be interacting with each other. The two layers are the presentation layer which runs in the client and the application layer which runs in the server.

4.3.1.1 The Presentation Layer

Since the system is running on the web, the client is triggered by the web browser. The browser will get connected to the application that runs on a web server. It will be the interface for interactions between users and the application running background.

4.3.1.2 The Application Layer

The GRAT application resides on the server. It binds to a certain IP address waiting for clients requests and respond to it. It should be ale to handle all synchronous and asynchronous jobs.

4.3.2 Object-Oriented Design

Object-oriented design complements object-oriented analysis (Wilkie, 1993). Therefore, the transition from analysis to design is straightforward. A full object-oriented design will consist of a generalized and optimized class hierarchy design along with an application design based on the class design. Since GRAT will be applying

traditional method for the implementation stage, this research has only included the class design and collaboration diagrams. The diagrams can be seen in Appendix A.

4.4 Summary

As seen and observed, this chapter presents the analysis and design specifications of GRAT. It was stared by stating the functional and non-functional requirements, followed by analysis, which was done using object-oriented methodology. Use case diagrams were developed at this stage. The second part looked into the design aspect of GRAT. It also contains the general architecture of GRAT. Again, using object-oriented modeling techniques, several class diagrams and interaction diagrams is presented. The diagrams are presented in Appendix A for the sake of simplicity.

CHAPTER 5: GRAT Implementation and Execution

5.1 Introduction

This chapter looks into the implementation of and the execution of GRAT. The implementation environment is discussed in the early part of this chapter. Following that, the implementation phase is discussed and finally the execution of GRAT is discussed in detail.

5.2 Implementation

There are a number of tools and applications used to implement GRAT. The following sub sections discuss all of them generally.

5.2.1 Environment

5.2.1.1 Lotus Domino and Lotus Notes

Domino is a workgroup application that allows people to share information using networks. Domino servers and Lotus Notes workstations can communicate over local area networks (LANs) and wide area networks (WANs). For this research, Lotus Domino Release 5 was used because it brings messaging, Internet integration, and scalability in one system. The new Domino server includes the latest innovations in Internet messaging, with native support for all the major Internet standards, industry-leading support for Web applications, including CORBA support and integration with Microsoft Internet Information Server (IIS) and increased server reliability and scalability, including improvements in performance, capacity, availability, and maximum database size (Burch, 1999). In addition, the server has a new administration

interface, with a task-oriented approach that makes Domino easier to deploy, use, and manage. Domino Release 5 continues to support a wide variety of clients, in addition to the traditional Notes client.

Messaging features are available to Web browsers and Internet mail clients (such as POP3 and IMAPv4 clients) where else directory features are available to browsers and Lightweight Directory Access Protocol (LDAP) clients. Discussion features are available to browsers and NNTP newsreader clients and administration features are available to browsers as well as the Notes client. Plus, Domino continues to be the best platform for designing dynamic Web applications, and with the new Domino Designer R5 a single application that looks and runs the same for both the Web and Notes clients can be built (Burch, 1999).

In summary, specific Domino R5 features include:

- Internet messaging and directories
  - Provide full-fidelity messaging for your users, with native MIME and SMTP support
  - Use the new Directory Catalog to save space and provide quick name lookups
  - Use new LDAP features to authenticate users in external directories and customize the directory

- Expanded Web application services
  - Design applications with CORBA-standard distributed objects, Java, or JavaScript
  - Use Web clusters for high availability of Web services, expanded security options, and more

- Run Domino using the Microsoft Internet Information Server (IIS) HTTP services

- Database improvements
  - Use transactional logging for faster restarts and data recovery
  - Convert to the new on-disk structure (ODS) for better performance and data integrity

- Easier administration
  - Manage users, databases, and servers with the new Domino Administrator
  - Migrate users from cc:Mail, Microsoft (MS) Mail, Exchange, GroupWise, Netscape Mail, LDAP, or Windows NT with the redesigned user registration
  - Use new tools for server monitoring and message management

Domino Designer is an integrated application development environment in which lets developers and Web site designers create, manage, and deploy secure, interactive applications for the Domino Server. Domino applications let people share, collect, track, and organize information, using Lotus Notes or the Web. Domino applications can cover a wide range of business solutions. For this research, the following features of Domino applications have been implemented.

- *Workflow*. Applications that route information.
- *Tracking*. Applications that monitor processes, projects, performance, or tasks.
- *Collaboration*. Applications that create a forum for discussion and collaboration.
- *Personalization*. Applications that produce dynamic content based on, for example, user name, user profile, access rights, or time of day.

Every Domino application starts with a Domino database. All Domino applications contain one or more Domino databases. A database is the container for the data, logic, and design elements of Domino application. Design elements are building blocks you use to create your application. Design elements that were used for this research include pages, forms, outlines, views, framesets and shared resources.

5.2.1.2 Formula

Formulas are expressions that have program like attributes. For example, one can assign values to variables and use a limited control logic. The formula language interface to Domino is through calls to @functions. @Commands, a subset of the @functions, provide access to the user interface (Lotus, 2000).

Formula language provides syntax and @functions for evaluating constants and variables, and for performing simple logic. Variables can be fields in Notes documents or temporary variables (also called temporary fields) used only for the immediate formula.

5.2.1.3 Lotus Script

LotusScript is an embedded, BASIC scripting language with a powerful set of language extensions that enable object-oriented application development within and across Lotus products (Lotus, 2000). LotusScript allows one to place more complex scripts in a greater variety of locations and events than traditional macros. LotusScript and its development toolset provide a common programming environment across

LotusScript offers a wide variety of features. Its interface to Lotus products is through predefined object classes. The products oversee the compilation and loading of

user scripts and automatically include class definitions to allow more efficient coding. LotusScript offers the following advantages (Lotus, 2000):

- Superset of BASIC

  Since LotusScript is a superset of the BASIC language, it is easy to learn, especially for Visual Basic users. Sophisticated scripts can be written using conditions, branches, subroutines, while loops, and other conventions

- Cross-platform

  LotusScript is a multi-platform BASIC-like scripting language. It works with platforms such as Windows, Macintosh, OS/2, UNIX, OS/390, and AS400. Scripts developed on Windows execute unchanged on any other supported platform. This portability is important as desktop applications become workgroup-enabled and documents are e-mailed to or shared by users.

- Object-oriented

  Lotus products provide Object Classes that are available to LotusScript. Scripts can be written to access and manipulate these objects. The scripts are event-driven, such as by an action, clicking the object or button, opening a document, or opening a view.

- Included in Lotus applications

  LotusScript is supported by Lotus products, so these products can access product classes using a product-supplied LotusScript extension.

5.2.1.4 Internet

Internet is a vast ocean of computers connected over network of cables, as is in Figure 5.1. As of 1996, the Internet was serving over 20 million users worldwide. A quick glance through the history of Internet reveals that Internet began from an obscure network known as ARPANET that was used by America's Department of Defense, its contractors and defense researchers (Honeycutt, 1997). In the late 1980s, this phenomenon wave started to spread its wing across academic organizations around the world and was greatly used by the academic community (Honeycutt, 1997). In 1991, the National Science Foundation of United States of America, who has been funding the Internet all this while, lifted the ban on commercial traffic (Moore, 1994). The results of it have basically changed the living and working culture for most of the people regardless of where they are.

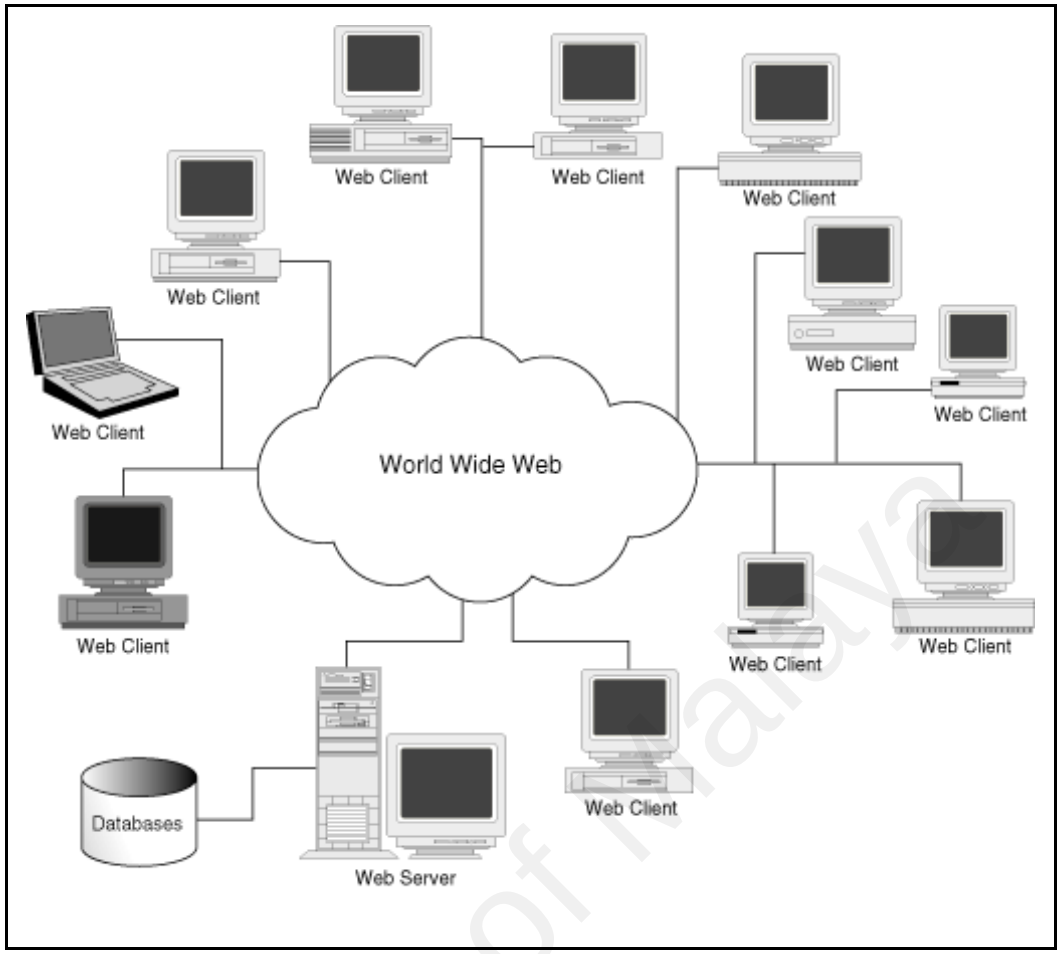


Figure 5.1: Internet in a glance.

The World Wide Web (web) is a network of information resources. The web relies on three mechanisms to make these resources readily available to the widest possible audience, which are (1) a uniform naming scheme for locating resources on the web, (2) protocols for access to named resources over the web and (3) hypertext for easy navigation among resources.

5.2.1.5 HyperText Markup Language (HTML)

To publish information for global distribution, one needs a universally understood language, a kind of publishing mother tongue that all computers may

potentially understand. The publishing language used by the World Wide Web is HTML. HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos and others.
- Retrieve online information via hypertext links or at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spreadsheets, video clips, sound clips, and other applications directly in their documents.

HTML was originally developed by Tim Berners-Lee while at European Laboratory for Particle Physics or CERN, and popularized by the Mosaic browser developed at NCSA (Reggett, 1999). During the course of the 1990s it has blossomed with the explosive growth of the Web. During this time, HTML has been extended in a number of ways. The Web depends on Web page authors and vendors sharing the same conventions for HTML. This has motivated joint work on specifications for HTML, which is headed by WorldWideWeb Consortium (W3C).

As it hearts, the WWW is nothing but a big collection of HTML files in the connected computers. Web browsers' takes the HTML codes and make it presentable on-screen to the user. HTML is not intended to be an all-encompassing, all-powerful page layout environment. HTML basically uses markup tags to indicate the relative position of elements on the page. However, there are limitations on HTML. Among them are type and size of font, color of the text and screen background. Figure 5.2 briefly depicts where HTML is during a web page is displayed on the web browser.

