Requirement elicitation for environmental life cycle processes

Mariam Y. Burmawalla
New Jersey Institute of Technology
Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen
The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.
ABSTRACT

To develop a user-satisfying product the most essential ingredient is to understand user’s needs and expectations from the system. Lack of communication between the user and the developer results in an unsatisfying output from the user’s point of view. Thus proper understanding between the stakeholder and the developer is the most important requirement during the development of any system.

Requirement elicitation process is one such means of expressing wants and requirements from the system by both the parties. Based on Software Engineering Institute’s model, requirement elicitation is a web-based application that allows all the people involved in the system to specify their requirements in a more sophisticated and chronological manner. All the requirements that are specified are passed through various phases when finally a definite and a well-defined set of requirements are defined for further development of the project.

An application of requirement elicitation process considered was the manufacturing of 120mm M829E3 armor piercing shell fired by the tank manufactured at the Iowa Army Ammunition Plant (IAAAP). The user specified various phases involved in the manufacturing with their requirements during each phase. Simultaneously developer’s also specified their requirements for the same. Thus by passing through all the phases of requirement elicitation a definite set of environmentally considered requirements were defined by both the teams.
This thesis describes the various phases of requirement elicitation with respect to the domain of study. It also describes the software approach adopted for designing and developing this web based product.
REQUIREMENT ELICITATION FOR ENVIRONMENTAL LIFE CYCLE PROCESSES

by
Mariam Y. Burmawalla

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

August 1998
REQUIREMENT ELICITATION FOR ENVIRONMENTAL LIFE CYCLE PROCESSES

Mariam Y. Burmawalla
BIOGRAPHICAL SKETCH

Author: Mariam Y. Burmawalla

Degree: Master of Science in Computer Science

Date: August 1998

Undergraduate and Graduate Education:

- Master of Science in Computer Science, New Jersey Institute of Technology, NJ, 1998

- Bachelor of Science in Computer Engineering, University of Bombay, India, 1996

Major: Computer Science
To my father, mother, baba, sisters Hasina and Zainab, brother Sandeep, Vinaya and all those who love me.
ACKNOWLEDGMENT

My sincere thanks and appreciation to my thesis advisor, Dr. Murat Tanik without whom my mater thesis would never have been accomplished. Dr. Tanik’s experience and guidance have molded me to not only develop a wonderful thesis, but also understand many aspects of life. I am also thankful to my thesis committee members, Prof. Ali Dogru, Prof. Leon Jololian and Prof. Franz Kurfess whose timely guidance and advice was of great help. I am grateful to Dr. Healey for not only having funded me but also for his guidance on the environmental aspect of my thesis.

I would also like to thank the Defense Army at Piccatinny for funding me and for providing information regarding the ammunition manufacturing.

I would sincerely thank Umang for his help and support during my entire thesis. Special thanks to Raj for his help in the initial phases of the project.

I appreciate my gratitude to the Graduate Studies and a special thanks to Annette for guiding me through my thesis documentation.

Finally but not the least I thank my parents, my sisters, my brother and all my friends who have been by my side and have supported me always. It is because of their prayers, love and patience that I have been able to complete my thesis. Lastly I would thank the Almighty for blessing me to complete my goals and for having been with me through my good and bad times.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>1.1</td>
<td>Requirements Engineering Overview</td>
</tr>
<tr>
<td>1.2</td>
<td>Knowledge Management Overview</td>
</tr>
<tr>
<td>1.3</td>
<td>Environmental Life Cycle Overview</td>
</tr>
<tr>
<td>2</td>
<td>STANDARD TECHNOLOGIES AND METHODOLOGIES</td>
</tr>
<tr>
<td>2.1</td>
<td>Knowledge Management</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Knowledge Representation</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Knowledge Filtering</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Knowledge Search</td>
</tr>
<tr>
<td>2.2</td>
<td>Software Engineering Institute's Requirement Engineering Framework</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Requirements Elicitation Framework</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Requirement Elicitation Process Model</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Requirement Engineering Techniques</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Requirement Elicitation Methodology and its Methods</td>
</tr>
<tr>
<td>2.2.4.1</td>
<td>Fact Finding</td>
</tr>
<tr>
<td>2.2.4.2</td>
<td>Gathering and Classification</td>
</tr>
<tr>
<td>2.2.4.3</td>
<td>Evaluation and Rationalization</td>
</tr>
<tr>
<td>2.2.4.4</td>
<td>Prioritization and Planning</td>
</tr>
<tr>
<td>2.2.4.5</td>
<td>Integration and Validation</td>
</tr>
<tr>
<td>Chapter</td>
<td>Domain and Problems</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
</tr>
<tr>
<td>3</td>
<td>Problems in Requirement Engineering</td>
</tr>
<tr>
<td>3.1</td>
<td>Problems of Scope</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Problems of Understanding</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Problems of Volatility</td>
</tr>
<tr>
<td>3.2</td>
<td>Problems in Knowledge Management</td>
</tr>
<tr>
<td>3.3</td>
<td>Problems in Environmental Life Cycle</td>
</tr>
<tr>
<td>4</td>
<td>Mapping of Ammunition Manufacturing in RePI</td>
</tr>
<tr>
<td>4.1</td>
<td>120MM M829E3 Ammunition</td>
</tr>
<tr>
<td>4.2</td>
<td>Ammunition Manufacturing Process</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Accumulation of Material from the External Sources</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Testing of Accumulated Material</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Assembly</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Coating Process</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Weight verification and Chambergage</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Leak Test and Shipment</td>
</tr>
<tr>
<td>4.3</td>
<td>Software design of 120MM M829E3</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Database Introduction</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Structure Chart</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Entity-Relationship Diagram</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Data Flow Diagram</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS

(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>46</td>
</tr>
<tr>
<td>4.4.1</td>
<td>47</td>
</tr>
<tr>
<td>4.4.1.1</td>
<td>48</td>
</tr>
<tr>
<td>4.4.1.2</td>
<td>52</td>
</tr>
<tr>
<td>4.4.1.3</td>
<td>54</td>
</tr>
<tr>
<td>4.4.1.4</td>
<td>56</td>
</tr>
<tr>
<td>4.4.1.5</td>
<td>57</td>
</tr>
<tr>
<td>4.4.2</td>
<td>59</td>
</tr>
<tr>
<td>4.4.2.1</td>
<td>60</td>
</tr>
<tr>
<td>4.4.2.2</td>
<td>64</td>
</tr>
<tr>
<td>4.4.2.3</td>
<td>65</td>
</tr>
<tr>
<td>4.4.2.4</td>
<td>68</td>
</tr>
<tr>
<td>4.4.2.5</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>71</td>
</tr>
<tr>
<td>4.5.1</td>
<td>71</td>
</tr>
<tr>
<td>4.5.2</td>
<td>75</td>
</tr>
<tr>
<td>4.5.3</td>
<td>77</td>
</tr>
<tr>
<td>5</td>
<td>87</td>
</tr>
<tr>
<td>5.1</td>
<td>87</td>
</tr>
<tr>
<td>5.2</td>
<td>88</td>
</tr>
<tr>
<td>5.2.1</td>
<td>88</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS
(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2</td>
<td>90</td>
</tr>
<tr>
<td>5.2.3</td>
<td>90</td>
</tr>
</tbody>
</table>

APPENDIX A ENTITY RELATIONSHIP DIAGRAM OF REQUIREMENT ELICITATION PROCESS ........................................... 91

APPENDIX B NORMALIZED TABLES FOR REQUIREMENT ELICITATION DATABASE .................................................. 92

APPENDIX C M829E3 RELATED DATA IN REQUIREMENT ELICITATION PROCESS .................................................. 96

REFERENCES ................................................................................................................................................. 98
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Steps in the Life-Cycle Assessment of a Product</td>
<td>6</td>
</tr>
<tr>
<td>2.</td>
<td>Adaptive Loops Learning Cycles</td>
<td>11</td>
</tr>
<tr>
<td>3.</td>
<td>Requirements Engineering is an Iterative Process</td>
<td>13</td>
</tr>
<tr>
<td>4.</td>
<td>Requirement Elicitation Process Model</td>
<td>15</td>
</tr>
<tr>
<td>5.</td>
<td>120mm M829E3 Ammunition</td>
<td>33</td>
</tr>
<tr>
<td>6.</td>
<td>High Explosive Ammunition</td>
<td>35</td>
</tr>
<tr>
<td>7.</td>
<td>Structure chart of 120mm M829E3 Ammunition</td>
<td>39</td>
</tr>
<tr>
<td>8.</td>
<td>A Sample E-R Diagram</td>
<td>40</td>
</tr>
<tr>
<td>9.</td>
<td>Level 0: Context Analysis Diagram</td>
<td>40</td>
</tr>
<tr>
<td>10.</td>
<td>Cartridge Manufacturing and Assembly</td>
<td>41</td>
</tr>
<tr>
<td>11.</td>
<td>Sub Assembly Accumulation</td>
<td>42</td>
</tr>
<tr>
<td>12.</td>
<td>Assembly Process</td>
<td>43</td>
</tr>
<tr>
<td>13.</td>
<td>Assembly of Ammunition</td>
<td>44</td>
</tr>
<tr>
<td>14.</td>
<td>Assembly of Ammunition</td>
<td>45</td>
</tr>
<tr>
<td>15.</td>
<td>User's Main Menu</td>
<td>47</td>
</tr>
<tr>
<td>16.</td>
<td>Identify Relevant People</td>
<td>48</td>
</tr>
<tr>
<td>17.</td>
<td>Describe the Problem</td>
<td>49</td>
</tr>
<tr>
<td>18.</td>
<td>Define Goals</td>
<td>50</td>
</tr>
<tr>
<td>19.</td>
<td>List Mission Scenarios</td>
<td>51</td>
</tr>
<tr>
<td>20.</td>
<td>Add Requirements</td>
<td>52</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES
(Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.</td>
<td>Requirements List</td>
<td>53</td>
</tr>
<tr>
<td>22.</td>
<td>Perform Abstraction</td>
<td>54</td>
</tr>
<tr>
<td>23.</td>
<td>Capture Rationale</td>
<td>55</td>
</tr>
<tr>
<td>24.</td>
<td>Prioritize the Requirements List</td>
<td>56</td>
</tr>
<tr>
<td>25.</td>
<td>Validate Requirements</td>
<td>57</td>
</tr>
<tr>
<td>26.</td>
<td>Obtain Authorization</td>
<td>58</td>
</tr>
<tr>
<td>27.</td>
<td>Developer's Main Menu</td>
<td>59</td>
</tr>
<tr>
<td>28.</td>
<td>Identify Domain Experts</td>
<td>60</td>
</tr>
<tr>
<td>29.</td>
<td>Identify Domain Models</td>
<td>61</td>
</tr>
<tr>
<td>30.</td>
<td>Conduct Technological Survey</td>
<td>62</td>
</tr>
<tr>
<td>31.</td>
<td>Assess Constraints</td>
<td>63</td>
</tr>
<tr>
<td>32.</td>
<td>Requirements List</td>
<td>64</td>
</tr>
<tr>
<td>33.</td>
<td>Perform Risk Assessment</td>
<td>65</td>
</tr>
<tr>
<td>34.</td>
<td>Perform Feasibility Analysis</td>
<td>66</td>
</tr>
<tr>
<td>35.</td>
<td>Cost/Benefit Analysis</td>
<td>67</td>
</tr>
<tr>
<td>36.</td>
<td>Prioritize Requirements</td>
<td>68</td>
</tr>
<tr>
<td>37.</td>
<td>Plan Incremental Development Stages</td>
<td>69</td>
</tr>
<tr>
<td>38.</td>
<td>Resolved Requirements</td>
<td>70</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Normalized Database</td>
<td>92</td>
</tr>
<tr>
<td>2. People Master</td>
<td>96</td>
</tr>
<tr>
<td>3. Goal Master</td>
<td>96</td>
</tr>
<tr>
<td>4. Scenario Master</td>
<td>96</td>
</tr>
<tr>
<td>5. Requirements Master</td>
<td>97</td>
</tr>
</tbody>
</table>
During a software project development, software requirement negotiation is the process where the customer needs are identified. This process is regarded as one of the most important parts of building a software system because during this stage it is decided precisely what will be built.

1.1 Requirement Engineering

Requirement negotiation is an iterative process where, through reflection and experience, users become familiar with the technology and developers become familiar with the user needs [HERLEA 97].

To produce quality products, understanding of requirements for a software system is a major concern. Requirement engineering is a common terminology used to specify various requirements related activities. Requirement engineering comprises of four specific processes [RAGHAVAN 94]:

1. Requirement Elicitation - The process, through which customers, buyers, or users of a software system discover, reveal, articulate, and understand their requirements. Experience over the last 30 years that incorrect, incomplete, or misunderstood requirements are the most common causes of poor quality, cost overruns, and late delivery of software systems. The ability to employ a systematic process for requirements elicitation is therefore one of the fundamental skills of a good software engineer.
2. 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.

3. Requirements specification - The process of recording the requirements in one or more forms, including natural language and formal, symbolic, or graphical representations; also, the product that is the document produced by that process.

4. Requirement validations - The process of confirming with the customer or user of the software that the specific requirements are valid, correct and complete.

To develop software products of great potential, software requirement elicitation process is an essential activity. A framework to illustrate these requirements was developed by Micheal Christal and Kyo Kang, members of the Software Engineering Institute (SEI), located at Carnegie Mellon University in Pittsburgh [MILLER 93]. The requirement elicitation process model consists of:

1. Fact-Finding Phase: This process allows an examination of the organization into which the target system will be placed, develops high level statements of the system's missions and roles, determines constraints on the architecture of the system's mission and roles, determines constraints on the architecture, and identifies the existence of the similar systems [CHRISTAL 92].

2. Requirement Gathering and Classification: This phase allows the gathering and organizing of information with the help of multiple views that express the information that is to be built.
3. Evaluation and Rationalization: Rationalization and evaluation phase is responsible for exposing inconsistencies in the gathered requirements and determining why the information has been expressed as a requirement.

4. Prioritization: This phase determines the relative importance of each requirement and the relative order the requirements.

5. Integration and Validation: This phase combines all the information acquired in the proceeding phases and creates a set of requirements. Validation is performed to determine that the requirements meet the goals and objectives outlined during the fact-finding stage.

### 1.2 Knowledge Management

Knowledge management entails formally managing knowledge resources in order to facilitate access and reuse of knowledge, typically by using advanced information technology. A wide range of technologies are being used to implement knowledge management systems: e-mail, databases and data warehouses, group support systems, browsers and search engines, internets and intranets, expert and knowledge-based systems, and intelligent agents [DANIEL 98].

Since earlier days knowledge has been stored on papers and in people's minds. Such a storage leads to inefficient management of knowledge as it is extremely difficult to maintain and update. Storing data in knowledge warehouses and knowledge bases greatly improves the management, maintenance and updating of data. Moreover, data stored in such a form has a wide accessibility.
It is also absolutely necessary for the user as well as the developers to understand the environment in which the system is supposed to function. The requirements must be specified such that the system can easily be integrated into the other existing systems. The lack of domain knowledge accessible by the user or developer can lead to improper knowledge management.

Gregory Piatetsky-Shapiro and William Frawley define knowledge discovery as "non-trivial extraction of implicit, previously unknown, and potentially useful information from data. Because knowledge discovery approaches can be designed to exploit characteristics and structures of the underlying application domain, knowledge discovery has found use in a wide range of applications.

Knowledge bases can become quite large and have great amount of information. Searching them efficiently becomes an extremely critical function. The most dominant search techniques include search engines, intelligent agents, and visualization models [DANIEL 98].
1.3 Environmental Life Cycle

A primary thrust of industrial ecology is that manufacturers practice product stewardship - designing, building, maintaining, and recycling products in such a way that they pose minimal impact to the wider world. Product stewardship should be broadly interpreted to include services, which should also be performed so as to have minimal impact. The way in which these tasks are addressed in formal manner is by the process of Life-Cycle Assessment (LCA), a family of methods for looking at materials, services, products, processes, and technologies over their entire life [GRAEDEL 95].

The essence of life-cycle assessment is the evaluation of the relevant environmental, economic, and technological implications of a material, process, or product across its life-span from creation to waste or, preferably, to re-creation in the same or another useful form.

LCA is made up of three stages: inventory analysis, impact analysis, and improvement analysis, as pictured in Fig. 1.1.1. First, the scope of the LCA is defined. An inventory analysis and an impact analysis are then performed, the result being an environmentally responsible product rating ($R_{ERP}$). This rating guides an analysis of potential improvements (which may feed back to influence the inventory analysis). Finally, the improved product is released for manufacture.
An effective LCA technology must be able to quickly and easily identify, then differentiate between, critical environmental impacts. This will allow designers to concentrate on the most important problems, reserving for later those that produce lesser impacts [GRAEDEL 95].

According to Robert Ayres of the European Institute of Business Administration, the work done on materials at the expense of energy represents society's battle against thermodynamics, and the energy invested per unit of material decreases as one approaches the top of the materials flow chain.

The generality of the options set should also be explicitly defined. In many cases, it is possible to generate numerous potential options, of which many cannot be feasibly reviewed. The challenge is to choose representatives options that provide valuable guidance for real-world decisions, but are limited and general enough to make an analysis realistic.
CHAPTER 2

STANDARD TECHNOLOGIES AND METHODOLOGIES

Enhancement in the field of technology has motivated people to represent their work in a more sophisticated manner adopting the latest methods and techniques. Information gathering and management is a very important factor. This chapter explains the various techniques used for managing information.

2.1 Knowledge Management

Cooperative work systems such as World Wide Web and Lotus Notes are beginning to tackle the aspect of knowledge management. Database systems can contribute much of the required functionality. Hence it is required to integrate functionality and ideas from these sources. With the web and the use of search engines, many people are already experiencing significant changes in how they use and manage knowledge [SKUCE 97].

2.1.1 Knowledge Representation

Knowledge representation can be done in many ways. The knowledge management systems represent knowledge in both human and machine readable forms. Human-readable knowledge is typically accessed using browsers or intelligent search agents. But some knowledge is accessible for machine-readable purposes, designed as an expert system's knowledge base to support decision making [DANIEL 98].
1. Human Readable Knowledge: Highlighted and case-specific information provides an appropriate level of representation required for users to make use of the knowledge. If the information to be represented is specific and highly filtered, then it is likely to be represented as a set of declarative statements. On the other hand, if the information is largely declarative, then knowledge text or rules might be used to represent the information and knowledge [DANIEL 98].

2. Logic: The most popular way of representing knowledge is by formal logic. Formal logic provides a very powerful tool for software development and can be used in all stages of the development process [PEYMAN 97].

3. Machine Readable Knowledge: Expert systems use their knowledge bases and user responses to help the user to provide solutions. Although some knowledge management systems contain such artificial intelligence-based systems, most knowledge management systems use artificial intelligence primarily in the form of intelligent agents to search human-readable knowledge [DANIEL 98].

4. Frames: Frames are used for declarative knowledge. Frames were developed because there is evidence that people subjected to new situations do not analyze the situation from scratch, but analyze situations using previous experience from objects, other people, locations, and so on. Frames contain information about many aspects of an object or an action. A frame might for example contain slots with necessary characteristics of an object, possible characteristics and typical instances of that object. Frames can be used in situations where there is a large amount of context dependent knowledge [PEYMAN 97].
5. **Scripts**: Scripts are clusters of facts about typical sequence of events in a given context. They can have a set of entry conditions, i.e. a set of conditions that must be met before the script can be executed. Scripts can be very useful in expressing events with very casual relationships. Scripts can also provide a way of dividing events into sub-events. Scripts can provide powerful tools in software development. Scripts can provide some support for reuse in design and the means for discovering and compensating the absence of a certain type of information in the requirements. They can be used to record facts about an existing software system, by keeping track of and organizing information about interaction between the systems components. They can also be used to describe the object interaction in a software model by describing normal flow of interaction between objects [PEYMAN 97].

6. **Semantic Nets**: Semantic nets are networks used to express semantic structures. The structure is very simple. They are built of nodes connected by arcs. There are objects and there are relationships between these objects [PEYMAN 97].

### 2.1.2 Knowledge Filtering

Humans do not resort to filtering, thereby generating knowledge which is error prone. Whereas the knowledge generated by the system is always accurate, precise, complete and error free. Perhaps the most visible and frequent use of computer-based filtering is the message filtering that categorizes and prioritizes e-mail messages. A number of products also help monitor qualitative databases [DANIEL 98].
2.1.3 Searching for Knowledge

Knowledge search is a very wide concept that adopts various search techniques. A few of them are:

1. Search Engines: It is one of the most widely used modes of searching on the web. A wide range of well-known internet search engines - like AltaVista, Excite, Infoseek, Lycos, WebCrawler, and Yahoo - have been used to guide users to surf for information on the Internet. These and other search engines can be adapted to Intranet environment for knowledge management [DANIEL 98].

2. Intelligent agents: Intelligent agents are used to connect people to the information available on the Internet or Intranets. Heuristics can be used to gather additional insights into user's interests. Based on message syntax, attempts can be made to determine significant phrases that provide insight into user goals.

3. Visualization models: Two emerging tools “Perspecta” and “InXight” represent different ways of visualizing knowledge space. Perspecta creates SmartContent using meta information derived from source documents. It is structured information in databases and tagged documents such as news feeds, or unstructured information in office documents and Web pages. For unstructured documents, it has a document analysis engine that performs linguistic analysis and automatically tags documents. InXight software, a spin-off from Xerox PARC, recently released its VizControl information visualization software for visualizing large hierarchies. VizControl technology offers several novel visualization formats, each of which exploit "focus + context" techniques that foreground objects of interest while preserving the overall structure of even very large data sets.
2.2 Software Engineering Institute's Requirement Engineering Framework

Requirements' engineering is "the disciplined application of scientific principles and techniques for developing, communicating and managing requirements." [STEP 91]. Loucopoulos and Champion define requirements engineering as "the systematic process of developing requirements through an iterative process of analyzing a problem, documenting the results of observation and checking the accuracy of the understanding gain."

![Figure 2 Adaptive loops learning cycles (RAGHAVAN 94).](image)

Requirements elicitation normally involves several developers (the requirements analysts and software engineers) and several customers (the buyers or users of the software). Each of these persons brings different knowledge and skills to the effort [RAGHAVAN 94].

There are three learning cycles, as shown in Figure 2. The developers are assisted by the users in gaining new viewpoints about their requirements, and through reformulating the requirements, the user learns more about them. The system receives
pressure for evolution as the users learn more about how it can be used, and the system induces that learning on the users. The system evolves by actions of the developers, who in turn gain enhanced understanding of the system through that.

The requirements elicitation process using the adaptive loops framework focuses on addressing, supporting, and facilitating these learning cycles. It is especially useful when there are requirements articulation problems, and it is helpful in overcoming some of the technical issues of requirements elicitation for the evolution of complex and incomplete systems.

2.2.1 Requirements Elicitation Framework

Rzepka decomposes the requirements engineering process into three activities [Rzepka 89]:

1. elicit requirements from various individual sources,
2. insure that the needs of all users are consistent and feasible and
3. validate that the requirements so derived are an accurate reflection of user needs.