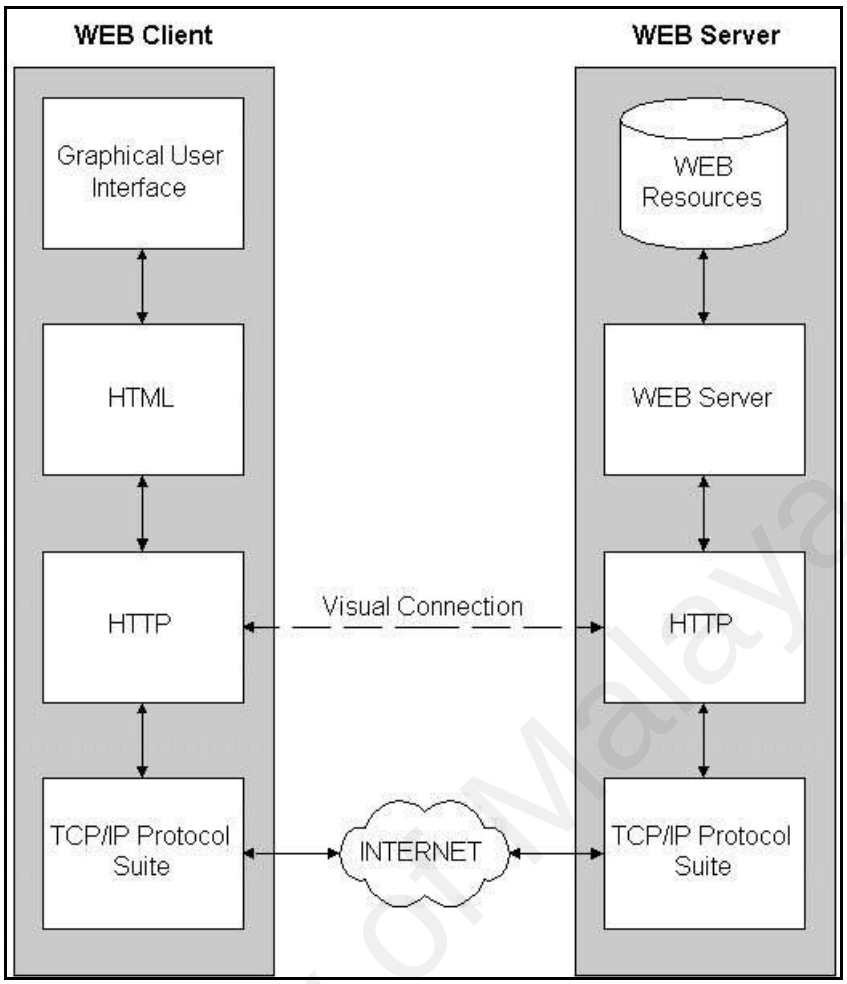


Figure 5.2 : HTML during a web page is displayed

HTML was first developed to maintain the compatibility of the files with any platform or any browsers. However, there are three major defects that make it impossible. The defects are:

- HTML is an evolving standard. Not all the browsers are compatible with the latest HTML standard set by the World Wide Web Consortium (W3C).

- Competition to dominate the browser market has caused two major players, Microsoft and Netscape, to introduce their very own browser-specific tags.

- The operating system and the display characteristics of a computer cause a variation in displaying of pages in the web.

From the research done, HTML's advantage is that it is easy to learn and easy to use. HTML is meant to be open-platform, meaning one could run it from text-only UNIX terminals to the flashiest Silicon Graphics workstation. Although, there certainly be some differences in the same Web page viewed on different machines with different browsers, the results are acceptable enough to convey the information on the page.

5.2.1.6 JavaScript

JavaScript is Netscape's cross-platform, object-oriented scripting language. Client-side JavaScript extends the core language by supplying objects to control a web browser and its Document Object Model (DOM) (JavaScript, 1999). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. JavaScript allows applications run over the Internet. Using JavaScript, dynamic HTML pages can be created that processes user input and maintain persistent data using special objects, files, and relational databases.

Web browsers can interpret client-side JavaScript statements embedded in an HTML page. When the browser requests such a page, the server sends the full content of the document, including HTML and JavaScript statements, over the network to the client. The browser reads the page from top to bottom, displaying the results of the HTML and executing JavaScript statements as they are encountered. The components of JavaScript are shown in the figure below.

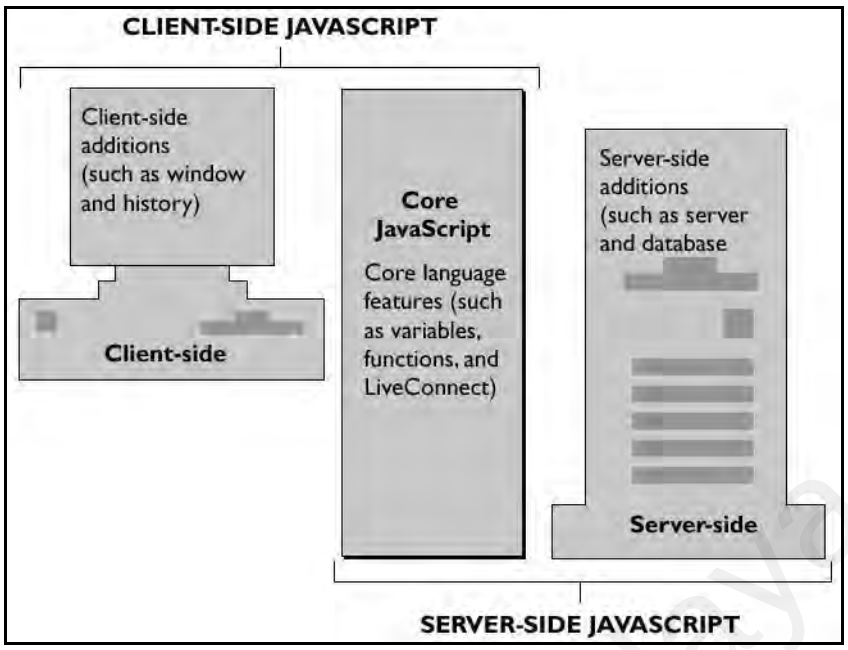


Figure 5.3: Components of JavaScript

Client-side JavaScript statements embedded in an HTML page can respond to user events such as mouse clicks, form input, and page navigation. For example, a JavaScript function could be used to verify that users enter valid information into a form requesting a telephone number or zip code. Without any network transmission, the embedded JavaScript on the HTML page can check the entered data and display a dialog box if the user enters invalid data.

5.2.1.7 Java Applets

Java applets provide a fascinating layer on top of the already dynamic Java language that extends far beyond traditional programming architecture and methodology (Weber, 1997). An applet is a program that is not only runs on any computer but also can be included in a standard HTML page. An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included. When a Java technology-enabled browser is used

to view a page that contains an applet, the applet's code is transferred to local system and executed by the browser's Java Virtual Machine (JVM).

The advantages of integrating Java applets with web browsers are,

- Applets can usually make network connections to the host they came from.
- Applets running within a Web browser can easily cause HTML documents to be displayed.
- Applets can invoke public methods of other applets on the same page.
- Applets that are loaded from the local file system have none of the restrictions that applets loaded over the network do.
- Although most applets stop running once one leaves the page, they don't have to.

However, there are some setbacks using applets. Current browsers impose the following restrictions on any applet that is loaded over the network(Sun, 2000).

- An applet cannot load libraries or define native methods.
- It cannot ordinarily read or write files on the host that's executing it.
- It cannot make network connections except to the host that it came from.
- It cannot start any program on the host that's executing it.
- It cannot read certain system properties.
- Windows that an applet brings up look different than windows that an application brings up

5.2.1.8 Web Browser

Web browsers are software used by web surfers to view information within World Wide Web. It is used to view pages on and navigate the World Wide Web. Internet surfers are not limited to a specific kind of machine or platform. The browser takes the information it gets from the web server, formats and displays it in screen. Different browsers depending on the capabilities of system and the browsers might display same files differently. Figure 5.4 describes the role of web server in the Internet.

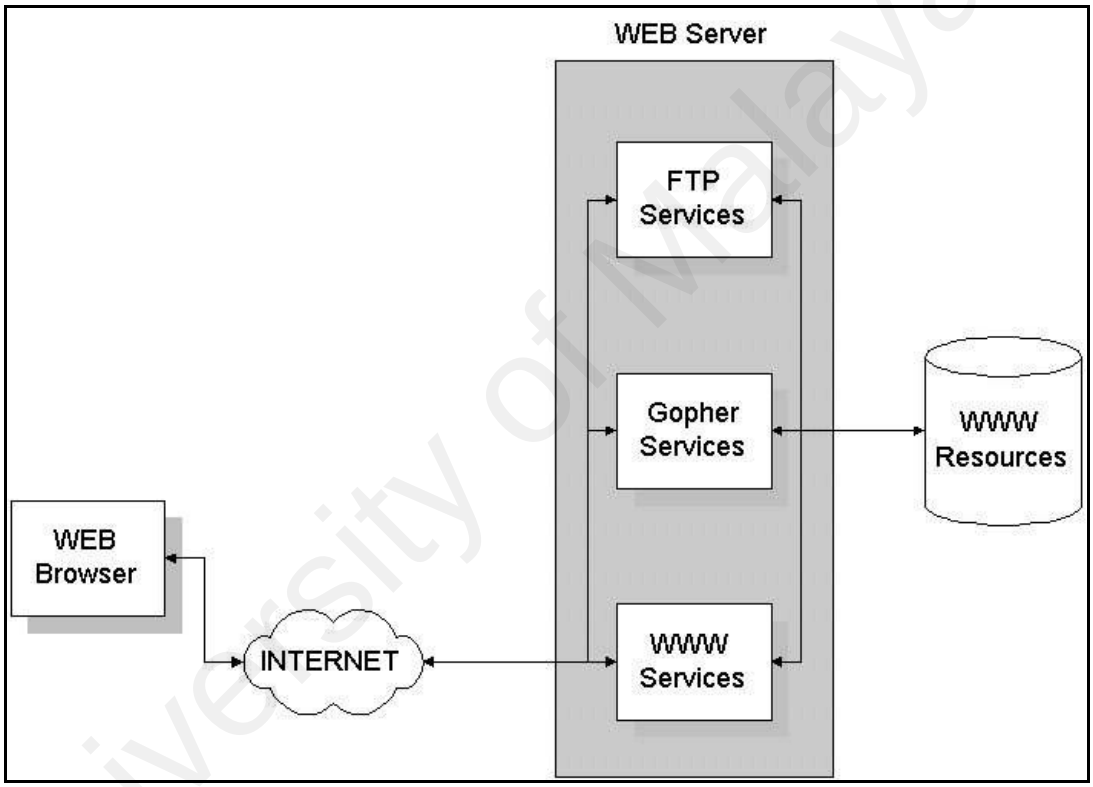


Figure 5.4: Web browser role in the Internet (Abstracted from Walther S.)

For this research, Microsoft Internet Explorer 5.5 or later is recommended due to its capabilities to support Java applets developed by Lotus, JavaScript codes and HTML 4.01.

5.2.1.9 Windows NT

Windows NT 4.0 is the fourth generation of Windows NT. Windows NT is a cross-platform product with identical versions is available for Intel X86, Digital Alpha, Silicon Graphics MIPS, and Apple/IBM/Motorola PowerPC computers (NTServer, 2000). Major hardware manufacturers such as Hewlett-Packard, IBM, Digital, Tandem, Amdahl, and Unisys offer high-end servers designed specifically to run A simple understanding on server and clients can be viewed in Figure 5.5.

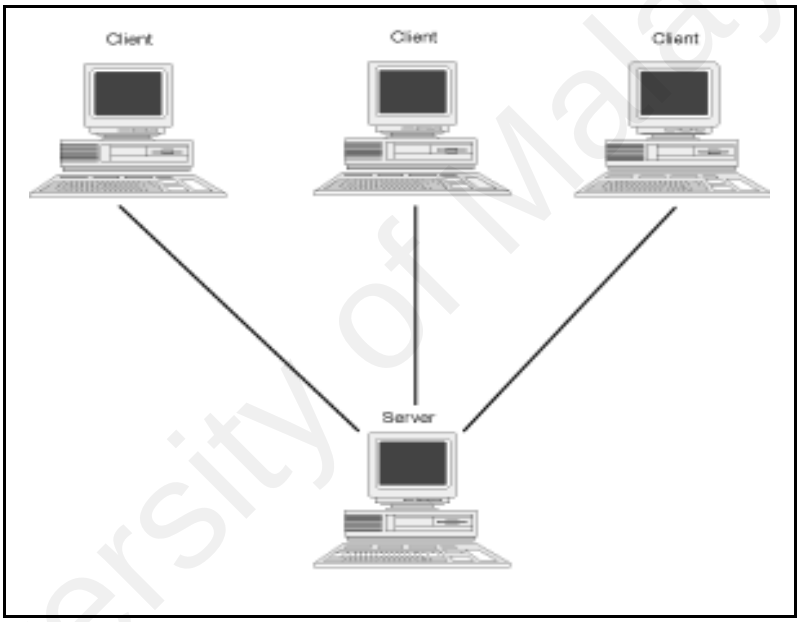


Figure 5.5: Client computer connected to a Server.