Elicitation will likely iterate with these activities during requirements development.

Prior to SEI's proposed requirements framework, the requirements elicitation process consisted of scattered processes, methodologies and techniques, each having their own benefits, strengths, and weaknesses. By themselves, none of these strategies were able to address all the attributes necessary to produce a quality set of requirements that met the needs of the stakeholder community. Existing models, methodologies, and techniques failed to address the problems inherent in the requirements elicitation activity
namely: system scope, understanding among participants of the process, understanding among personnel effected by the process, and a recognition of the volatility of requirements [MILLER 93].

In an effort to address these issues, the framework was established. It is composed of a process model, methodology, and a group of supporting techniques. The process model guides the methodology and techniques imposed by the framework. A methodology is a fine grained activity that supports a process. Methodologies are prescriptive in nature and often recommend methods, techniques, and tools to enact them.

![Diagram of Requirements Engineering](image)

**Figure 3** Requirements Engineering is an Iterative Process [CHRISTEL 92].
One of the integral parts of the requirement elicitation framework is formal specification language. The formal specification language offer:

1. Clarity: Formal Specification languages can remove some elements of ambiguity from the process. They offer explicit syntax and semantics that define the language and a set of relations that precisely defines object interaction [PLAYLE 96].

2. Consistency: Because the language is relatively fixed, a formal specification language reduces the chance for misinterpretation when it passes through various development groups and lifecycle phases.

3. Completeness: Formal specification languages are incomplete, and some languages are purposely incomplete to allow designers some freedom. Developers and designers must guard against incompleteness to ensure all the requirements are treated consistently.

4. Prototyping: Prototyping, as a means of requirements elicitation, can be highly effective. Tamai and Itou also found that during the prototyping phase users were not intimately familiar with the system, so developers provided a simplified interface. As the users became more familiar with the system, they demanded more functionality from the software, resulting in a more complicated interface. When this product was delivered, the users required both the simplified version and the complicated version. The conclusion was that “features” and “interim requirements” may become part of the users’ expectations and eventually become system requirements.
2.2.2 Requirement Elicitation Process Model

The communication between the user-oriented and developer-oriented activities is cyclical, and enhanced via modeling. The communication enhancement is desirable. The representation of the requirements should promote understanding while allowing for inevitable change, and hence this representation should be introduced as early into the requirements engineering process as possible while still maintaining the desirable characteristics of modifiability, readability, and analyzability [CHRISTEL 92].

![Figure 4 Requirement Elicitation Process Model [CHRISTEL 92].](image)
This elicitation process model is first executed during the concept exploration phase of system development, which is initiated after the creation of a mission needs statement. Following this phase, the first level of detail in the requirements specification is achieved. During the subsequent demonstration and validation phase, these specifications are validated, the unclear requirements clarified, the unknown requirements identified, and the existing ones refined as necessary. Based on communication mechanisms (such as prototyping) employed during this phase, these elicitation steps are then cycled through again beginning with “requirements gathering” to detail and improve the requirements document. The demonstration and validation phase is entered with incomplete requirements, and therefore that these elicitation process steps are returned to after the first pass through the concept exploration phase [CHRISTEL 92].

With regard to the user community, fact-finding begins with identifying the relevant parties at multiple levels within the community, e.g., from a high-level commander for a strategic long term perspective to an end user for the immediate perspective. The operational context and problem context are defined, perhaps through goal trees and mission statements, which help with the later filtering of the requirements. This includes an objectives analysis, which studies the user organization’s objectives, constraints against full achievement of the objectives, and their influences and interactions. Context analysis and the determination of operational modes and mission scenarios complete the user-oriented task fact-finding activities. The developer oriented fact-finding tasks are performed in parallel.
2.2.3 Requirement Engineering Techniques

The requirements definition process comprises these steps: [BRACKETT 90]

- Requirements Identification,
- Identification of software development constraints,
- Requirements analysis,
- Requirements representation,
- Requirements communication and
- Preparation for validation of software requirements.

Techniques for requirement elicitation generally provide operational-level tactics and guidelines. They usually focus narrowly on specific aspects of the elicitation process. Such techniques are [RAGHAVAN 94]:

1. Brainstorming: Brainstorming is a simple group technique for generating ideas. It allows people to suggest and explore ideas in an atmosphere free of criticism or judgment. The session consists of two phases. In the generation phase, participants are encouraged to offer as many ideas as possible, without discussion of the merits of the ideas. In the consolidation phase, the ideas are discussed, revised, and organized. For purposes of software requirements elicitation, brainstorming can be helpful in generating a wide variety of views of the problem and in formulating the problem in different ways. It is especially useful very early in the elicitation process. When used correctly, it can help overcome some of the underlying difficulties of requirements elicitation:
• It stimulates imaginative thinking to help users become aware of their needs.
• It helps build a more complete picture of the users' needs.
• It can avoid the tendency to focus too narrowly too soon.
• For some personality types, it provides a more comfortable.

Good brainstorming sessions are very helpful in overcoming some of the cognitive limitations of participants by allowing (or forcing) them to expand their thinking. The lack of criticism and judgment during the generation phase also helps overcome some of the communication barriers of requirements elicitation.

2. Interviewing: Interviewing is an important technique for eliciting detailed information from an individual. It is commonly used in requirements elicitation for large systems as part of some of the high-level elicitation techniques. It can also be used for small projects as the only requirements elicitation technique.

Interviewing is not simply a matter of asking question. It is a more structured technique that can be learned, and software engineers can gain proficiency with training and practice. It requires the development of some general social skills, the ability to listen, and knowledge of a variety of interviewing tactics.

A skilled interviewer can help the user to understand and explore software requirements, thus overcoming many of the articulation problems and communications barriers.
2.2.4 Requirement Elicitation Methodology and its Methods

Most software engineering methods presume that requirements are explicitly and completely stated; however, experience shows that requirements are rarely complete and usually contain implicit requirements [PLAYLE 96].

The elicitation methodology should be prescriptive in nature in order to provide the guidance as to how the specifications should be elicited from the user. Guidelines for tailoring the methodology to specific problems will most likely be developed, validated, and refined iteratively. As the methodology matures and more problem areas are addressed, the framework will grow as well [CHRISTEL 92].

2.2.4.1 Fact Finding: The very first step in requirements elicitation involves determining what is the problem to be addressed, and who needs to be involved in this decision-making as well as who will be affected by the problem’s formulation and eventual solution. The output from this activity includes [CHRISTEL 92]:

- a statement of the problem context,
- the overall objectives of the target system and
- boundaries and interfaces for the target system.

Activities performed in this phase are the creation of operational, problem, and organizational contexts, identification and documentation of similar systems, and the assessment of cost and implementation constraints imposed by the customer.

If the understanding and the representation of the problem domain is mature, the objectives of this phase may be easily identified, understood, and completed. However, if this is not the case, cross functional teams should be engaged to perform this task. The
objective of cross functional teams is to involve experts to aid in the definition and development of the outputs for this stage, and to involve the relevant parties in order to create a sense of commitment and shared ownership. The accuracy and completeness of this phase is critical to the success of the entire process. Multiple passes through this phase should be considered to promote completeness [MILLER 93].

An effective approach to achieving this cross-disciplinary communication for fact-finding is the use of a group process technique, such as Joint Application Design (JAD). All the affected parties should be represented in the group that will perform these early fact-finding tasks. This promotes shared ownership, rapid early problem formulation, and an aligned perspective and understanding between the elicitation communities of the problem to be solved and the scope of the subsequent requirements [CHRISTEL 92].

2.2.4.2 Gathering and Classification: It is important to gather as much information as possible from users, developers, and customers. Some of this information may come from the group development techniques employed during fact-finding, such as JAD. More information can be gathered through the use of interviews directly with end users and other affected parties. Questionnaires, observations, and simulation environments are other techniques that can be utilized to get information from different individuals and groups. The output from this activity includes [CHRISTEL 92]:

- customer and user oriented objectives and
- customer and user oriented needs and requirements.
A tailored JAD session serves to provide the framework for this step in the requirements elicitation process. Structured interviews, and questionnaires are used to capture and document information and underlying rationale. The JAD process starts with a problem-research phase. Analysts perform structured interviews to identify the relevant parties and to acquire the information and rationale necessary to meet the goals and objectives of this phase. The structure and the format of the interviews are dictated by the techniques used to model and represent the information required [MILLER 93].

The views are better understood if they can be structured into manageable pieces. This is especially true given that the elicitation process will be incremental, to deal with inevitable changes in requirements. If we return to monolithic views of the complete system, it will be very difficult to both comprehend such a large view and also to find portions of that view which may be affected by an incremental change to the requirements. Thus, there must also be a decompositional process associated with requirements gathering, where the views can be broken down into meaningful components [CHRISTEL 92].

2.2.4.3 Evaluation and Rationalization: The goal of this phase is to fully develop and evaluate the underlying rationale behind the requirements gathered to this point. A risk assessment should be performed to address technical, cost, and schedule concerns. In addition, the rationale behind the information gathered in previous stages should be examined to determine whether the true requirements are hidden in this rationale instead of being explicitly expressed. This rationale and risk assessment are the two main outputs from this activity [CHRISTEL 92].
The objectives, goals, and constraints developed in the fact-finding phase are compared to the requirements detailed in this document and models developed in requirements gathering and classification phase. The evaluations are performed by the requirements analyst with the aid of the stakeholders through a series of structured interviews [MILLER 93].

The evaluation is performed by comparing the requirements representations against the organizational, goal, and constraint models. The comparison determines if the requirements address the right issues.

The rationale process is performed in parallel with the prior evaluation. This consists of a series of interviews in which the analyst and the stakeholders evaluate the requirements model against the rationale stored. The objective is to determine why each requirement is present.

Once the rationale has been collected and examined, inconsistencies can ideally be found and better choices on decision points or issues made to both resolve these inconsistencies and address the needs reflected in the rationale. In addition, this rationale is extremely useful in later passes through the elicitation methodology as documentation on why particular choices were made. If incremental changes to the requirements are to be made, these changes can be checked to see if they are in agreement with the rationale underlying the rest of the existing requirements [CHRISTEL 92].
2.2.4.4 Prioritization and Planning: The goal of the prioritization phase is to arrange the requirements in order of relative importance from the view of the client and the view of the developer [CHRISTEL 92].

Incremental development, of both the system and the requirements, is stressed in the process model. If requirements are prioritized, then high priority needs can be addressed first, and the subsequent requirements changes defined and reexamined, before low priority needs (which could change as well) are implemented. This can result in cost and time savings when processing the inevitable requirements changes during system development. The requirements must be prioritized based on cost, dependency, and user needs [CHRISTEL 92].

2.2.4.5 Integration and Validation: The goal of the integration and validation is to reduce the conflicts found in the requirements, to address completeness, and to validate the requirements [CHRISTEL 92].

Outputs of this activity are a set of complete requirements or a set of incomplete but validated requirements ready for the specification and formal validation processes. Output may also be in the form of requirements which are lacking in some quality. The incomplete requirements are iterated through the requirements elicitation model to resolve the open issues [MILLER 93].

Integration of multiple views should occur as much as possible through the involvement of all the affected communities, so that this shared ownership is not lost. Validation of the requirements by all affected parties ensures that their concerns are met. Subsequent passes through the elicitation methodology outlined here address the
requirements deficiencies, inconsistencies, and other problems found during the demonstration and validation steps [CHRISTEL 92].

The DoD software technology plan states that “early defect fixes are typically to orders of magnitude cheaper than late defect fixes, and the early requirements and design defects typically leave more serious operational consequences.” One way to reduce requirements error is by improving requirements elicitation [CHRISTEL 92].
CHAPTER 3

DOMAIN AND PROBLEMS

Problems encountered during any phases of development life cycle leads to unsatisfactory output which may result in the failure of the system. Here we have described the various kinds of domains and their related problems.

3.1 Problems in Requirements Engineering

There are many problems associated with requirements engineering, including problems in defining the system scope, ensuring understanding between developer and user, and problems in dealing with the volatile nature of requirements. These problems may lead to poor requirements and the cancellation of system development, or else the development of a system that is later judged unsatisfactory or unacceptable, has high maintenance costs, or undergoes frequent changes. Issues involved in this problem area include:

1. Achieving requirement completeness without unnecessarily constraining system design.

2. Analysis and validation difficulty.

3. Changing requirements over time [CHRISTEL 92].

Software requirements characteristically suffer from inconsistency, incompleteness, nonspecific language (ambiguity), duplication, and inconstancy. On a complex system, the documents that make up the requirements are voluminous [PLAYLE 96].
Cooperation between participants in the process is quite difficult, even if they meet in a physical meeting room. The process involves a social network of people with different professional backgrounds and different views over the system that must be built. If the participants in the process are in different organizations or different cities, meetings can be costly, inconvenient and infrequent. This leads to problems of communication, which can greatly impact the quality of the elicited requirements [HERLEA 97].

Requirements problems have been identified as major contributors to program cost overruns and schedule slips. This “problem with requirements” can be attributed to two major factors [IVY 90]:

1. The initial requirements were incomplete, inaccurate, and / or misunderstood by the designers / developers.
2. Circumstances changed which resulted in changes to the requirements.

Problems of requirements elicitation can be grouped into three categories: [CHRISTEL 92].

1. problems of scope, in which the requirements may address too little or too much information.
   - the boundary of the system is ill-defined and
   - unnecessary design information may be given.

2. problems of understanding, within groups as well as between groups such as users and developers.
   - users have incomplete understanding of their needs,
   - users have poor understanding of computer capabilities and limitations,
   - analysts have poor knowledge of problem domain,
• user and analyst speak different languages,
• ease of omitting "obvious" information,
• conflicting views of different users and
• requirements are often vague and untestable, e.g., "user friendly" and "robust".

3 problems of volatility, i.e., the changing nature of requirements.

• requirements evolve over time.

3.1.1 Problems of Scope

Elicitation techniques need to be broad enough to establish boundary conditions for the target system, yet still should focus on the creation of requirements [CHRISTEL 92].

Requirements elicitation must begin with an organizational and context analysis to determine the boundary of the target system as well as the objectives of the system. Less ambitious elicitation techniques not addressing this concern run the risk of producing requirements which are incomplete and potentially unusable, because they do not adhere to the user’s or organization’s true goals for the system. Performing an organizational and context analysis allows these goals to be captured, and then used to verify that the requirements are indeed usable and correct.

The problem is manifested by the fact that the requirements focus is often in what is to be built without consideration for how it will be operation. Operations costs are directly driven by program definition and the requirements imposed by all elements of the program. Frequently, the requirements do not consider the operations concepts,
constraints, or plan. When operations requirements are finally considered, addition
requirements will be needed to overcome this initial oversight [IVY 90].

Elicitation techniques can be overly ambitious as well. Elicitation must focus on
the creation of requirements, not design activities, in order to adequately address users’
concerns and not just developers’ needs [CHRISTEL 92].

Environmental factors have a strong influence on requirements elicitation.
Environmental factors include: [CHRISTEL 92].

- hardware and software constraints imposed on a target system (the target system will
typically be a component of some larger system with an existing or required
architecture already in place),

- the maturity of the target system’s domain,

- the certainty of the target system’s interfaces to the larger system and

- the target system’s role within a larger system

3.1.2 Problems of Understanding

Problems of understanding can be separated into three issues: [CHRISTEL 92]

- The communities involved in elicitation possess a variety of backgrounds and
experience levels, so that which is common knowledge to one group may be
completely foreign to another. This makes it difficult for a requirements analyst to
interpret and integrate information gathered from these diverse communities.

- The language used to express the requirements back to these stakeholder communities
may be too formal or too informal to meet the needs of each of the groups, again
because of the diversity of the communities.
• The large amount of information gathered during elicitation necessitates that it be structured in some way. The understanding of this structure is dependent on the characteristics of the stakeholder communities.

3.1.3 Problems of Volatility

Requirements change. During the time it takes to develop a system the users’ needs may mature because of increased knowledge brought on by the development activities, or they may shift to a new set of needs because of unforeseen organizational or environmental pressures. If such changes are not accommodated, the original requirements set will become incomplete, inconsistent with the new situation, and potentially unusable because they capture information that has since become obsolete [CHRISTEL 92].

One primary cause of requirements volatility is that “user needs evolve over time”. The requirements engineering process of elicit, specify, and validate should not be executed only once during system development, but rather should be returned to so that the requirements can reflect the new knowledge gained during specification, validation, and subsequent activities. A requirement engineering methodology should be iterative in nature, “so that solutions can be reworked in the light of increased knowledge.”

3.2 Problems in Knowledge Management

Knowledge is a critical resource but we still do not have many new ideas on how to manage it. Most knowledge is currently kept in conventional documents that are hard to structure, classify, browse, search, and even find. Most of the knowledge is still recorded as unstructured natural language, with all its shortcomings of precision and conciseness.
Documents in particular, small parts of them such as sentences, are often hard or impossible to find, hard to update cooperatively, and hard to keep coordinated merging of information from various sources is difficult. Most systems today offer nothing in the way of inference, semantic checking, or natural language processing. Finding and organizing documents is very difficult [SKUCE].

For frames to be used in a formal manner, there needs to be some standard ways of representing them. There needs to be a specific number of slots in each frame, or type of frame. [PEYMAN 97]. Even though semantic nets were developed a long time ago, except for very basic relationships there are no standard methods of how the arcs should be labeled. It should be mentioned that even the most basic facts have problems associated with them [OBJA 95].

3.3 Problems in Environmental Life Cycle

In college and university computer science programs, instructors usually create the requirements specification; students are rarely involved in the process. It is even more rare for students to be taught the specific techniques that software engineers use for requirements elicitation. This can probably be attributed to the absence of these techniques from most computer science textbooks and the lack of familiarity with these techniques on the part of instructors [RAGHAVAN 94].

The problems in requirement engineering, knowledge management and environmental life cycle cite that the software tools build for them are not used at all, or are used by a small minority of people. The most basic reason of this is that the end users
were not involved while developing such tools. It is very difficult to build a successful system without end user involvement.
CHAPTER 4
MAPPING OF AMMUNITION MANUFACTURING IN REPI

Requirement specification is the most important requirement during any product development. It completely reflects the needs of the customer. Precise and validated requirements include implicit assumptions, no inconsistencies and it also aids the system designers with clear specified inputs.

Requirement elicitation process as described in the earlier chapters is one such method for requirement illustration. To explain the use of this system, we have considered the manufacturing process of 120MM M829E3 ammunition, a cartridge manufactured by the Department of Defense. This system is thus used to do the software requirement analysis for the environmental lifecycle simulation model.

4.1 120MM M829E3 Ammunition

120MM M829E3, is an armor piercing shell fired by the tank and is manufactured by the Department of Defense of US located at IAAA.

The ammunition is composed of subparts that are combined together during manufacturing. Each phase undergoes rigorous testing. The various components of 120MM M829E3 as pictured in the Figure 5 are:

1. Base Case  2. Tracer  3. Penetrator
4. Igniter  5. Primer  6. Fin
**BASE CASE**
- Steel Turning
- Solid Rubber
- Water Soluble Coolants (WSC)
- Phosphate Sludge

**TRACER**
- Barium Sulfate
- Barium Peroxide
- Igniter
- Hydrogen Flouride
- Combustion Prod.

**PENETRATOR**
- Depleted Uranium Wastes
- Low level Radioactive Waste
- WSC

**IGNITER**
- Black Powder (QA Tests)
- Lead
- Cardon Disulfide

**PRIMER**
- Lead Thiocyanate
- Potassium Chlorate
- Benite or Black Powder

**PACKAGING**
- Pallet - CARC
- Paint

**COMPOSITE**
- Carbon Fiber
- Scrap Composite
- Sulfuric Acid
- Water Soluble Coolants

**FIN**
- Water Soluble Coolants

**PROPELLANT - 3 compositions**
- Acetic Acid
- Nitric Acid
- Acetic Anhydride
- Nitrocellulose Fines
- Cyclohexone
- Nox
- Diethyl phthalate
- RDX/HMX
- DEGDN/BTNN/NG/TEGDN
- Dimethyl Formamide
- Sulfuric Acid

*Figure 5* 120MM M829E3 Ammunition [DEPARTMENT of DEFENSE](http://www.pica.army.mil/orgs/ccac/cch/cch-a.html)
4.2 Ammunition Manufacturing Process

Ammunition manufacturing and assembly is done at IAAA. The following section explains the various phases of manufacturing.

4.2.1 Accumulation of Material from External Sources

All the materials that is required for the manufacturing of the 120MM M829E3 are shipped form external sources either by rail or road. The material are either in solid or liquid state. The following materials are accumulated:

<table>
<thead>
<tr>
<th>1. Inert Materials</th>
<th>2. Flammable liquids</th>
<th>3. Projectiles</th>
</tr>
</thead>
</table>
HIGH EXPLOSIVE AMMUNITION

CHEMICAL EQUATION

\[ \text{KClO}_3 + \text{Pb(CNS)}_2 + \text{Sb}_2\text{S}_3 + \text{TNT} + \text{GROUND GLASS} \]

Figure 6 High Explosive ammunition [HEALEY 97]
4.2.2 Testing of Accumulated Material

After each material is received, it is inspected under various conditions. If the material satisfies the testing criteria, it is transferred to storage and segregated per lot. If it is rejected, it is segregated, tagged and held for disposition back to the customer.

4.2.3 Assembly

Tested material accumulated in the storage locations is assembled together, tested again for weight, size and shape.

The steps followed during the assembly of the ammunition are:

1. Inert materials, projectiles along with base case, spring disc and retain ring are assembled together to form the propelling charge case assembly (PCCA).
2. Inspected projectiles are stenciled and then assembled with tracers using adhesive.
3. Greased O-ring is assembled with the primer that is then inserted into the PCCA.
4. Stenciled projectile with tracers is then assembled with the case adapter.
5. PCCA is assembled with the primer and the projectile is assembled with the case adapter. Both these components are then binded to each other.
4.2.4 Coating Process

The bonded cartridge in the above step is then coated with a base coat and top coat. After each coating is applied, it is tested for its thickness.

Base coat thickness : 20/60 um.
Top coat thickness : 20 um minimum.

4.2.5 Weight Verification and Chambergage

The primed loaded cartridge is weighed and then chambergaged. If chambergaged is not accepted, then the cartridge is tagged and submitted to quality for disposition.

4.2.6 Leak Test and Shipment

On satisfying all the above processes the cartridge is packed into palletized cans and then tested for any leaking. If it passes the leak test, the loaded pallet is shipped.

4.3 Software Design of 120mm M829E3

The users have to specify the requirements required for the manufacturing of the ammunition. A way to specify this is with the help of software design where users and developers interact with each other to understand each others requirements. Requirements specified by each of them is stored in the database and then evaluated for final results.

Requirement elicitation process is one such software technique used to specify the requirements. A database is required to store the information for future evaluation.
4.3.1 Database Introduction

A database-management system (DBMS) consists of a collection of interrelated data and a set of programs to access those data. Database is referred to as collection of information related to a particular enterprise. The most important use of a database is to provide an environment that is both convenient and efficient to use in retrieving and storing database information. Advantages of Database Systems are:

1. Eliminates data redundancy and inconsistency.
2. Facilitates easy access of data.
3. Provides data isolation.
4. Provides data integrity.
5. Facilitates concurrent-access of the database information

4.3.2 Structure Chart

A structure chart explains the hierarchy of the system. It is a tree type structure with the main entity at its root. The nodes in the structure chart are the different components of the system.