Windows NT 4.0 adopts Windows 95's user interface (UI) and operating system shell. Beneath the interface improvements, Windows NT Server 4.0 provides several new networking features, the most important of which for networking is the Distributed Common Object Model (DCOM) and a substantial improvement in the Domain Name Service (DNS) for TCP/IP networks. Figure 5.6 shows how Windows NT complies with Open System Interface(OSI) (NTServer, 2000).

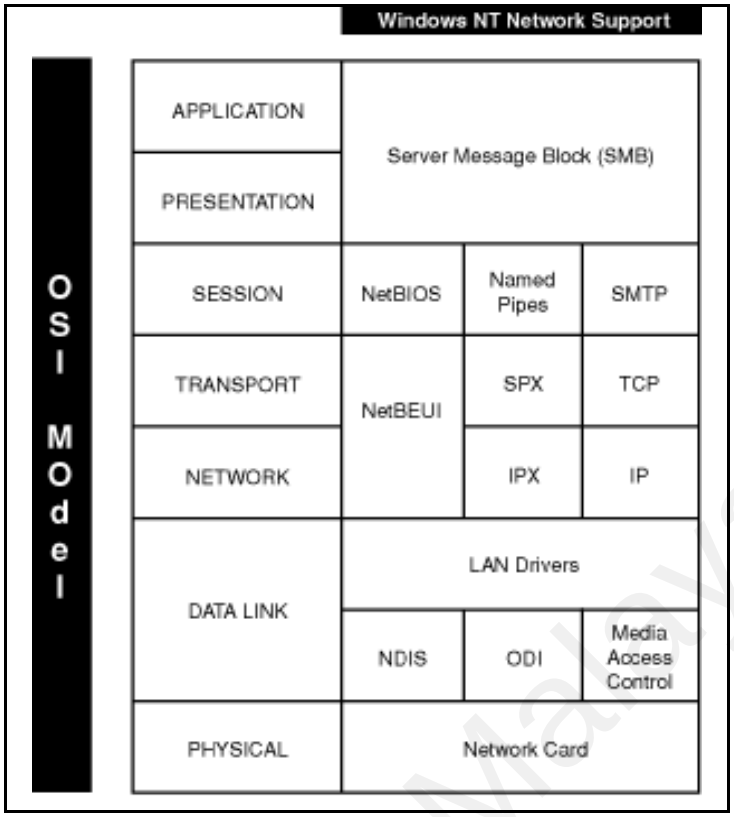


Figure 5.6: Windows NT and OSI (Abstracted from Wolters V.)

Ever-increasing network traffic and expansion of LANs and WANs to accommodate a larger number of domains requires commensurate enhancement of server capabilities, and especially the performance of servers used as domain controllers. Microsoft claims up to double the throughput over 100BaseT networks compared with Windows NT Server 3.51, based on tests conducted by National Software Testing Laboratories (NTSL) (NTServer, 2000).

Basically, in this project, in order to for Domino and its services to run, Windows NT Server 4.0 was used although Windows 2000 was released in the mid of the project development.

5.2.2 GRAT Phases

This section describes how the various phases identified in earlier chapters are implemented. A more detailed description along with the screen capture is presented in Appendix B.

5.2.2.1 Project Repository

Figure 5.7 shows the design and the layout of the Project Repository which lists all project that is currently active. Users can either choose to create a new project by clicking on raised button on the picture or view the project details by double clicking on the project's name itself. Only the Domino Administrator is able to view and see the delete button. The project list is shown by using an applet.

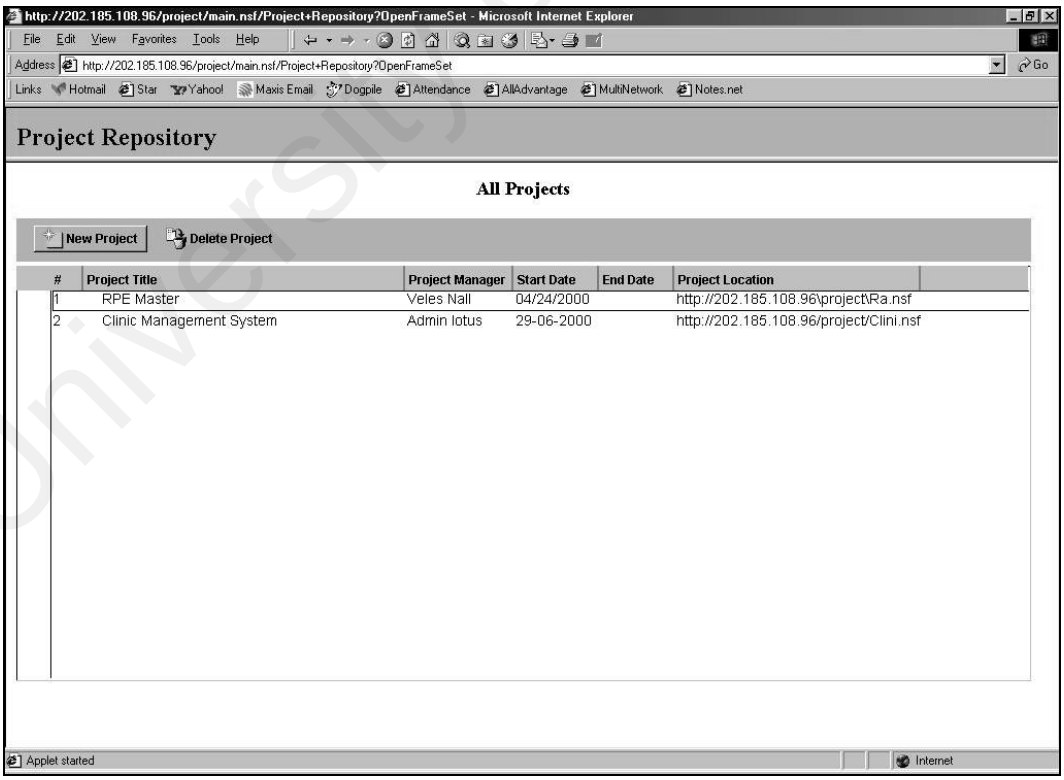


Figure 5.7: Project Repository

5.2.2.2 Domain Understanding

Figure 5.8 shows the list of submitted documents in a applet. Same as Project repository, to submit a new material, simply click a button and to view it, double click on the material's information on the applet.

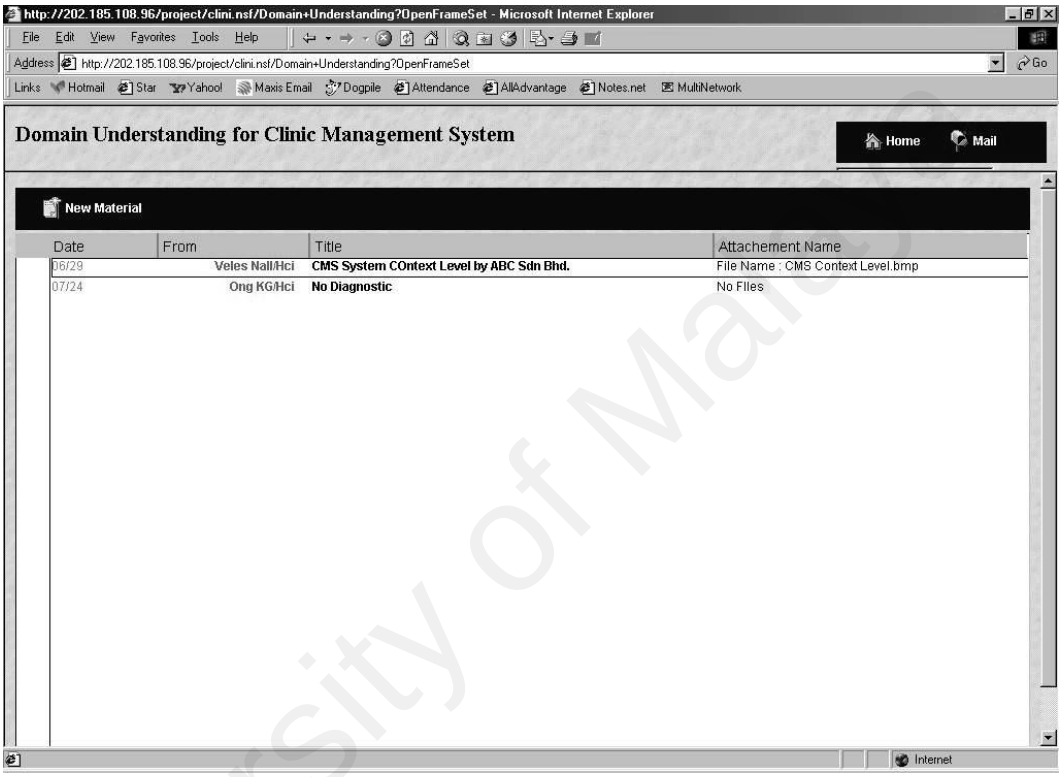


Figure 5.8: Domain Understanding

5.2.2.3 Requirements Collection

Figure 5.9 shows the platform to collect the requirements and to view the requirements submitted. The view is updated every time a requirements is submitted or every 15 seconds, whichever is sooner. The frame below the view is the place where team members are required to type the requirements.

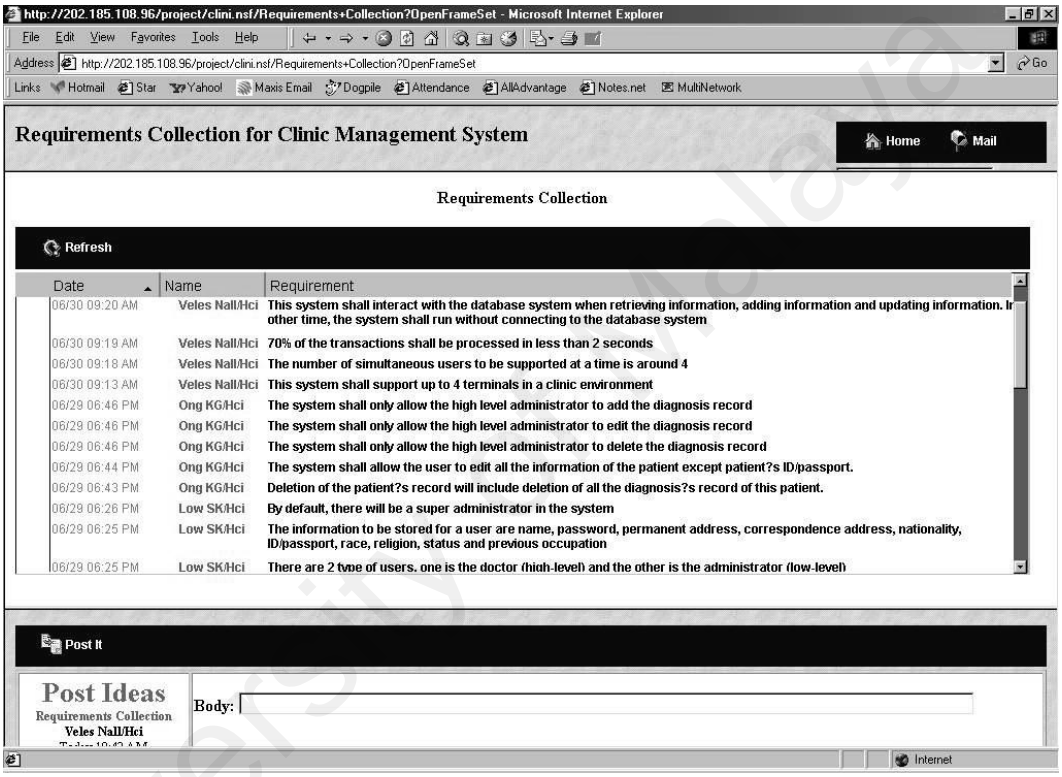


Figure 5.9: Requirements Collection

5.2.2.4 Categories Collection

Figure 5.10 depicts the categories collection phase. The picture shows the list of requirements and categories, which team members think is suitable for the requirements listed. To submit a category, team members have to click a button to add category and submit the category. Again applets are used to show the list of requirements and categories