The structure chart of 120MM M829E3 ammunition is shown in the Figure 7.
4.3.3 Entity-Relationship Diagram

The entity-relationship (E-R) data model is made up of collection of basic objects called entities and relationships among these objects. An entity is a real world "object" or "things" which has an independent existence. Each entity has its own specifications that are indicated as attributes in the E-R schema. A relationship is an association between various entities of the E-R model.

The overall logical structure of a database can be expressed graphically by an E-R diagram. The components of the E-R schema are:

1. Rectangles: This represents the entity sets.
2. Ellipses: Represents attributes of the entities.
3. Diamonds: Represents relationship among entity sets.
4. Lines: Links attributes to entity sets and entity sets to relationships.
4.3.4 Data Flow Diagram

Data flow diagrams explain the different processes in the system and the flow of information in each of these processes. They are divided into different levels.

The first level called the level-0 or context analysis diagram explains the relationship between all external entities with the system. The next level, level-1 breaks the main system into its high level processes along with its inputs and outputs. Each level below then breaks the corresponding processes explains the flow of information through each of them.
Figure 10 Cartridge Manufacturing and Assembly
Figure 11 Sub Assembly Accumulation
Figure 12 Assembly Process
Figure 13 Assembly of Ammunition
Figure 14 Assembly of Ammunition
4.4 120MM M829E3 Specification in REPI

The following sub-sections describe the mapping of the manufacturing if 120MM M829E3 cartridge into the different phases of requirement elicitation process.

Requirement elicitation process has two main parts:

1. User menu and
2. Developers menu.

Each of them is described using the five different phases of requirement elicitation as described in the previous sections. They are:

1. Fact-Finding.
2. Gathering and Classification.
3. Evaluation and Rationalization.
4. Prioritization and Planning.
5. Integration and Validation.
4.4.1 User’s Menu

In this menu, the users describe their requirements with respect to the system domain.

Users personal information and project related information is stored in the databases.

The users menu is as shown in the figure below.

![User's Main Menu](image)

**Figure 15** User's main menu
4.4.1.1 Fact Finding: This phase has the following sub-sections:

1. Identify relevant people.

People from the military who have requested for the manufacturing of the ammunition insert their personal information in this form. Information like their name, telephone number, email and their category are specified. The following screen layout shows the format of representation.

![User's Fact Finding Phase](image)

**Figure 16** Identify relevant people
2. Describe the problem.

Here the user specifies the problem under consideration. The problem specification in this case is the manufacturing of the ammunition. The screen layout for the above phase is

![Diagram of User's Fact Finding Phase](image)

*Figure 17* Describe the problem
3. Define goals.

In this screen, the user specifies the various objectives of the system. In the manufacturing of 120MM M829E3 ammunition, the various goals could be:

1. Receive materials from external sources.
2. Testing of each process.
3. Assembly of gathered material.
4. Applying of coats of paint to different components.
5. Verification of weight.
6. Shipment of the manufactured ammunition.

Figure 18 Define goals
4. List mission scenarios.

Each goal has many missions to accomplish. This screen allows the user to enter the different scenarios that are related to a particular goal specified in the above screen. Each scenario has an event, action and reaction related to it. The figure below shows the screen layout of the list mission scenario option and its event, action and reaction.

Figure 19 List mission scenarios
4.4.1.2 Gathering and Classification: Requirement specification is the most important and essential need in requirement engineering. In this phase the user is allowed to specify his needs to the developer. This phase has the following sub-sections:

1. Add requirements.

The various of requirements, a user can specify for the manufacturing of the ammunition are:

1. Gather inert materials.
2. Accumulate flammable liquids.
3. Stencil the projectile.
4. Test the sabot, penetrator and cartridge for touchup.

And many more requirements that can be helpful for the understanding of the manufacturing process. The screen layout is as shown below.

![Figure 20 Add requirements](image-url)
2 List requirements.

To assist the user to view the requirements that he has listed or any other user has listed, this screen is activated. Here the listing can be done on user basis or all the listing can be done.

For example, for the requirement inert materials, they are required because they are used in the manufacturing of the ammunition. Similarly, the user states the reasons for all the other requirements. The screen layout is as shown below.

![Requirements list](http://cache.ntl.edu/500/MJ_60_61.jpg)

**Figure 21** Requirements list
4.4.1.3 Evaluation and Rationalization: The user evaluates his requirements on basis on the necessity of the requirement. This phase consists of two sub-parts:

1. Perform abstraction.

Every requirement that the user mentions in the above screens, must have a reason for existence. The user in this screen mentions the reason why the requirement is valid and necessary.

For example, for the requirement inert materials, they are required because they are used in the manufacturing of the ammunition. Similarly, the user states the reasons for all the other requirements. The screen layout is as shown below.

![User's Evaluation & Rationalization Phase - Netscape](image)

Figure 22 Perform abstraction
2. Capture rationale.

In this sub-menu, the user explains the reason for the requirement from his viewpoint. The screen layout is as shown below.

![Diagram showing user interface](image)

**Figure 23** Capture rationale
4.4.1.4 **Prioritization and Planning:** All the requirements are prioritized depending on their importance level, understanding and priority. The user specifies these three levels for each of the requirements that he has specified. The screen layout for the screen, prioritize the requirement list is as shown below.

![Figure 24](image)  
*Figure 24* Prioritize the requirements list
4.4.1.5 Integration and Validation: This phase is divided into two parts:

1. Validate requirement.

In this screen the user confirms his requirement against the goal that he has specified. During the course of analysis, if the goal specified against a particular requirement, during the gathering and classification phase does not hold valid, the user can thus change the goal at this point of analysis.

The screen layout is as shown below.

Figure 25 Validate requirements
2. Obtain authorization.

An authorized person authorizes the requirements specified by the user. This confirms that the requirement specified by the user is valid against the goal specified and is necessary for the development of the ammunition. The screen layout is as shown below.

![Figure 26 Obtain Authorization](image-url)
4.4.2 Developer's Menu

Developer's prospective towards the project is specified in the menus in this section. The phases are similar to the ones in the users menu, except for some screens that vary in the two sections. The developer's menu is as shown below.

Figure 27 Developer's main menu
4.4.2.1 Fact Finding: This consists of:

1. Identify domain experts.

This is similar to the identify relevant people screen. Here the developers information is entered unlike the users information. The screen layout is as shown below.

![Developer's Fact Finding Phase](image-url)

**Figure 28** Identify domain experts
2. Identify domain models.

In this screen, the developer specifies the domain model and the architectural model required for this project. In case of the manufacturing process, the architectural model is the use of relational databases where the information is stored in the database of future access and decision making. The screen layout is as shown below.

![Figure 29 Identify domain models](image-url)
3. Conduct technological survey.

Technological survey aids the developer to mention the technology to be used for the manufacturing process. The screen layout is as shown below.

![Developer's Fact Finding Phase](image)

**Figure 30** Conduct technological survey
4. Assess constraints.

The developer specifies the various constraints that should be considered during the manufacturing process. The screen layout is as shown below.

Figure 31 Assess constraints
4.4.2.2 Gathering and Classification: This phase is composed of:

1. Add requirements.
   
   This is similar to the add requirements menu in the users menu. Here the developer enters his requirements related to the domain.

2. List requirements.
   
   This is again similar to the menu in the users phase. Here either all requirements can be listed or the requirements related to a particular developer.

3. Classify requirements.

   This is an additional screen as compared to users menu. Here the requirements can be listed based on their category. If no category is mentioned, then all the requirements are listed. The screen layout is as shown below.

![Requirements list](image-url)

**Figure 32** Requirements list
### 4.4.2.3 Evaluation and Rationalization

The requirements specified by the developer can be evaluated and rationalized in this phase. It consists of:

1. **Risk assessment.**

The risk involved in a particular requirement can be stated at this point. The risk could be in terms of machinery usage, hardware, manpower etc. The screen layout is as shown below.

![Figure 33: Perform risk assessment](image-url)

**Figure 33** Perform risk assessment
2. Feasibility analysis.

Specify the feasibility of the requirement with respect to the manufacturing of the ammunition. The screen layout is as shown below.

Figure 34 Perform feasibility analysis
3. Cost/Benefit analysis.

Analyze and state the cost and the benefit related to the particular requirement.

The screen layout is as shown below.

Figure 35 Cost/Benefit analysis
4.4.2.4 Prioritization and Planning: The requirements specified need to be prioritized for decision making. This phase is made of:

1. Prioritize requirements.

Requirements specified by the developer are prioritized based on cost, dependence level and priority. The screen layout is as shown.

Figure 36 Prioritize requirements
2. Plan incremental development stages.

This screen lists all the requirements specified by the user and developer with the priorities based on different criteria. The screen layout is as shown below.

![Developer's Prioritization & Planning Phase](image)

**Figure 37** Plan incremental development stages
4.4.2.4 Integration and Validation: This is the last phase of requirement engineering. This phase consists of only one sub-section, resolved requirements, which list all the resolved requirements from the user and developer’s menu so that they can be considered in the next phase of software engineering. The screen layout is as shown below.

Figure 38 Resolved requirements
4.5 Implementation

The web based tool for Requirement Elicitation consists of three major components.

- Client PC running web browser.
- Web server running html front pages, java applets and application server for communicating with the database.
- Database server which stores all the information entered by various users and developers.

4.5.1 Front End Code

Following is the sample front end code written HTML. The only difference in all the programs is that the passing parameters have changed in all the programs.

```html
<!-- REPI Web Site -->
<!-- Demo Version - November 24, 1997 -->
<!-- Demo Version - August 14, 1998 -->
<!-- Copyright (c) 1997 -->
<!-- Author: Deepak Pandit -->
<!-- Umang Dave -->
<!-- Mariam Burmawalla -->
<!-- Email: tanik@homer.njit.edu -->

<HTML>

<HEAD>
  <META NAME="Author" CONTENT="Deepak Pandit">
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=iso-8859-1">

  <LINK REL="stylesheet" TYPE="text/javascript" HREF="REPI.CSS"
         TITLE="Style Sheet">

  <SCRIPT LANGUAGE="JavaScript" SRC="REPI.JS"> </SCRIPT>

</HEAD>
```
<SCRIPT LANGUAGE="JavaScript">
val = new Array();
temp = new Array();
var templen;
function getfromDB(id) {

cnt = 0;
temp = document.U_FF_1j.GetFromDataBase(id);
document.U_FF_1j.Stakeholder_Info_FirstName.value = temp[0];
document.U_FF_1j.Stakeholder_Info_LastName.value = temp[1];
document.U_FF_1j.Stakeholder_Info_Email.value = temp[2];
document.U_FF_1j.Stakeholder_Info_WorkPhone.value = temp[3];
}

function writetoDB(id, fname, lname, phone, email, type) {
var flag;
var cat='U';
for (i=0; i<type.length; i++)
if (typeof(i==0))flag = "E";
else if(i==1)
flag = "C";
else if(i==2)
flag = "M";
break;
}
document.U_FF_1j.UpdateDataBase(id, fname, lname, email, phone, flag, cat);
}
</SCRIPT>

<TITLE>User's Fact Finding Task 1: Identify relevant people</TITLE>
</HEAD>
<APPLET CODE="U_FF_1.class" NAME="U_FF_1" WIDTH = 10 HEIGHT = 10></APPLET>
<BODY LINK="BLUE" VLINK="BLUE" ALINK="White"
BACKGROUND="REPI_BK2.JPG">
<BIG><CENTER><STRONG>
  Task 1: Identify relevant people
</STRONG></CENTER></BIG>
</CENTER><IMG SRC="REPI_LN1.GIF" WIDTH="800" HEIGHT="5"></CENTER>

<FORM NAME="U_FF_1j" METHOD="GET">
<P ALIGN="Left">
This form is used to identify potential Stakeholders of this project.

Please provide the requested information for each Stakeholder of the project, that you can think of.

<table>
<thead>
<tr>
<th>ID:</th>
<th>Stakeholder_Info_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name:</td>
<td>Stakeholder_Info_FirstName</td>
</tr>
<tr>
<td>Last name:</td>
<td>Stakeholder_Info_LastName</td>
</tr>
<tr>
<td>Work Phone:</td>
<td>Stakeholder_Info_WorkPhone</td>
</tr>
<tr>
<td>E-mail:</td>
<td>Stakeholder_Info_Email</td>
</tr>
</tbody>
</table>

Categorize this person into one of these categories:
<table>
<thead>
<tr>
<th>Stakeholder Category</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>End User</td>
<td>&quot;End User&quot;</td>
</tr>
<tr>
<td>Customer</td>
<td>&quot;Customer&quot;</td>
</tr>
<tr>
<td>Management</td>
<td>&quot;Management&quot;</td>
</tr>
</tbody>
</table>

```html
<TABLE>
  <TR ALIGN="Center" COLSPAN="4">
    <INPUT TYPE="RADIO" NAME="Stakeholder_Category" VALUE="EndUser">
    <INPUT TYPE="RADIO" NAME="Stakeholder_Category" VALUE="Customer">
    <INPUT TYPE="RADIO" NAME="Stakeholder_Category" VALUE="Management">
  </TR>
  <TR ALIGN="Center" COLSPAN="4">
    <INPUT TYPE="Button" NAME="U_FF_1_Enter" VALUE="Enter Information" onClick="writetoDB(Stakeholder_Info_ID.value, Stakeholder_Info_FirstName.value, Stakeholder_Info_LastName.value, Stakeholder_Info_WorkPhone.value, Stakeholder_Info_Email.value, Stakeholder_Category))">
  </TR>
  <TR ALIGN="Center" COLSPAN="4">
    <INPUT TYPE="Button" NAME="U_FF_1_Get" VALUE="Get Information" onClick="getfromDB(Stakeholder_Info_ID.value))">
  </TR>
  <TR ALIGN="Center" COLSPAN="4">
    <INPUT TYPE="Reset" VALUE="Clear Form">
  </TR>
</TABLE>
```

</FORM>
</CENTER>
</BODY>
</HTML>
4.5.2 Java Code

Following is the sample code written in Java. This code is connected to the front end with the help of the applet. This program allows retrieval of information from the database. The difference in all the programs is that the passing parameters have changed.

    <!-- REPI Web Site Java code -->
    <!-- Demo Version - August 14, 1998 -->
    <!-- Copyright (c) 1998 -->
    <!-- -->
    <!-- Author: Umang Dave -->
    <!-- Mariam Burmawalla -->
    <!-- Email: tanik@homer.njit.edu -->

    import java.applet.*;
    import java.net.*;
    import java.util.*;

    public class U_FF_1 extends Applet{
        public String stServerName;
        public int iPortNumber;
        public String ServerReply;
        public DBClient dbclient;

        public String GetFromDataBase(String people_id) {
            stServerName = "cache.njit.edu";
            iPortNumber = 7001;
            DBClient dbclient = new DBClient(stServerName, iPortNumber);
            dbclient.ProcessCommand("S");
            ServerReply = dbclient.ProcessCommand("PEOPLE_FNAME, PEOPLE_LNAME, PEOPLE_EMAIL, PEOPLE_PHONE@people_m@PEOPLE_ID=\""+people_id.trim()+\"\"");
            System.out.println("This is the answer": ServerReply);
            StringTokenizer sttok = new StringTokenizer(ServerReply, ",");
            while(sttok.hasMoreTokens()) {
                System.out.println("Token is": sttok.nextToken());
            }
            return ServerReply;
public void updatePage() {
    StringTokenizer sttok = new StringTokenizer(ServerReply,"^");
    while(sttok.hasMoreTokens()) {
        System.out.println("Token is"+sttok.nextToken());
    }
}

public void UpdateDataBase(String people_id, String people_fname, String people_lname, String people_email,String people_phone, String people_type) {

    stServerName = "cache.njit.edu";

    iPortNumber = 7001;

    DBClient dbclient = new DBClient(stServerName,iPortNumber);

dbclient.ProcessCommand("U");

    ServerReply = dbclient.ProcessCommand("insert into people_m(people_id,people_fname,people_lname,people_email,people_phone,people_type)
     values("+people_id+","+people_fname+","+people_lname+","+people_email+","+people_phone+","+people_type+")");

    System.out.println("This is the answer"+ ServerReply);
}
}
4.5.3 Java Application Server Code

```java
import java.awt.*;
import java.sql.*;
import java.lang.*;
import java.util.*;
import java.net.*;
import java.io.*;

class DBAdmin {
    private static final boolean DEBUG = true;
    private Connection con;
    private DatabaseMetaData dbmetad;
    private String stCatalog = " ";
    private Statement SQLstatement;
    public int nRowNum, nColNum;
    DBAdmin(String stUrl, String stName, String stPwd) {
        String stStatus;
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con = DriverManager.getConnection("jdbc:oracle:mariam/m15502@logic");
            dbmetad = con.getMetaData();
            stCatalog = con.getCatalog();
            SQLstatement = con.createStatement();
            stStatus = "Establishing a connection with a DataBase";
        } catch (Exception e) {
            stStatus = "Connection Failed" + e.getMessage();
        }
        if (DEBUG) System.out.println(stStatus);
    }

    public boolean fnExecSQLUpdate(String stQuery) {
        try {
            int nTmpVar = SQLstatement.executeUpdate(stQuery);
            return true;
        }
    }
}
```
catch(SQLException se) {
    if (DEBUG) System.out.println(se.getMessage());
}
return false;

private ResultSet fnExecSQLSelect(String stFieldName, String stTableName, String stCondition) {
    String stQuery = "";
    ResultSet tmpres = null;
    try {
        stQuery = fnstQueryBuilder("SELECT", stFieldName.trim(),
                                stTableName.trim(), stCondition.trim());
        System.out.println("Say bye");
        tmpres = SQLstatement.executeQuery(stQuery.trim());
        System.out.println("Say bye1");
    } catch(SQLException se) {
        if (DEBUG) System.out.println(se.getMessage());
        if (DEBUG) System.out.println(se.getErrorCode());
    }
    return tmpres;
}

public String[][] fnGetFieldsDB(String stFieldName, String stTableName, String stCondition) {
    String stTmp = "";
    String stFieldVal[][];
    ResultSet tmpres;
    ResultSetMetaData tmpresmetdat;
    int nColumns = 0, nRows = 0;
    int i = 0;
    System.out.println("Bye1");
    tmpres = fnExecSQLSelect(stFieldName, stTableName, stCondition);
    if (tmpres == null) {
        } else { try {
            tmpresmetdat = tmpres.getMetaData();
            nColumns = tmpresmetdat.getColumnCount();
            System.out.println("ByeBye");
            if (nColumns >= 1) {
                String stColumnName[] = new String[nColumns];
                String stColumnType[] = new String[nColumns];
                for (i = 0; i < nColumns; i++) {

stColumnName[i] =
    tmpresmetdat.getColumnName(i+1);

stColumnType[i] =
    tmpresmetdat.getColumnTypeName(i+1);

System.out.println("Column type is "+stColumnType[i]);
}
while(tmpres.next()){
    nRows = nRows + 1;
    for(i=0;i<nColumns;i++){
        System.out.println("Column type again is "
            +stColumnName[i]+stColumnType[i]);

        if(stColumnType[i].equals("VARCHAR2")){
            System.out.println("Say Hi");
            stTmp = stTmp +
                tmpres.getString(stColumnName[i]) + "\n";
            // Each field value is separated by a space
        }

        if(stColumnType[i].equals("LONG")){
            stTmp = stTmp + Long.toString
                (tmpres.getLong(stColumnName[i])) + "\n";
            // Each field value is separated by a space
        }

        if(stColumnType[i].equals("BYTE")){
            stTmp = stTmp + Byte.toString
                (tmpres.getByte(stColumnName[i])) + "\n";
            // Each field value is separated by a space
        }

        if(stColumnType[i].equals("SHORT")){
            stTmp = stTmp + Short.toString
                (tmpres.getShort(stColumnName[i])) + "\n";
            // Each field value is separated by a space
        }

        if(stColumnType[i].equals("SINGLE")){
            stTmp = stTmp + Integer.toString
                (tmpres.getInt(stColumnName[i])) + "\n";
            // Each field value is separated by a space
        }

        if(stColumnType[i].equals("DOUBLE")){
            System.out.println("Column type is "+stColumnType[i]);
        }
    }
    System.out.println("\n");
}
stTmp = stTmp + Double.toString
tmpres.getDouble(stColumnName[i])) + "^";
//Each field value is separated by a space
}
}
if(stTmp.endsWith("^")){
    stTmp = stTmp.substring(0,stTmp.length()-1);
}
stTmp = stTmp.trim() + "%";
//Each Tuple is separated by an underscore
}
else{
    String stColumnType =
        tmpresmetad.getColumnTypeName(1);
    String stColumnName =
        tmpresmetad.getColumnName(1);
    while(tmpres.next()) {
        nRows = nRows + 1;
        if(stColumnType.equals("TEXT")) {
            stTmp = stTmp +
                tmpres.getString(stColumnName) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("LONG")) {
            stTmp = stTmp +
                Long.toString(tmpres.getLong(stColumnName))
                + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("BYTE")) {
            stTmp = stTmp + Byte.toString
                (tmpres.getByte(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("SHORT")) {
            stTmp = stTmp + Short.toString
                (tmpres.getShort(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("SINGLE")) {
            stTmp = stTmp + Integer.toString
                (tmpres.getInt(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("DOUBLE")){
            stTmp = stTmp + Double.toString
                (tmpres.getDouble(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("LONG")){
            stTmp = stTmp + Long.toString
                (tmpres.getLong(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("BYTE")) {
            stTmp = stTmp + Byte.toString
                (tmpres.getByte(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("SHORT")) {
            stTmp = stTmp + Short.toString
                (tmpres.getShort(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("SINGLE")) {
            stTmp = stTmp + Integer.toString
                (tmpres.getInt(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        if(stColumnType.equals("DOUBLE")){
            stTmp = stTmp + Double.toString
                (tmpres.getDouble(stColumnName)) + "%";
            //Each field value is separated by a space
        }
        stTmp = stTmp + "^";
        //Each Tuple is separated by a underscore
    }
}
if(stTmp.endsWith("^")){
    stTmp = stTmp.substring(0,stTmp.length()-1);
}
stTmp = stTmp.trim() + "%";
//Each Tuple is separated by an underscore
}
stTmp = stTmp + Double.toString
	(tmpres.getDouble(stColumnName)) + " ";
// Each field value is separated by a space

}

}

catch(SQLException se){}

try {

tmpres.close();

}
catch(SQLException se){}

stTmp = stTmp.trim();
if(stTmp.endsWith("%")){
    stTmp = stTmp.substring(0, stTmp.length()-1);
}
stFieldVal = new String[nRows][nColumns];
stFieldVal = fnFieldParser(stTmp, nRows, nColumns);
return stFieldVal;