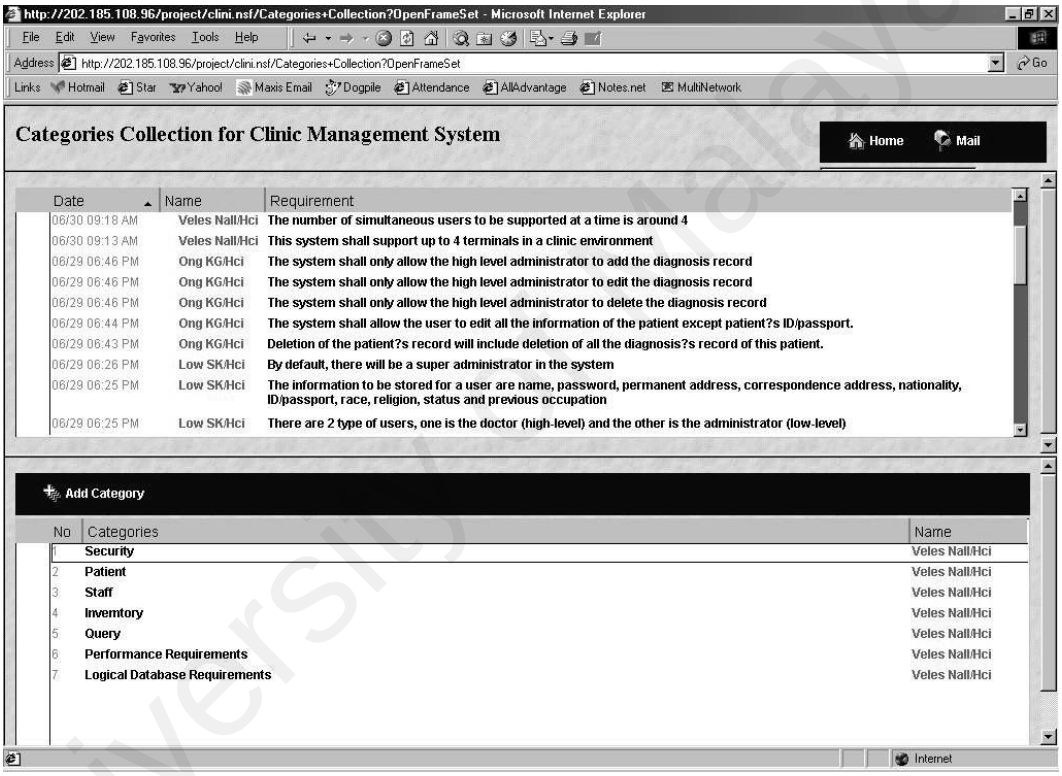


Figure 5.10: Project Repository

5.2.2.5 Classification

Figure 5.11 shows how team members do the classification process. They need to match the appropriate requirements with the category they feel suitable. The categories are listed in a list box. Since tables were used to list the boxes and the list box and since both of number of requirements and categories are not limited, meaning that they are dynamic, JavaScript was used to create the dynamic HTML as seen in the picture.

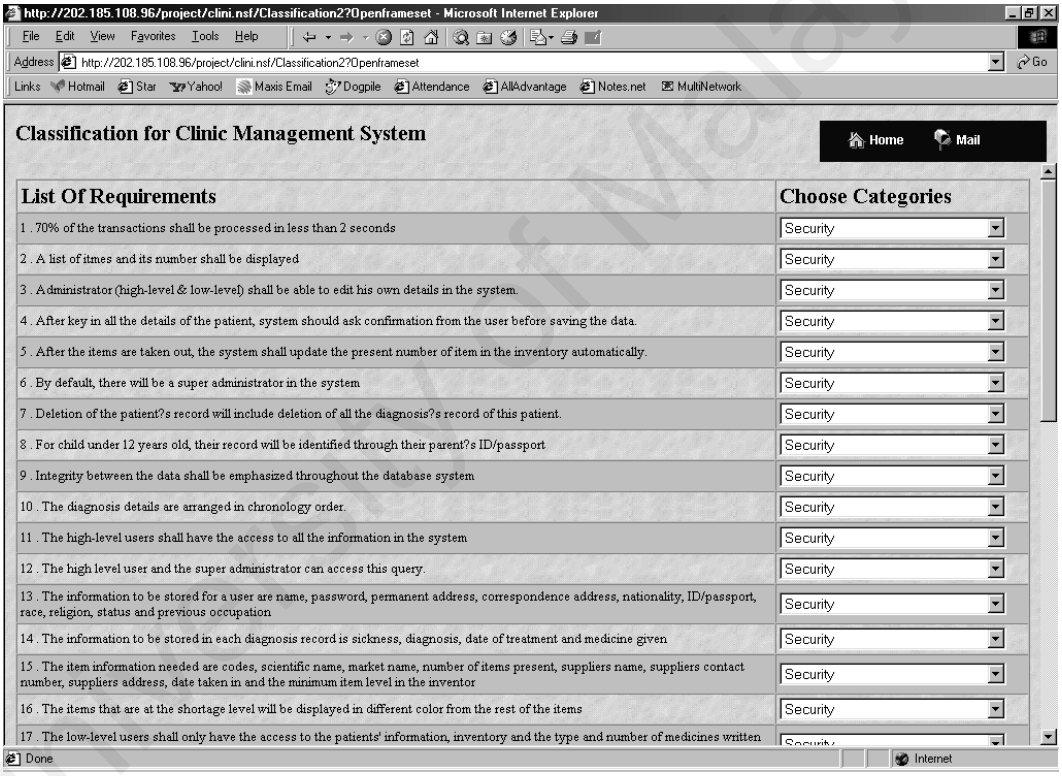


Figure 5.11: Classification

5.2.2.6 Conflict Resolution

The team members are able to view the conflicts and the response to the conflict in a hierarchy manner as shown in Figure 5.11. In this manner, viewers are able to look at the requirements according to the categories and the conflicts according to the requirements and the responds according to the conflicts. The title of the conflicts and responds are shown on the view and for more detailed description of the conflicts and responds, team members have to double-click on the selection.

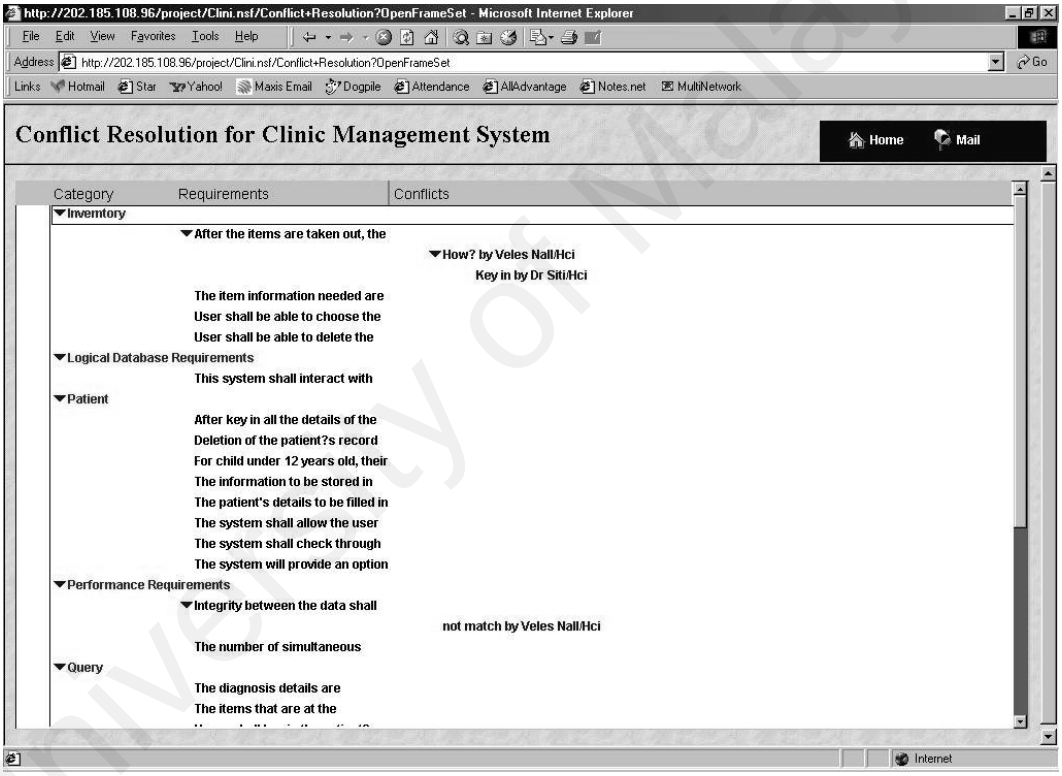


Figure 5.12: Conflict Resolution

5.2.2.7 Prioritization

Figure 5.13 depicts the process of prioritization. The concept of prioritization is same as classification as mentioned before. Team members need to match the requirements with the priority, which are listed in a list box. The scale starts from 1 means the highest priority. Supposing a category has 6 requirements, than the scale are from 1 to 6 with 1 being the highest priority and 6 as the lowest priority.

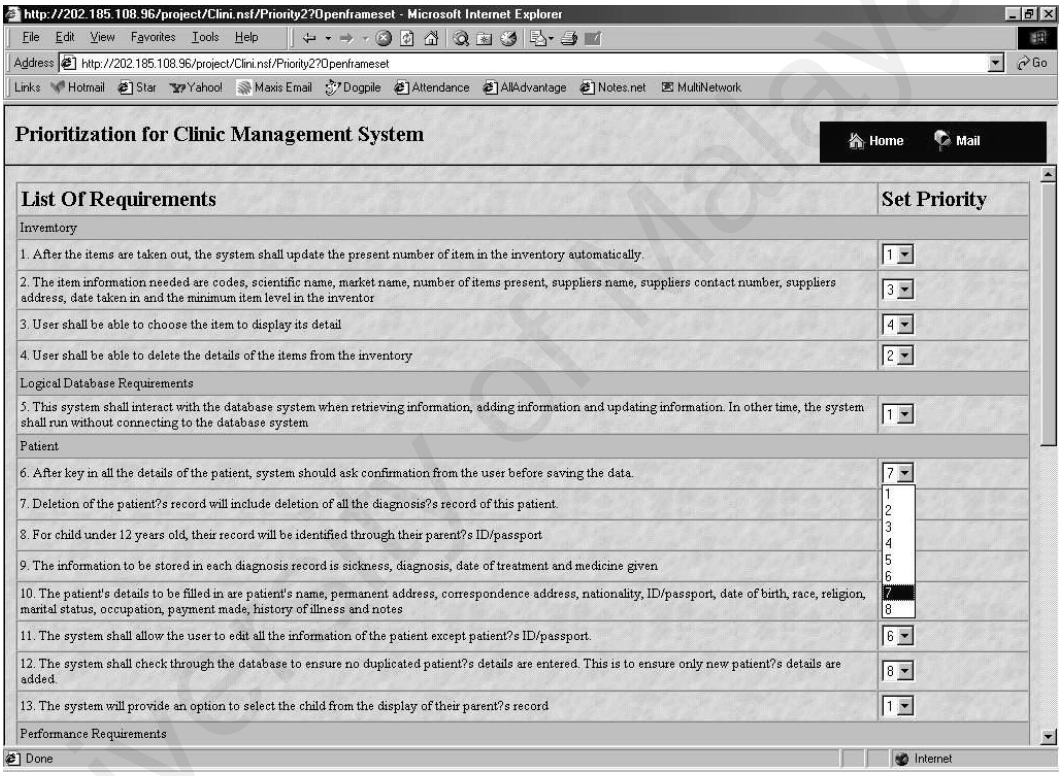


Figure 5.13: Prioritization

5.2.2.8 Requirements Validation

The team members are able to view the invalid characteristics of the requirements in a hierarchy manner as shown in Figure 5.14. In this manner, viewers are able to look at the prioritized requirements according to the categories and the invalidities according to the requirements. If team members feel a requirements is invalid, than he or she have to double-click the requirements and then submit a requirements invalid form.

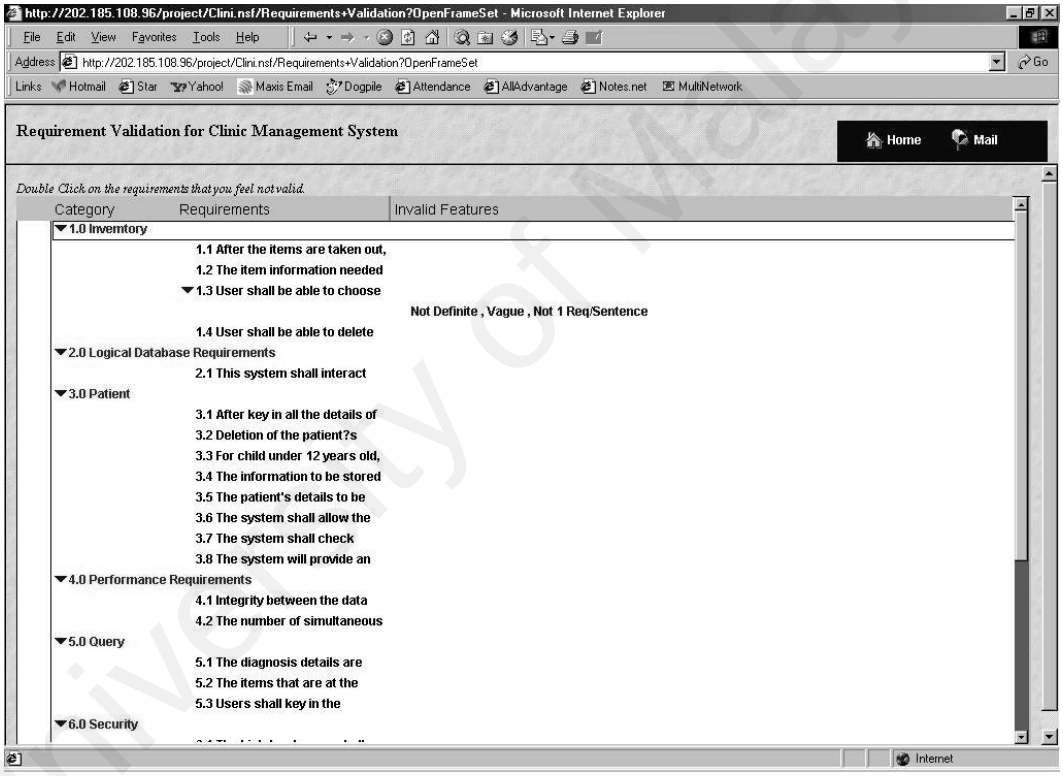


Figure 5.14: Requirements Validation

5.2.2.9 Change Phases

This function is only available to the project manager. The project manager is supposed to control the flow of the requirements analysis process by setting and changing the phases of the process. This is done as shown in Figure 5.15. Project Manager needs to double click on the phase to set the phases.

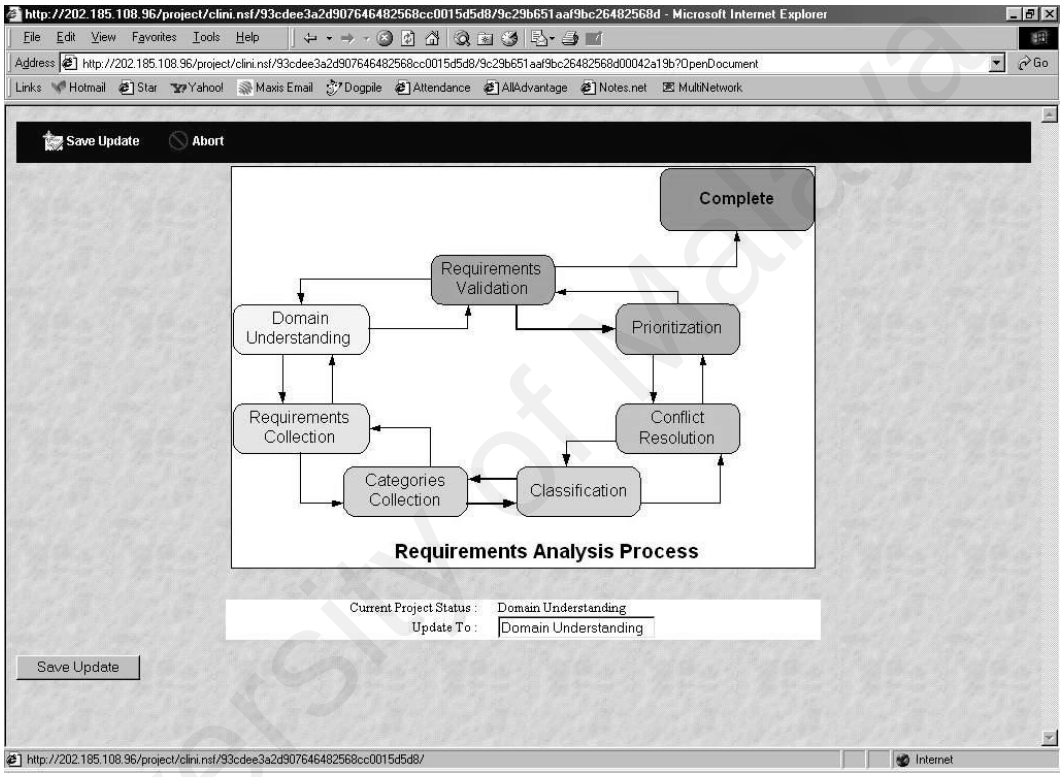


Figure 5.15: Change Phase

5.2.2.10 Activity Scheduling

This service is available to all members. By simply filling and submitting the form as shown in Figure 5.16, the scheduled activities are able to be seen by other members when they login to the project.

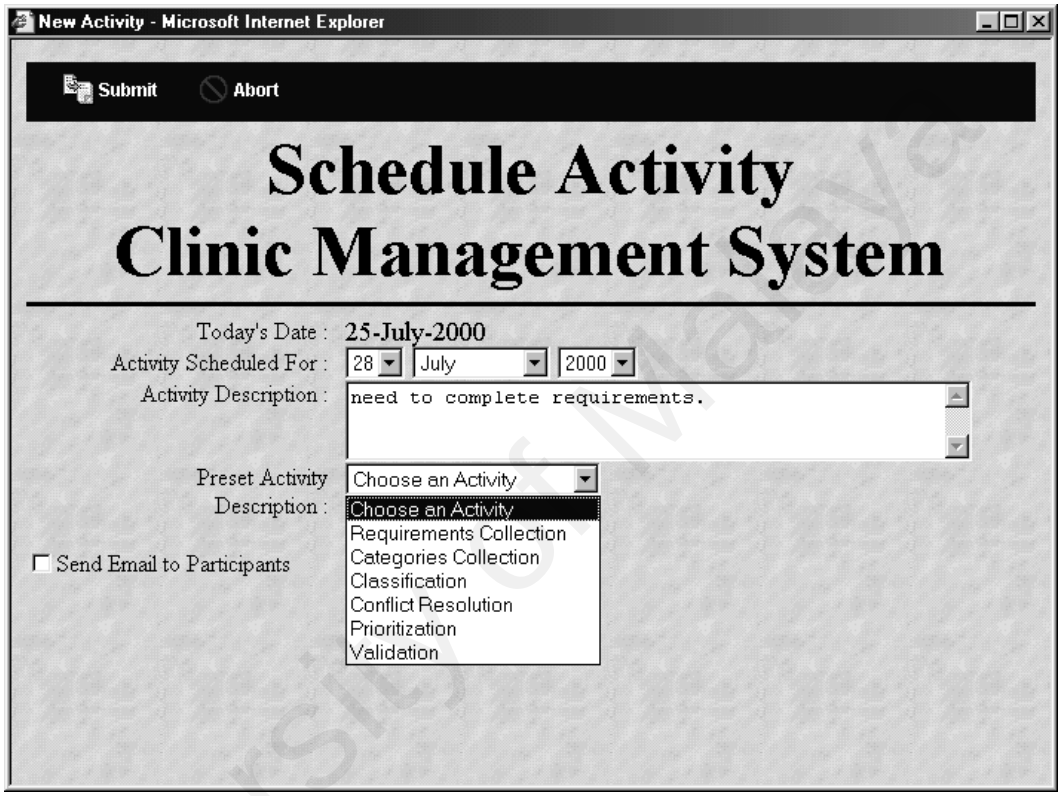


Figure 5.16: Activity Scheduling

5.2.2.11 Completion

At the end of the requirements analysis process, the completed list of requirements for the project are available to users as shown in Figure 5.17.

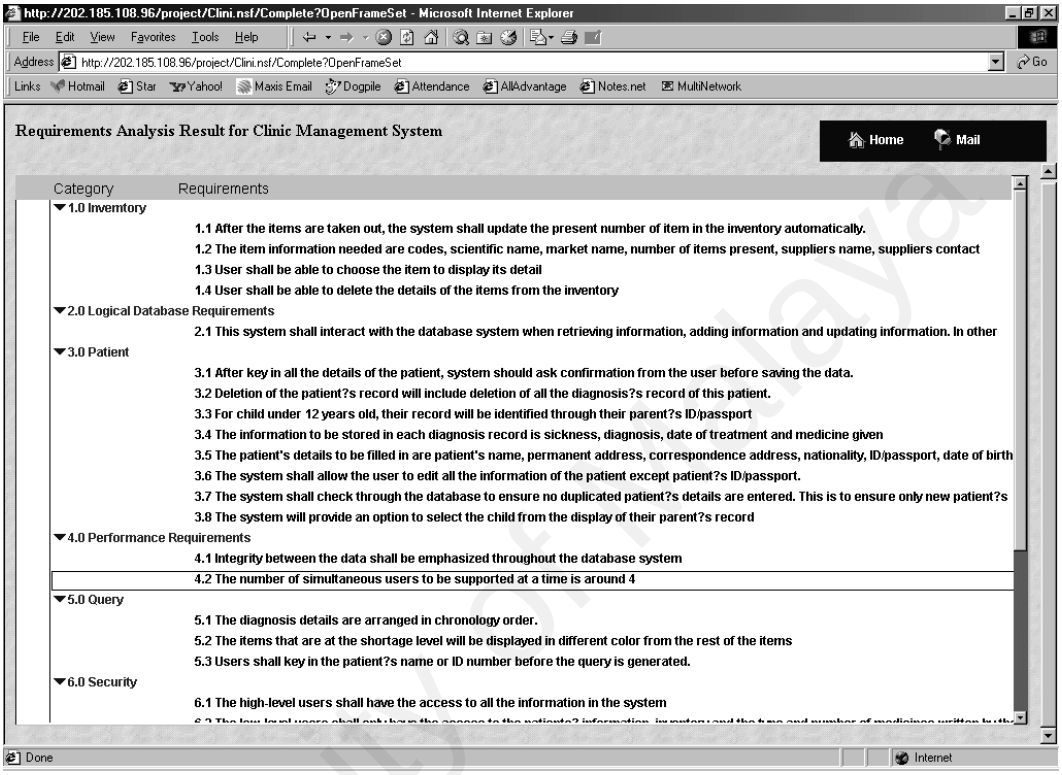


Figure 5.17: Project Completion

5.3 Execution

Having described the platform and the phases of GRAT, this section briefly introduce on how to run the system.

A project manager who wants to use the system has to create the project by submitting a form to the project repository. Once the project manger submits the form, the Domino Administrator is required to allocate the project space on the server. As this cannot be done on the web, the Domino Administrator is required to log in to a Lotus Client machine and allocate the project space and the web site. This can be done at a

click of a button. Once completed, the details of that project are updated and the project location is saved for other users to visit the page.

The project starts with the Domain Understanding phase. This is where team members share their knowledge, experience and materials. All they have to do is submit a form and the materials they submitted are available to others. The materials could range from opinion to web links to files.

Once the project manager feels that this phase is complete, he or she could change the phase by simply clicking at the appropriate phase. Next time the user logs in, it directly updates with the updated phase. Anyone could also post an activity so that others know what is going on that particular day.

The next phase is Requirements Collection. Here all the team members have to do is simply brainstorm the requirements they feel suitable for the project. There would not be any discussion among team members. The requirements list are updated when a requirements is posted or 15 seconds, whichever is sooner.

Following that would be the Categories Collection. The concept is same as requirements collection. The categories are collected. For the ease of users, the lists of requirements are also shown so that categories appropriate to the requirements can be derived.

Next is the Classification. This is a matter of matching the requirements and the categories collected earlier. Once all the requirements have been matched, users can preview and edit the classification. Once satisfied, they can submit. Members who have not submitted are identified and listed.

Moving to the Conflict Resolution phase, here team members can raise any doubts and conflicts they find in the requirements. Those who feel can clarify the conflict can choose to respond to the conflict. At the end of the phase, the Project Manager has to delete the requirements which he or she finds it conflict.

The Prioritization phase is also conceptually same as Classification. The team members are required to match the requirements with the right level of priority. The level of priority starts from 1 representing the highest level of priority and lowest priority level is equivalent to the number of requirements in a category. Each requirement in a category must have a unique value. Once this condition is satisfied, users preview the result and reprioritize. They can choose to submit and as the Classification process, the names of members who have not completed this phase are listed.