}

private String fnstQueryBuilder(String stQueryType, String stFieldName, String stTableName, String stCondition){
    String stQuery = " ";
    if(stQueryType == "SELECT"){
        if(stCondition == " "){
            stQuery = "SELECT " + stFieldName + " FROM " + stTableName; // + ",";
        } else{
            stQuery = "SELECT " + stFieldName + " FROM " + stTableName + " WHERE " + stCondition; // + ",";
        }
    }
    else{
        if(DEBUG) System.out.println(stQuery); //Debugging
    }
    return stQuery;
}

private String[][] fnFieldParser(String stQueryResult, int nRows, int nCols){
    String stRows[] = new String[nRows];
    String stFieldVal[][] = new String[nRows][nCols];
    int nRowIndex = 0, nColIndex = 0;
    nRowNum = nRows;
    nColNum = nCols;
    StringTokenizer STToken1 = new StringTokenizer(stQueryResult, ",");
    while(STToken1.hasMoreTokens()){
stRows[nRowIndex] = STToken1.nextToken(); // Separating the Rows
StringTokenizer STToken2 = new StringTokenizer(stRows[nRowIndex], "\n");
while(STToken2.hasMoreTokens()){
   stFieldVal[nRowIndex][nColIndex] = STToken2.nextToken(); // Separating the columns in each row
   nColIndex = nColIndex + 1;
}
   nColIndex = 0;
   nRowIndex = nRowIndex + 1;
return stFieldVal;
}


class ApplicationServer extends Thread {
   private static final boolean DEBUG = true;
   public final static int DEFAULT_PORT = 7001;
   protected static int port=7001;
   protected ServerSocket server_port;
   protected ThreadGroup CurrentConnections;
   public ServerConnection c;
   protected Vector connections;
   protected ConnectionWatcher watcher;
   DBAdmin admin;
   static String stDBURL = "jdbc:oracle";
   static String stDBUsrName = "mariam";
   static String stDBPwd = "m15502@logic";
   static public void main(String args[]) {
      new ApplicationServer(port, stDBURL, stDBUsrName, stDBPwd);
   }
public ApplicationServer(int port, String stDBURL, String stDBUsrName, String
stDBPwd) {
   super("Server");
   admin = new DBAdmin(stDBURL, stDBUsrName, stDBPwd);
   try {
      server_port = new ServerSocket(port);
      System.out.println(port);
   }
   catch (IOException e) {System.out.println(e);}
   CurrentConnections = new ThreadGroup("Server Connections");
   connections = new Vector();
   watcher = new ConnectionWatcher(this);
   this.start();
}
public void run() {
   try {
      while(true) {

Socket client_socket = server_port.accept();
c = new ServerConnection(client_socket, CurrentConnections, 3, watcher,
    this, admin);
    // prevent simultaneous access.
synchronized (connections) {
        connections.addElement(c);
    }
}
}
catch (IOException e) {System.out.println(e);}
    System.exit(0);
}
public String fnDisplay(){
    String tmpString="";
    tmpString = c.getInfo();
    return tmpString;
}
public int getNumConnections(){
    return c.numberOfConnections;
}
}
class ServerConnection extends Thread {
    private static final boolean DEBUG = true;
    static int numberOfConnections = 0;
    protected Socket client;
    protected ConnectionWatcher watcher;
    protected DataInputStream in;
    protected PrintStream out;
    Connection con;
    public String stDBurl;
    public String stUsrName;
    public String stPwd;
    public ApplicationServer AS;
    DBAdmin admin;
    public ServerConnection(Socket client_socket, ThreadGroup CurrentConnections,
        int priority, ConnectionWatcher watcher, ApplicationServer AS, DBAdmin
        _admin) {
        super(CurrentConnections, "Connection number" + numberOfConnections++);
        this.setPriority(priority);
        client = client_socket;
        this.watcher = watcher;
        AS = AS;
        admin = _admin;
        try {
            in = new DataInputStream(client.getInputStream());
            out = new PrintStream(client.getOutputStream());
```java
} catch (IOException e) {
    try {
        client.close();
    } catch (IOException e2) {
        if (DEBUG) System.err.println("Exception while getting socket streams: " + e);
        return;
    }
} this.start();
}
public void run() {
    String inline;
    try {
        while (true) {
            inline = in.readLine(); // read in a line
            System.out.println("Printing inline" + inline);
            if (inline == null) break;
            inline = inline.trim();
            if (inline.toCharArray()[0] == 'S') {
                System.out.println("\n" + this.getInfo() + " has performed a SELECT Operation");
            }
            String stFieldValue[][]; // The String Array that stores the results of a query
            String stFieldName = " ";
            String stTableName = " ";
            String stCondition = " ";
            String stOutPut = " ";
            int ni, nj;
            out.println("DONE");
            inline = in.readLine();
            inline = inline.trim();
            System.out.println("Printing inline 2" + inline);
            String tokenizer = new StringTokenizer(inline, "@");
            try {
                while (tokenizer.hasMoreTokens()) {
                    stFieldName = tokenizer.nextToken().trim();
                    stTableName = tokenizer.nextToken().trim();
                    stCondition = tokenizer.nextToken().trim();
                }
            } catch (Exception e) {
            }
```

if(DEBUG) System.out.println(stFieldName + stTableName + stCondition);  //Debugging

stFieldVal = admin.fnGetFieldsDB(stFieldName.trim(), stTableName.trim(), stCondition.trim());
for(ni=0; ni<admin.nRowNum; ni++){
    for(nj=0; nj<admin.nColNum; nj++){
        stOutPut = stOutPut + stFieldVal[ni][nj] + "\n";
        System.out.println("Query value" + stFieldVal[ni][nj]);
        System.out.println("Query value" + stOutPut);
    }
}
if(stOutPut.endsWith("\n\n")){
    stOutPut = stOutPut.substring(0, stOutPut.length()-1);
}
stOutPut = stOutPut.trim() + ";%";
}
if(stOutPut.endsWith("%;")) {
    stOutPut = stOutPut.substring(0, stOutPut.length()-1);
}
if(DEBUG) System.out.println(stOutPut);  //Debugging

System.out.println(stOutPut + "SWED");
out.println(stOutPut.trim());
out.println("DONE");  
}
if(inline.toCharArray()[0] == 'U'){
    System.out.println("n" + this.getInfo() + " has performed an INSERT Operation");
    if(DEBUG) System.out.println("Preparing to write DataBase");  //Debugging
    out.println("DONE");  //Debugging
    inline = in.readLine();
    System.out.println(inline);
    inline = inline.trim();
    boolean boquery = admin.fnExecSQLUpdate(inline);
    if(DEBUG) System.out.println(inline + boquery);  //Debugging
    out.println("DONE");
}
    out.flush();
}

}

try {
    client.close();
}
}

catch (IOException e) { }

finally {
    try {
        synchronized (watcher) {Watcher.notify();
    }

} catch (IOException e2) {  
    synchronized (watcher) {Watcher.notify();
    }
public String getInfo() {
    return (client.getInetAddress().getHostName());
}

class ConnectionWatcher extends Thread {
    protected ApplicationServer server;
    protected ConnectionWatcher(ApplicationServer s) {
        super(s.getCurrentConnections(), "ConnectionWatcher");
        server = s;
        this.start();
    }
    public synchronized void run() {
        try {
        this.wait(10000);
            } catch (InterruptedException e) {
                System.out.println("Caught an Interrupted Exception");
        }
        synchronized(server.connections) {
            for(int i = 0; i < server.connections.size(); i++) {
                ServerConnection c;
                c = (ServerConnection)server.connections.elementAt(i);
                if (!c.isAlive()) {
                    System.out.println("Disconnecting from "+c.getInfo());
                    server.connections.removeElementAt(i);
                    i--;
                    c.numberOfConnections = c.numberOfConnections - 1;
                }
            }
        }
    }
}
CHAPTER 5
CONCLUSION AND FUTURE WORK

“Early defect fixes are typically to orders of magnitude cheaper than late defect fixes, and the early requirements and design defects typically leave more serious operational consequences” [CHRISTEL 92]. Designing a system to the satisfaction of the user's requirements is a most vital requirement in any development cycle. To meet this requirement accurate specification of user's needs and ideas is an important factor. All this can thus be achieved using the requirement elicitation process.

5.1 A Road towards Success

Requirement elicitation is the first steps towards a successful implementation of a project. The objective is to be able to integrate a collection of requirements and to establish their accuracy prior to design implementation – rather then during testing and evaluation. In addition, as we learn, the effects certain combinations of characteristics and relationships have on the development process, we can more readily identify potential difficulties and correct them before we are committed to an approach that maybe impracticable [HARWELL 93].

Requirement elicitation process aids the people in the army to specify their requirements in a more systematic manner. The objective of mapping the requirement elicitation process with the environment is:
1. Proper, precise and accurate specification of materials required for the manufacturing of the ammunition.

2. Specification of the effects of the manufacturing process on the environment, for example, the effect of cost, chemicals used in the production etc.

3. Proper communication between the people at the site of manufacturing and the army personnel who request for the ammunition production.

5.2 Future Work

With the advent of new technologies, our system can also be improved with more enhancement and additions. Following are few of the techniques that can be adopted to enhance the features of the system.

5.2.1 Knowledge based Result Generation using Expert System

To improve the decision making process, a mechanism internal to the system that can generate its own decisions has to be incorporated. This can be achieved with the help of C Language Integrated Production System (CLIPS).

CLIPS is an expert system tool developed by the Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center. CLIPS is designed to facilitate the development of software to model human knowledge or expertise. There are three ways to represent knowledge in CLIPS:
1. Rules, which are primarily intended for heuristic knowledge, based on experience.

2. Defunctions and generic functions, which are primarily intended for procedural knowledge.

3. Object-oriented programming also primarily intended for procedural knowledge.

The shell of CLIPS comprises of the basis elements, which form the heart of the system:

1. Fact-list and instance-list: Global memory for data

2. Knowledge-base: Contains all the rules, the rule-base

3. Inference engine: Controls overall execution of rules

A program written in CLIPS may consist of rules, facts, and objects. The inference engine decided which rules should be executed and when. A rule-based expert system written in CLIPS is a data-driven program where the facts, and objects if desired, are the data that stimulate execution via the inference engine. [GIARRATANO 97]

CLIPS also supports the six generally accepted features of object-oriented programming:

1. Classes

2. Message-handlers

3. Abstraction

4. Encapsulation

5. Inheritance

6. Polymorphism
Rules may pattern match on objects and facts [GIARRANTANO 97]. Various software's that use CLIPS are developed only using rules, only objects, or a mixture of objects and rules.

CLIPS can be integrated with other programming languages as C or Ada. It can be called from any procedural languages, perform its execution and transfer control back to the main program. It can also be used as stand-alone software. It is based on rules, which pattern-match on facts and objects and produce outputs [GIARRANTANO 97].

The various conditions required for the manufacturing of the ammunition have to be specified as rules in CLIPS. Based on these rules, the inference engine would draw decisions for the various requirements specified in the requirement elicitation process.

Thus CLIPS helps in a more accurate and faster decision making process.

5.2.2 Security Features
Incorporate security measures at all levels of the design. Each user may be provided with a login and password, based on which he can have an access to the system. This provides more reliable and accurate information.