Now, the Requirements Validation is a phase, which requires team members to identify requirements that they feel not valid. The identified requirements are shown in a form with different characteristics of invalidity. Team members need to select the appropriate checkboxes and submit the form. Again the project manager justification comes into play. If the Project Manager feels a requirement is not valid based on the invalidity identified by the user, he or she has to delete the requirements.

Finally, once all the process is completed, the Completion phase simply would the final list of requirements.

5.4 Summary

This chapter has described the implementation and the execution of GRAT. The implementation looked into developmental environment of GRAT and the manners of the GRAT phases were implemented. Each phase was briefly described with a general screen capture presented. A more detailed description of the system with all the images is presented in Appendix B. The second part, execution discusses about the flow and the actions required for the GRAT system.

CHAPTER 6: GRAT Evaluation and Results

6.1 Introduction

This chapter presents GRAT evaluation process. Evaluating GRAT's compliant with the objectives stated in Chapter 1 is one way of assuring the validation of the system.

6.2 Pilot Study

The objective of executing this test is to evaluate the usability of the system. As mentioned earlier, the test was carried out two groups ranging from students with little to moderate knowledge about software engineering generally and requirements analysis specifically. The groups have never used any tools to carry out requirements analysis. At the moment, they have been doing it manually by carrying out meetings and appointing a recorder to record the minutes and the outcomes of the meetings. This study might be able to provide some results on the effectiveness of GRAT compared to the manual system but not in contrast to other available tools.

6.3 Participants

The evaluation of GRAT was conducted using two test cases and two different groups. The members of groups were post-graduate students of Faculty of Computer Science and Information Technology (FCSIT), University of Malaya. The study done is explained in this chapter. Members from the industry were unable to test the system due to the lack of time, space and resources for both parties.

According to R.A. Verzi, 80% of system usability problems can be identified by using 4-5 test users (Verzi, 1992). This evaluation involves two volunteer groups. The first group, which consists of 5 members who are currently pursuing their masters degree in computer science. The second group consists of 6 members who are doing their masters degree at the Department of Software Engineering, FCSIT, Universiti Malaya.

The first group undertook a project called e-Dictionary while the second group did a project entitled Room Booking System for FSKTM.

6.4 Experimental Material

The two groups have made use of GRAT to carry out their requirements analysis. The test cases were their mini projects that they were involved. Since all of them were not exposed to such a system, a simple mock-up project was used to train them. The materials for the mock-up project were derived from one of the projects done by the former students of Department of Software Engineering entitled Clinic Management System.

Since GRAT was meant to be used from any terminals connected to the Internet, the hardware specification for the personal computers by the participants varies from a low-end machine to a high-end server. Generally they used a Pentium machine with a minimum of 32MB RAM. Due to the constrains on the some of the Java applets, the participants were advised to used Microsoft Internet Explorer ver5.5 and above with Microsoft Virtual Machine installed, as GRAT was fully tested running on that browser.

Both the groups were trained earlier. The groups were asked to select a project manager. Each group member where given the questionnaire after the project were completed, roughly a week after they were asked to use the GRAT. The questionnaire contains questions specific to the different phases of GRAT and also general questions,

which looks at GRAT services and how it had influenced the team's requirements analysis process.

6.5 Measurements and Results

Based on the questionnaire presented in Appendix D, the results of the survey are divided and presented in the following sub topics.

6.5.1 Participants Background

From the results shown in Table 6.1, the participants are well versed with requirements analysis. However none of them have ever used any tool that supports collaborative activities. Meaning, all of them used to have face-to-face meetings in conducting their requirements analysis for previous projects.

Table 6.1: Summary of participants' background.

Questions	Ave	SD
1. How well did you know requirements analysis process before using the GRAT?1	2.025	0.388
2. Have you ever used any tools that support distributed working particularly in a group environment?2	2	0

1(1:Very good 2:Good 3:Moderate 4:Very little 5:Not at all)

21(1:Yes 2:No)

6.5.2 Ease of Use

Based on data presented in Table 6.2 and graph presented in Figure 6.1, the strong points for GRAT comes from 73 percent of participant able to predict what happens next as shown in the second column. 63 percent in the meantime were happy on the instructions and prompts provided by the system. A high percentage has agreed that the information provided by the system enabled them to understand and act according to the generated information. 7 out of 11 participants are happy on the layout of the options for the system. 91 percent of the participants in the mean time agreed when asked whether GRAT made their tasks straightforward. Slightly less than three-quarter of the participants seem to be happy with the attractiveness of its presentation. When asked whether working with GRAT was satisfying, a high 91 percent on the positive side was recorded.

Analyzing the down side of GRAT, only 3 of the participants felt that it is easy to move from one phase to another. This might be due to less control given to users in changing of phases as project managers make the decision. Less than one tenth of the participants felt in command while using GRAT. GRAT limits users in modifying the analysis process or skipping of a phase. As expected, 73 percent of the participants were definitely unhappy with the speed of the application. Although the speed and capacity of the network and reliability of the server contribute to this, from the application aspect, the usage of applets is one of the main reasons.

Table 6.2: Summary of GRAT's Ease of Use.

Questions	Agree	Undecided	Disagree
It is relatively easy to move from one part of a task to another.	3	7	1
The software always did what I was expecting	8	2	1
The instructions and prompts are helpful	7	1	3
I can understand and act on the information provided by this application.	9	2	0
I feel in command of this application when I am using it.	1	5	5
It is easy to see at a glance what the options are at each stage.	7	3	1
Tasks can be performed in a straightforward manner using this application.	10	0	1
The speed of this application is fast enough	2	1	8
The software has a very attractive presentation.	8	2	1
Working with this application is satisfying	10	0	1

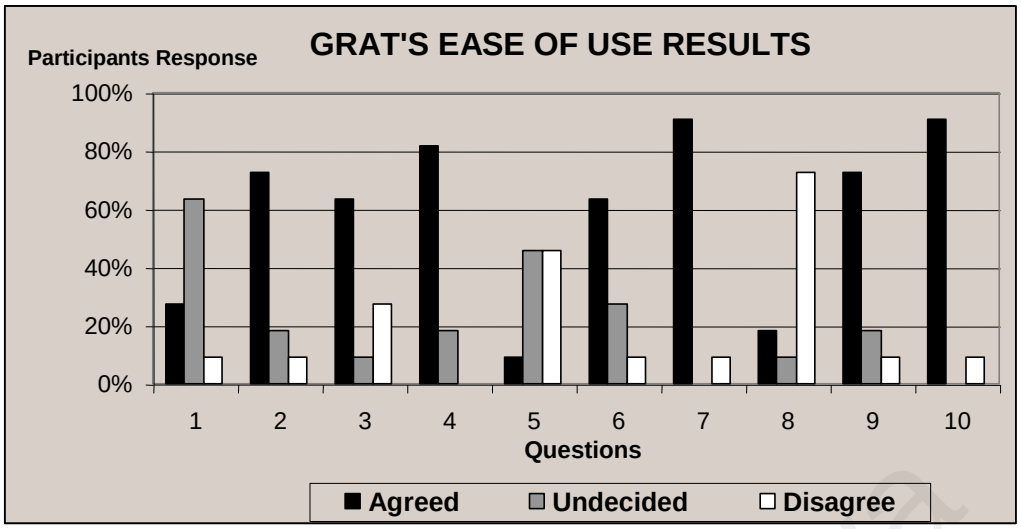


Figure 6.1: GRAT's Ease of Use Results

6.5.3 Components Functionality

The next section was aimed to evaluate each component of GRAT. The questions were constructed in such a manner to extract as much as information regarding how users feel about the facilities provided and the functionality of the components.

The first question was on Project Repository. As shown in Table 6.3, an average score of 2.5 and 1.96 was achieved in a scale between 1 to 5 where 1 denotes most positive response and 5 represents most negative response. The questions were targeted on the workflow implemented in creating the project and the easiness of accessing the project web site. A relatively low standard deviation achieved in these questions further strengthens the fact that the components have achieved its objectives.

Table 6.3: Summary of GRAT's components functionality.

Questions	Ave	SD
1. In Project Repository,		
a. How do you feel about the workflow implemented in creating the project? 1	2.5	0.85
b. Could you gain easy access to the project web site? 2	1.96	0.31
2. In Domain Understanding, how easy did you find it was to share information regarding the project? 1	2.2	0.88
3. In Requirements Collection,		
a. How easy did you find it was to produce requirements in a brainstorming manner regarding the project by using GRAT? 1	1.86	1.76
b. Would you agree that by using GRAT, the requirements were more easily collected than previous methods that you used to apply? 3	2.05	0.86
4. In Classification, did you find it easy to match the requirements with the appropriate classification by using GRAT? 1	3.50	0.31
5. In Conflict Resolution,		
a. How good was it in solving and clarifying doubts and conflicts? 4	3.66	1.59
b. How did you find the hierarchy listing of the doubts and the clarifications for the doubts or conflicts? 5	2.83	1.63
6. In Prioritization,		
a. Did you find that all necessary information was included that helped you to match the requirements with the appropriate priority? 2	2.62	0.97
b. How did the results of the prioritization effective? 5	3.10	0.43
7. In Requirements Validation,		
a. How good would you rate the requirements invalid form? 4	3.89	1.55
b. Did it effectively help you in identifying the invalid requirements? 5	3.50	0.84
8. In Activity Scheduling,		
a. Was electronic mail (e-mail) notifications helpful? 6	1.36	0.34
b. How effective is the scheduling in the tracking the project	1.69	0.31

development?5		
c. How useful were the utilities provided? 6	3.45	0.23
9. How would you rate the overall facilities provided in GRAT?		
a. Project Repository 7	1.89	0.64
b. Domain Understanding 7	2.23	0.31
c. Requirements Collection 7	1.97	0.87
d. Categories Collection 7	3.64	0.31
e. Classification 7	3.21	0.97
f. Conflict Resolution 7	3.06	0.34
g. Prioritization 7	3.45	1.46
h. Requirements Validation 7	2.65	1.01
i. Activity Scheduling 7	1.49	0.21

1 (1:Very easy 2:Easy 3:Average 4:Difficult 5:Very difficult)

2 (1:Always 2:Most of the time 3:Moderate 4:Very little 5:Never)

3 (1:Totally agree 2:Agree 3:Average 4:Disagree 5:Totally disagree)

4 (1:Very good 2:Good 3:Average 4:Bad 5:Very bad)

5 (1:Totally Effective 2:Effective 3:Average 4:Ineffective 5:Totally Ineffective)

6 (1:Totally helpful 2:Helpful 3:Average 4:Unhelpful 5:Totally unhelpful)

7 (1:Totally good 2:Fairly good 3:Satisfactory 4:Fairly Poor 5:Totally Poor)

Similarly, a high acceptance result was achieved for GRAT in Domain Understanding when participants were asked on the easiness of sharing information at this phase.

For Requirements Collection phase, the effectiveness of this component is clearly shown by the result. An average of 1.86 and 2.05 were achieved when

participants were asked about the easiness to produce requirements and collect them compared to their previous methods that they have used. However, the high standard deviation in the former shows the opinions of the participants varies from one another although they seem to agree strongly for the latter. This can be seen from the relatively lower standard deviation.

In Classification, the score of 3.5, which slightly on the downside than the average was, recorded when participants were asked on their opinion in matching the requirements with the appropriate classification using GRAT. This point was even more strengthening with the 0.31 standard deviation achieved.

As the questions move on to the next phase, Conflict Resolution, the effectiveness in solving and clarifying doubts seems to be questionable as the average score computed is only 3.66 with standard deviation of 1.59. However, most of the participants seem to be happy with the hierarchy listings as an average score of 2.83 was achieved also with a high 1.63 in standard deviation.

Looking at Table 6.3, most of the participants were merely satisfied on the functionality provided in Prioritization as the low standard deviation proves this point. 2.62 average points was recorded when participants were asked about how effective could the prioritization process executed with the information provided. Meanwhile a 3.10 average was computed when asked about the effectiveness of the prioritization process.

Moving on to requirements validation, a rather lower than average scores were achieved. Participants were rather not happy with the requirements invalid form as it might be due to the standardized form with no free form of critics could be addressed or voiced. A rather similar reason attributes to the lower average score in effectively identifying the invalid requirements.

Activity Scheduling recorded a very strong positive note as the functionality of the components managed to impress the participants. The email facility and the effectiveness in tracking the progress of the project achieved one of the lowest averages and with the support of small standard deviation mean the effectiveness and the efficiency of the component is rather high.