5.2.3 Map other Areas of the Environmental Life Cycle
Requirement elicitation process can be used to specify requirements for all areas of the environmental life model namely,

1. Design,
2. Production,
3. Distribution,
4. Packaging and
4. Demanufacturing.
APPENDIX A

ENTITY RELATIONSHIP DIAGRAM OF REQUIREMENT ELICITATION PROCESS

E-R DIAGRAM FOR REPI

CL : Compliance Level
GL : Goal Name
AU : Authorized User ID
C&B : Cost & Benefit
Feas : Feasibility
Dep : dependency
D/T : date/Time
RT : Refers To
RB : Referred By
APPENDIX B
NORMALIZED TABLES FOR REQUIREMENT ELICITATION DATABASE

Table 1 Normalized Database

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Table Name</th>
<th>Attribute Name</th>
<th>Type &amp; Length</th>
<th>Key</th>
<th>Null/Not Null</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>People_m</td>
<td>people_id</td>
<td>Char(11)</td>
<td>PK</td>
<td>NN</td>
<td>User/Developer Identification Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_fname</td>
<td>Char(15)</td>
<td>NN</td>
<td></td>
<td>First name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_lname</td>
<td>Char(15)</td>
<td></td>
<td></td>
<td>Last name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_addr1</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td>Address Line 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_addr2</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td>Address Line 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_email</td>
<td>Char(27)</td>
<td></td>
<td></td>
<td>Email</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_phone</td>
<td>Char(12)</td>
<td></td>
<td></td>
<td>Phone Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_type</td>
<td>Char(2)</td>
<td>NN</td>
<td></td>
<td>User or Developer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_desc</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_category</td>
<td>Char(2)</td>
<td></td>
<td></td>
<td>Category of developer or user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>people_filename</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td>File name for information</td>
</tr>
<tr>
<td>2</td>
<td>Goal_m</td>
<td>goal_id</td>
<td>Char(11)</td>
<td>PK</td>
<td>NN</td>
<td>Goal Identification Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>goal_pid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>User/Developer Identification Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>goal_desc</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Goal Description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>goal_name</td>
<td>Char(40)</td>
<td></td>
<td></td>
<td>Goal name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>goal_filename</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td>File name for goal description</td>
</tr>
<tr>
<td>3</td>
<td>Req_m</td>
<td>req_id</td>
<td>Char(11)</td>
<td>PK</td>
<td>NN</td>
<td>Requirement Identification Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>req_gid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>Goal Identification #</td>
</tr>
<tr>
<td>Sr. No.</td>
<td>Table Name</td>
<td>Attribute Name</td>
<td>Type &amp; Length</td>
<td>Key</td>
<td>Null/Not Null</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>------------------</td>
<td>---------------</td>
<td>-----</td>
<td>---------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>req_pid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td></td>
<td>User/Developer Identification Number</td>
</tr>
<tr>
<td></td>
<td>req_title</td>
<td>Char(50)</td>
<td></td>
<td></td>
<td></td>
<td>Requirement Title</td>
</tr>
<tr>
<td></td>
<td>req_desc</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td></td>
<td>Requirement description</td>
</tr>
<tr>
<td></td>
<td>req_category</td>
<td>Char(25)</td>
<td></td>
<td></td>
<td></td>
<td>Category</td>
</tr>
<tr>
<td></td>
<td>req_valid</td>
<td>Char(1)</td>
<td></td>
<td></td>
<td></td>
<td>Validity of the requirement</td>
</tr>
<tr>
<td></td>
<td>req_aui</td>
<td>Char(11)</td>
<td></td>
<td></td>
<td></td>
<td>Authorization Identification Number</td>
</tr>
<tr>
<td></td>
<td>req_importance</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>Importance of requirement</td>
</tr>
<tr>
<td></td>
<td>req_understanding</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>Understanding of requirement</td>
</tr>
<tr>
<td></td>
<td>req_costlevel</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>Costlevel of requirement</td>
</tr>
<tr>
<td></td>
<td>req_deplevel</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>Dependence level of requirement</td>
</tr>
<tr>
<td></td>
<td>req_uppriority</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>User priority of requirement</td>
</tr>
<tr>
<td></td>
<td>req_dpriority</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>Developer priority of requirement</td>
</tr>
<tr>
<td></td>
<td>req_complevel</td>
<td>Char(25)</td>
<td></td>
<td></td>
<td></td>
<td>Compliance level of requirement</td>
</tr>
<tr>
<td></td>
<td>req_status</td>
<td>Char(25)</td>
<td></td>
<td></td>
<td></td>
<td>Status of requirement</td>
</tr>
<tr>
<td></td>
<td>req_type</td>
<td>Char(2)</td>
<td></td>
<td></td>
<td></td>
<td>Requirement type</td>
</tr>
<tr>
<td></td>
<td>req_filename</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td></td>
<td>Filename to store requirement information</td>
</tr>
<tr>
<td></td>
<td>req_why</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td></td>
<td>Why the requirement is required</td>
</tr>
<tr>
<td></td>
<td>req_reason</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td></td>
<td>Reason of requirement</td>
</tr>
<tr>
<td></td>
<td>req_risk</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td></td>
<td>Risk involved in the requirement</td>
</tr>
<tr>
<td></td>
<td>req_feasible</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td></td>
<td>Feasibility of the requirement</td>
</tr>
</tbody>
</table>
Table 1 (continued) Normalized Database

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Table Name</th>
<th>Attribute Name</th>
<th>Type &amp; Length</th>
<th>Key</th>
<th>Null/Not Null</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>req_costben</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td></td>
<td>Cost &amp; benefit of the requirement</td>
</tr>
<tr>
<td></td>
<td>req_dummy</td>
<td>Char(3)</td>
<td></td>
<td></td>
<td></td>
<td>Dummy variable required</td>
</tr>
<tr>
<td>4</td>
<td>Survey_</td>
<td>survey_id</td>
<td>Char(11)</td>
<td>PK</td>
<td>NN</td>
<td>Survey Identification Number</td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>survey_pid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>User/Developer Identification Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>survey_name</td>
<td>Char(40)</td>
<td></td>
<td></td>
<td>Survey Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>survey_desc</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Survey description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>survey_fileaname</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td>Filename to store survey information</td>
</tr>
<tr>
<td>5</td>
<td>Const_</td>
<td>const_pid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>User/Developer Identification Number</td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>const_name</td>
<td>Char(40)</td>
<td></td>
<td>NN</td>
<td>Constraint name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>const_info</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Constraint information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>const_asses</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Accessibility of the constraint</td>
</tr>
<tr>
<td></td>
<td></td>
<td>const_fileaname</td>
<td>Char(20)</td>
<td></td>
<td></td>
<td>Filename to store constraint information</td>
</tr>
<tr>
<td>6</td>
<td>Model_</td>
<td>model_pid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>User/Developer Identification Number</td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>model_domain</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Domain of the model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model_arch</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Architecture of the model</td>
</tr>
<tr>
<td>7</td>
<td>Scenario_</td>
<td>scen_id</td>
<td>Char(11)</td>
<td>PK</td>
<td>NN</td>
<td>Scenario Identification Num.</td>
</tr>
</tbody>
</table>
### Table 1 (continued) Normalized Database

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Table Name</th>
<th>Attribute Name</th>
<th>Type &amp; Length</th>
<th>Key</th>
<th>Null/Not Null</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>scen_gid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>Goal Identification Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>scen_pid</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>User/Developer Identification Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>scen_name</td>
<td>Char(20)</td>
<td></td>
<td>NN</td>
<td>Scenario Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>scen_desc</td>
<td>Char(200)</td>
<td></td>
<td></td>
<td>Scenario description</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Scenario_det</td>
<td>scen_did</td>
<td>Char(11)</td>
<td>FK</td>
<td>NN</td>
<td>Scenario Identification Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>scen_event</td>
<td>Char(25)</td>
<td></td>
<td>Scenario Event</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>scen_action</td>
<td>Char(25)</td>
<td></td>
<td>Scenario Action</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>scen_reaction</td>
<td>Char(25)</td>
<td></td>
<td>Scenario Reaction</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C

120mm M829E3 RELATED DATA IN REQUIREMENT ELICITATION PROCESS

Table 2: People Master

<table>
<thead>
<tr>
<th>People ID</th>
<th>People First Name</th>
<th>People Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1000</td>
<td>Mark</td>
<td>Healey</td>
</tr>
<tr>
<td>U2000</td>
<td>Murat</td>
<td>Tanik</td>
</tr>
<tr>
<td>U3000</td>
<td>Franz</td>
<td>Kurfess</td>
</tr>
<tr>
<td>D1000</td>
<td>Umang</td>
<td>Dave</td>
</tr>
<tr>
<td>D2000</td>
<td>Deepak</td>
<td>Pandit</td>
</tr>
</tbody>
</table>

Table 3: Goal Master

<table>
<thead>
<tr>
<th>Goal ID</th>
<th>Goal Name</th>
<th>People ID</th>
<th>Goal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Receive Materials</td>
<td>U1000</td>
<td>Receive materials for cartridge manufacturing from external sources</td>
</tr>
<tr>
<td>202</td>
<td>Assembly of gathered materials</td>
<td>U2000</td>
<td>Assembling of the components required for cartridge manufacturing</td>
</tr>
<tr>
<td>203</td>
<td>Testing of each process</td>
<td>U2000</td>
<td>Testing of each process involved in the manufacturing of the cartridge</td>
</tr>
<tr>
<td>204</td>
<td>Applying of coating</td>
<td>U1000</td>
<td>Applying of coating to the components of the cartridge</td>
</tr>
<tr>
<td>205</td>
<td>Weight Verification</td>
<td>U3000</td>
<td>Verification of weight after the cartridge is assembled before it is shipped</td>
</tr>
</tbody>
</table>

Table 4: Scenario Master

<table>
<thead>
<tr>
<th>Scenario ID</th>
<th>Scenario Name</th>
<th>Goal ID</th>
<th>People ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Assembly of projectile</td>
<td>202</td>
<td>U2000</td>
</tr>
<tr>
<td>112</td>
<td>Greasing of O-ring</td>
<td>203</td>
<td>U2000</td>
</tr>
<tr>
<td>113</td>
<td>Primer Test</td>
<td>203</td>
<td>U2000</td>
</tr>
<tr>
<td>114</td>
<td>Installation</td>
<td>202</td>
<td>U2000</td>
</tr>
<tr>
<td>115</td>
<td>Stake primer</td>
<td>203</td>
<td>U2000</td>
</tr>
<tr>
<td>106</td>
<td>Receive materials</td>
<td>201</td>
<td>U1000</td>
</tr>
<tr>
<td>107</td>
<td>Receive projectile</td>
<td>201</td>
<td>U1000</td>
</tr>
</tbody>
</table>
Table 5  Requirement Master

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Requirement Name</th>
<th>Goal ID</th>
<th>People ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>UR24</td>
<td>Receive annular bag</td>
<td>201</td>
<td>U1000</td>
</tr>
<tr>
<td>UR26</td>
<td>Test sabot, penetrator and cartridge for toch up</td>
<td>203</td>
<td>U2000</td>
</tr>
<tr>
<td>UR7</td>
<td>Retaining ring from storage</td>
<td>201</td>
<td>U1000</td>
</tr>
<tr>
<td>UR11</td>
<td>Base coat</td>
<td>201</td>
<td>U1000</td>
</tr>
<tr>
<td>UR14</td>
<td>Stencil table</td>
<td>201</td>
<td>U1000</td>
</tr>
<tr>
<td>UR16</td>
<td>Testing of inert materials</td>
<td>203</td>
<td>U2000</td>
</tr>
<tr>
<td>UR20</td>
<td>Test retaining ring</td>
<td>203</td>
<td>U2000</td>
</tr>
<tr>
<td>UR2</td>
<td>Projectile material</td>
<td>201</td>
<td>U1000</td>
</tr>
<tr>
<td>DR1</td>
<td>Entry of inert materials</td>
<td>201</td>
<td>D1000</td>
</tr>
<tr>
<td>DR2</td>
<td>Latest machinery</td>
<td>202</td>
<td>D2000</td>
</tr>
</tbody>
</table>
REFERENCES


[OBJA 95] Object Agency Semantic Networks for Object Oriented software engineering.


(14 Feb, 1998)