Referring to Table 6.3 on how the participants rate the overall facilities in GRAT, Figure 6.2 was derived. It reveals the average and the standard deviation of GRAT respective to the components or phases. One look at the graph, a summary is derived that most of the participants felt that the phases came with good or above average facilities or components. However, Categories Collection and Prioritization was rather unpopular with the participants.

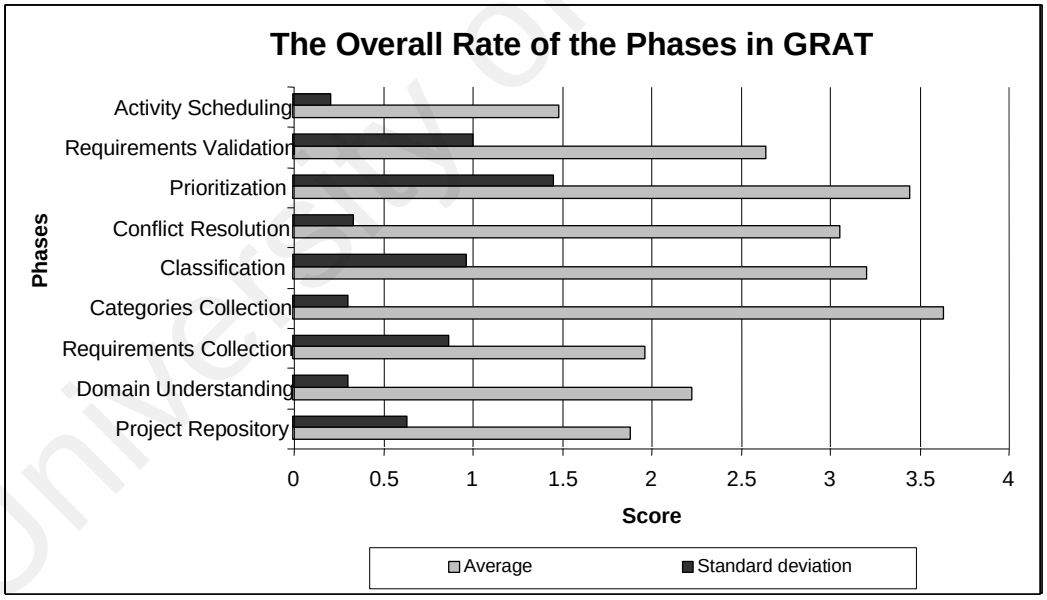


Figure 6.2: Rate of the Overall facilities in GRAT

6.5.4 Achievement of Objective

The fourth part of the questionnaire was aimed to get users perspective on GRAT, whether or not it has achieved its objective of serving as a groupware platform

in executing the requirements analysis process. Generally, most of the participants were satisfied when they were inquired to rate the set of requirements produced. An average of 2.14 was recorded with a low 0.46 on standard deviation. The results are presented in Table 6.4.

Table 6.4: Summary of GRAT's achievement of objectives.

Questions	Ave	SD
1. How would you rate the set of requirements produced by using GRAT? 1	2.14	0.46
2. GRAT is a groupware application that supports and augments the activities of groupware. How would you rate GRAT for the following characteristics?		
a. Arranging time to work together or for meetings 2	2.48	0.23
b. Sharing information among group members 2	2.94	0.97
c. Having discussions and making decisions 2	3.84	0.13
d. Participation of group members throughout the project 2	2.49	0.51
e. Awareness of self involvement or participation during the project 2	2.65	0.75
f. Communication between you and other team members throughout the requirements analysis process 2	3.98	0.33

1 (1:Completely satisfied 2:Satisfied 3:Average 4:Dissatisfied 5:Completely dissatisfied)

2 (1:Totally good 2:Fairly good 3:Satisfactory 4:Fairly Poor 5:Totally Poor)

The next question served well in analyzing GRAT's groupware architecture. Figure 6.3 shows the average score and the calculated standard deviation based on the questions asked. Most of the score was between fairly good and satisfactory. GRAT has done fairly well in participation issues, either in self-awareness in contributing to the project or group members' involvement or even in scheduling for events to work

together. However, it was only rated satisfactory in communicating with fellow group members thus reducing the effect of discussions and decision.

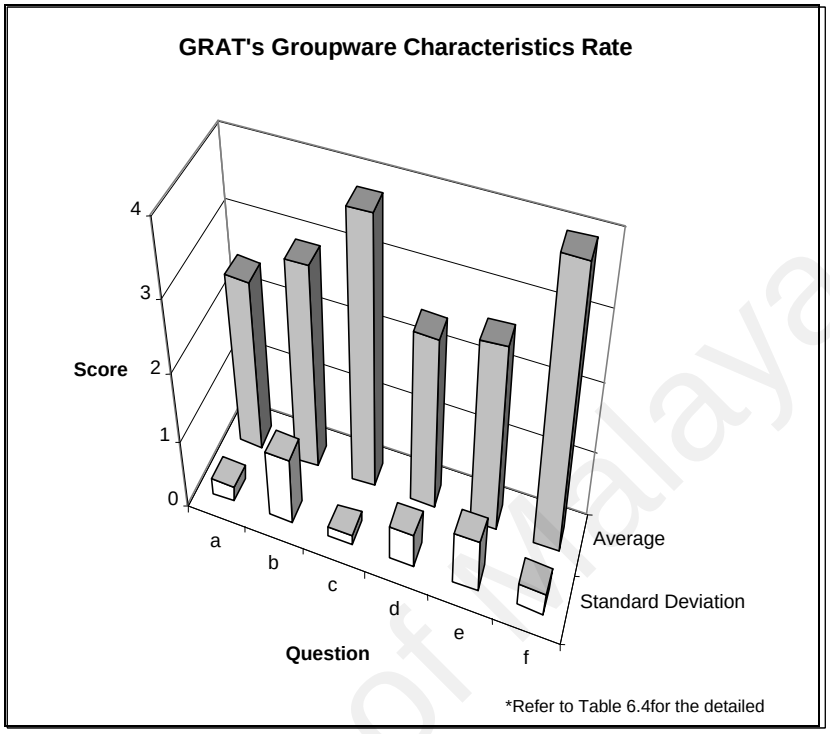


Figure 6.3: Average and standard deviation score for Question 2.

6.5.5 Enhancements of GRAT

The final part of the questionnaire contains two free form comment questions. They were asked on other enhancements that could be implemented on GRAT that improves the effectiveness of the system and also to voice out further opinions about the system.

For the first question, most of the comments were on providing real-time communication facility. Among the given examples were on-line chat, video conferencing, spelling and grammar checking utility, ability to save the work and resume back from where it was left for phases like classification and prioritization.

Other than the real time communication, some also stated that a drawing tool would be useful in illustrate diagrams or models.

The second question was not populated by the participants. Many of them re-mentioned the reasons answered in the earlier question. Some mentioned to include some sort of Microsoft Project kind of layout for project tracking. Some of the participants also requested that the GRAT to be expanded to cover the whole of requirements engineering and software development life cycle.

6.6 Summary

This chapter has evaluated GRAT against its requirements and usability overall. Pilot study regarding the evaluation process is also included. Qualitative measurements were derived to measure the level of satisfaction of GRAT. The results are presented in this chapter.

CHAPTER 7: Conclusion

7.1 Research Summary

A groupware architecture or application should provide an easily accessible, widespread platform for gathering and sharing information and for capturing ideas. Moreover, it should support person-to-person collaboration. Getting products to market as soon as possible has motivated in depth studies on collaboration technology in addition to distributed workforces and information overload faced by many organizations. The technology is used in communicating, cooperating, coordinating, solving problems, competing and negotiating.

Requirements analysis is the most important step in system development. As the most critical task in software engineering, it involves activities such as gathering of requirements, identifying missing requirements, related requirements being categorized or classified and finally requirements examining and evaluating for conflicts or inconsistencies. In other words, software engineers are able to specify the functions and performances of the system being developed. In addition, a more complex requirements analysis helps in recognizing problems, evaluating and synthesizing solution.

As a final product, requirement analysis must be able to develop clear, complete, agreed upon and feasible requirements for a product. The acceptability of the system after it has been delivered very much depends on how well it meets the customer needs.

This research has analyzed, designed and implemented an architecture based on groupware technology that provides as a platform for requirements analysis. Although there are a number of requirements analysis tool in the market, but none is built based on groupware architecture. The review has helped in capturing a feasible requirements analysis methodology and the architecture for the groupware support. Meaning, the

features of the outgrowth of this thesis, GRAT, have been successful in capturing the main features and to reveal the missed ones in other similar tools. The framework of GRAT was documented and presented in Chapter 3. So, based on an in-depth study of groupware, GRAT was conceptualized and implemented.

Based on these understanding from the research of both the domains, groupware and requirements analysis, GRAT's functionality was developed. An extensive analysis and design was executed and the results are presented in Chapter 4. From here, the research moved on to implementation where various tools and developmental environment that were used was discussed. GRAT was developed and discussed in Chapter 5.

To know how GRAT fare, a validation was carried out by a group of post-graduate students. The results, findings and measurements are presented in Chapter 6. The measures revealed some limitations in GRAT although in overall perspective GRAT has managed to prove its usability and achieving its objectives.

7.2 Contribution

GRAT has contributed in two different areas or domain if seen at a wider scope.

- A web-based application was introduced. It has managed to implement the proposed model of the conceptualized version of GRAT. Running on the web means the space constrains has been eliminated although current bandwidth might slightly degrade the speed or the performance of GRAT.

- It also helped to understand the importance of groupware support for collaborative activities. The results from questionnaire reveal the exposure

for such a tool is needed. The testers were impressed on how the tool helped to track the requirements and issues.

7.3 Future Work

The research described in this thesis can be extended in several ways. Among them are,

- At the moment, GRAT could only support requirements analysis process. It would be better is the system could cover the requirements engineering process where the actually requirements documentation could be extracted.

- In chapter 2, it is said that GRAT was build based on requirements analysis proposed by Ian Sommerville (1996). To make GRAT more beneficial, it should be able to support other requirements analysis methodologies, giving the choice to the users.

- An extra functionality that was proposed during the evaluation period was to include video conferencing tools to carry out live meetings and decisions. Also a synchronous sort of chat facility to be provided. However, this is rather impossible to do without the use a third party application, as Lotus Notes does not support synchronous activities.

- The system should also provide a checklist kind of service where participants would be able to keep track of the project.

- GRAT should also be able to support all project management activities which mean managing of resources, tracking the progress of project, keeping track of the project against its objectives and scheduling, which was incorporated in the system.

- The evaluation of GRAT. Although the validation and testing provided initial evidence about the effectiveness and the efficiency of the introduced system, it still lacks of any explicit scientific proof. This is due to the reason that the evaluators are only students and not industrial experts who have used some requirements analysis techniques in developing larger scale of projects in their organizations.

Last but not least, groupware and requirements analysis are two domains, which need to be explored to its maximum potential and benefit in order to integrate both the concepts.

Bibliography

Aksit, M., and Bergmans, L., 1992. Obstacles in Object Oriented Software Development, in, *Proceedings on Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'92), Seventh Annual Conference*, 18-22 October 1992, Vancouver, Canada, 18-22 October 1992, pp.341-358,

Antón, A.I., 1996. Goal Based Requirements Analysis, in, *Proceedings of the Second International Conference on Requirements Engineering*, Colorado Springs, Colorado, 15-18 April 1996, pp. 117-119.

Antón, A.I., Liang, E., and Rodenstein, R.A., 1996. A Web-Based Requirements Analysis Tool, in, *Proceedings of the 5th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Stanford, California, June 19-21 1996, pp. 238 –243.

Antón, A.I., McCracken ,W.M., and Potts, C., 1994. Goal Decomposition and Scenario Analysis in Business Process Reengineering, Advanced Information System Engineering, in, *Proceedings of 6th International Conference CAiSE 1994*, Utrecht, The Netherlands, 6-10 June 1994, pp. 94-104.

Baecker, R.M., 1993. Readings in groupware and computer-supported cooperative work: *Assisting human-human collaboration*. San Francisco, CA. Morgan Kaufmann Publishers.

Basili, V.R., and Periccone, B.T., 1984. Software errors and complexity: An empirical investigation. *Communications of the ACM.* Vol. 27, No. 1.

Beizer, B., 1990. *Software Testing Techniques, 2nd edition*. New York. Von Nostrand.

Bock, G.E., 1995. *Designing Groupware : A guidebook for designers, implementers and users*. USA, McGraw Hill Inc.

Boehm, B. W., 1988. A Spiral Model of Software Development and Enhancement, *IEEE-CS Computer*, Vol. 21, No. 5, May 1988, pp. 61-72.

Boehm, B.W., 1981. *Software Engineering Economics*. Englewood Cliffs, Prentice Hall.

Boehm, B.W., and Bose, P., 1994. A Collaborative Spiral Software Process Model Based on Theory W, in, *Proceedings of the Third International Conference on the Software Process, Applying the Software Process*, Vancouver, October 1994.

Boehm, B.W., and Pappacio, C., 1988. Understanding and controlling software costs. *IEEE Transactions on Software Engineering*. Vol. 14, No. 10, October 1988, pg 1470.

Brinck, T., 1998. http://www.usabilityfirst.com/groupware/intro.html. Last updated: 1998. Diamond Bullet Design.

Brooks, P., and Frederick, J., 1987. No silver bullet: essence and accidents of software engineering. *Computer* . 15(1).

Burch, B., 1999. Domino R5 Technical Overview, in, *Iris Today*, Iris Technologies Inc.

Carlshamre, P., and Karlson, J., 1996. A Usability Approach to Requirements Engineering, *Proceedings of the Second International Conference on Requirements Engineering*, Colorado Springs, Colorado, 15-18 April 1996, pp. 147-149

Champeaux, D., 1991. Object Oriented Analysis and Top-Down Software Development. *European Conference on Object-Oriented Programming 1991 Proceedings*, Vol. 512, pp. 360-376.

Champeaux, D., and Faure, P., 1992. A Comparative Study of Object Oriented Analysis Methods. *Journal of Object Oriented Programming*, March/April 1992, Vol. 5, No. 1, pp. 21-33.

Christel, G.C., and Kyo, C.K., 1992. Issues in Requirements Elicitation, in, *Technical Report Software Engineering Institute, Carnegie Mellon University,* CMU/SEI-92-TR-12, September 1992.

Ciborra, C., 1993. *Teams, Markets and Systems: Business Innovation and Information Technology*. Cambridge, Cambridge University Press.

Clark, K.B. and Wheelwright, S.C., 1993. *Managing New Product and Process Development. Text and Cases.* New York, Free Press.

Coleman, D., 1997. *Groupware: Collaborative Strategies for Corporate LANs and Intranets*. Englewood Cliffs, Prentice Hall.

Collaborative Strategies, 1996. http://www.collaborate.com. Last Updated : March 2000. Collaborative Strategies LLC.

Curtis, B.H., et al., 1988. A field study of the Software Design Process for Large Teams. *Communication in ACM* , Vol. 31, No 11, pp.1268-1270.

Dardenne, A., van Lamsweerde, A., and Fickas, S., 1993. Goal-directed Requirements Acquisition, *Science of Computer Programming*, 20(1-2), April 1993, pp. 3-50.

Davis, A.M., 1993. *Software Requirements, Objects, Functions and States*. Englewood Cliffs, Prentice Hall.

DeMarco, T., 1996. The Role of Software Development Methodologies: Past, Current and Future, in, Proceedings on 18th International Conference on Software Engineering, Berlin, Germany, IEEE Computer Society, 25-29 March 1996, pp. 2-4.

Dennis, A.R. and Wheeler, B.C., 1997. Groupware and the Internet: charting a new world, in, *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, Maui, Hawaii, Vol. 4, 1997, pp. 287 –296.

Douglas, T., 1970). *A Decade of Small Group Theory, 1960-1970*. London, Bookstall Publications,

Drake, J.M., 1993. Approach and Case Study of Requirements Analysis Where end users take an Active Role. *Proceedings of 15th International Conference on Software Engineering*.

Ellis, C.A., et al., 1991.Groupware – Some Issues and Experiences. *Communications of the ACM*, Vol. 34, No. 1, 38-58., January 1991

Facilitate.com, 2000. http://www.facilitate.com/. Last Updated: 26 February 2000, Facilitate.com Inc.

Greenbaum, J., and Kyng, M., 1991. *Design at Work: Cooperative Design of Computer System*. New Jersey, Lawrence Erlbaum Associates.

GroupSystems.com, 2000. http://www.ventana.com/. Last Updated: 7 May 2000. GroupSystems.com.

Hiltz, S.R. and Turoff, M., 1978. *The Network Nation: Human Communication via the Computer*, Reading MA, Addison Wesley.

Hiltz, S.R., 1984. *Online Communities: A Case Study of the Office of the Future*, Norwood, NJ, Ablex.

Honeycutt, J., 1997. *Special Edition Using the Internet, Fourth Edition*, New York, US, Que.

Horowitz, E., Lee, J.H. and Lee, J.S. 1999. WinWin: A System For Negotiating Requirements, in, *Proceedings of the 1999 International Conference on Software Engineering*, Los Angeles, USA, May 16-22, 1999, pp. 646-649.

Hsia, P., Davis, A. and Kung, D., 1993. Status Report: Requirements Engineering. *IEEE Software*, Vol. 10, No. 6.

IEEE, 1990. Dorfman, M. and Thayer, R.H. *Standard, guidelines, and examples on system and software Requirements Engineering.* Los Alamitos, IEEE Computer Society Press.

JavaScript, 1999. *Client-Side JavaScript Guide*, Netscape Communications Corporation.

Johansen, R.. 1988. Groupware: *Computer support for business teams*. New York, The Free Press (Macmillan), pp. 44.

Judy, H., 1991. *Joint Application Design: The Group Session Approach to Systems Design*, Englewood Cliffs, N. J., Prentice-Hall, Inc.

Kaiya, H., and Saeki, M., 1993. A Supporting Tool for Face-to-Face Meetings to Develop Software Specifications. *IEICE Technical Report*, KBSE93-13, pp. 9-10.

Kawakita, J., 1982. The Original KJ Method. Tokyo, Kawakita Research Institute.

Kraemer, K.L. and King, J.L.. 1988. Computer-Based Systems for Cooperative Work and Group Decision Making. *ACM Computing Surveys*, Vol. 20 No. 2, June 1988, pp. 115-146.

Kramer, J. et al., 1988. Tool Support for Requirement Analysis. *Software Engineering Journal*. May 1988, pp. 86-96.

Krasner, H., McInroy, J. and Walz, D.B.. 1991. Groupware research and technology issues with application to software process management. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21 No. 4, July-August 1991 , pp. 704 –712.

Laranjeira, L.A., 1990. Software Size Estimation of Object Oriented Systems. *IEEE Transactions on Software Engineering*, Vol. 16, No. 5, May 1990, pp. 510-522.

Leite, S. P., 1987. A Survey on Requirements Analysis. *Advanced Software Engineering Project Technical Report RTP-071*, University of California at Irvine, Department of Information and Computer Science.

Lewis, J.A., et al., 1992. On the Relationship between the Object Oriented Paradigm and Software Reuse: An Empirical Investigation. *Journal of Object Oriented Programming*, Vol. 5, No. 4, July/August 1992, pp. 35-41.

Li, M., 1996. A Cooperative Solving Model Supporting Users-Oriented Requirements Analysis, in, *IEEE International Conference on Systems, Man and Cybernetics*, Beijing, China, 14-17 October 1996.

Lotus, 2000. http://www.notes.net/, Last Updated: 10 May 2000, Iris Associates Inc.

Lubars, M., et al., 1993. A review of the state of the practice in Requirements Modeling. *Proceedings Requirements Engineering 1993: International Symposium Requirements Engineering*. IEEE Computer Society Press.

Lyytinen, K., and Hirschheim, R., 1987. *Information System Failures – a survey and classification of the empirical literature*. Oxford University Press. Vol 4.

Macaulay, L., 1996. *Requirements Engineering (Applied Computing)*. Great Britain, Springer-Verlag London, Ltd.

Macaulay, L., 1997. *The Role of the Facilitator in Distributed Teamwork*, PhD Thesis, University of Manchester Institute of Science and Technology.

Marciniak, J.J., 1994. *Encyclopedia of Software Engineering*. New York**,** John Wiley and Sons, Inc.

Marsh, S., 1991. *Facilitating and training in Quality Function Deployment*. Methuen, Massachusetts, GOAL/QPC.

McLaughlin, F., 1989. ICSE 11 prompts engineer to reflect on yesterday look to tomorrow. *Computer*, Vol. 22 No. 7, July 1989, pp. 110-111.

Moore, S., 1994. Evolution of Internet, in, *Electro International 1994*, Boston MA, 10-12 May 1994, pp. 264-266.

Nielsen, J., 1993. Usability Engineering. Cambridge, AP Professional.

NTServer, 2000, http://www.microsoft.com/ntserver/, Last Update: 29 June 2000, Microsoft Corporation.

Nunamaker, F.J., et al., 1991. Electronic Meeting Systems To Support Group Work. *Communications of the ACM (CACM)*, Vol. 34 No. 7, August 1991, pp. 40-61.

Orlikowski, W.J., Yates, J.A., Okamura, K. and Fujimoto, M., 1995. Shaping Electronic Communication: The Metastructuring of Technology in the context of Use. *Organization Science*. Vol 6**.**

Ow, S.H., 1998. *Manuals on the Writing of Specifications and Review Checklist*, Petaling Jaya, Malaysia, Sejana Publication.

Pfleeger, S.L., 1998. *Software Engineering Theory and Practice*. Upper Saddle River, Prentice Hall.

Reggett, D. (ed), et al.,1999. *HTML 4.01 Specifications*, 1997-1999, WorldWideWeb Consortium (W3C).

Schach, S.R., 1999. *Classical and Object-Oriented Software Engineering with UML and Java, Fourth Edition*, New York, McGraw-Hill,

Sodhi, J., 1992. *Software Requirements Analysis and Specifications*. New York, McGraw Hill Inc.

Sommerville, I., 1996. *Software Engineering Fifth Edition.* Essex , Addison Wesley,

Sommerville, I., and Rodden, T., 1994. Requirements Engineering for Cooperative Systems. Lancaster University. Research report : CSCW/1/1994.

Sun, 2000. http://java.sun.com , Last Updated : 20 July 2000. Sun Microsystems Inc.

Sridhar, R. et al., 1994. Lecture Notes on Requirements Elicitation. *Software Engineering Institute.* CMU/SEI-94-EM-10.

Takahashi K., et al., 1996. Hypermedia Support for Collaboration in Requirements Analysis, in, *Proceedings of the Second International Conference on Requirements Engineering*, Colorado Springs, Colorado, 15-18 April 1996, pp. 31-40

Takashashi, K., and Yamamoto, S., 1994. An Analysis of Traceability in Requirements Documents, *IEICE Transactions on Information and Systems*, Vol. E78-D No 4, pp. 394-396.

Takeda, N., et.al., 1993. Requirement analysis by the KJ editor, in, *Proceedings of IEEE International Symposium on Requirements Engineering (RE'93)*, San Diego, USA, 4-6 January 1993, pp. 98-101.

Vallee, J., et al., 1978. *Group Communication through Computers, Social, Managerial and Economic Issues*, Vol. 4, Menlo Park, CA Institute for the Future.

Vanwelkenhuysen, J., 1996. Quality Requirements Analysis in Customer-Centered Software Development, in, *Proceedings of the Second International Conference on Requirements Engineering*, Colorado Springs, Colorado, 15-18 April 1996, pp. 117-119.

Verzi. R.A., 1992, Refining the test phase of usability evaluation: How many subjects is enough. *Human Factors*. 34(4): 457-468, 1992

Viller, S.A., 1990. *Computer Support for Group Facilitators: An investigation*, M.Sc. Thesis, University of Manchester.

Weiser, M., 1997. Software Engineering that Matters to People, in, *Proceedings of the 19th International Conference on Software Engineering*, Massachusetts, USA, ACM Press, 17-23 May 1997, pp. 589

Weber, J., 1997. *Special Edition Using Java 1.1, 3rd Edition*, New Jersey, US, Que.

Wetherbe, J., 1984. *Systems Analysis & Design: Traditional, Structured, and Advanced Concepts and Techniques*. St. Paul, Minnesota, West Publishing.

Wilkie, G., 1993. *Object-Oriented Software Engineering*. Addison Wesley

Woolley, D.R., 1996. Conferencing on the web, http://freenet.msp.mn.us/people/drwool/webconf.html. Last updated: , .

Yu, E. and Mylopoulos, J., 1994. Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering, in, Proceedings of 27th Hawaii International Conference on System Sciences, Maui, Hawaii, Vol. IV, January 4-7 1994, pp. 234- 243.