A Requirements Engineering Tool Based on Use Cases

A Manuscript
Submitted to
The Department of Computer Science
and the Faculty of the
University of Wisconsin-La Crosse
La Crosse, Wisconsin

By
Jenny Gijo
in Partial Fulfillment of the
Requirements for the Degree of
Master of Software Engineering
August, 2014
A Requirements Engineering Tool Based on Use Cases

By Jenny Gijo

We recommend acceptance of this manuscript in partial fulfillment of this candidate’s requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

Dr. Kasilingam Periyasamy
Examination Committee Chair

Date

Examination Committee Member

Date

Examination Committee Member

Date
Abstract


The outcome of a requirements engineering process is a structured document that describes in detail all requirements of a software system. The IEEE recommends using a well-defined format for writing requirements so that software designers and developers can easily understand the requirements. When the programming paradigm shifted from functional to object-oriented approach, the IEEE recommended format needed to be changed to match closely with object-oriented design, especially the one developed using UML. Though the IEEE recommended format is independent of the design methodology, deriving an OO design from functional requirements itself is a challenge. The shift in the programming paradigm made some researchers develop use case based requirements. This manuscript describes the development of an editor that supports writing a structured requirements document using use cases. In addition to providing support for writing the document, the tool also outputs an estimated cost using the use case point approach. Most information for this calculation is extracted directly from the document and hence saves a lot of time for users.
Acknowledgements

I would like to express my sincere appreciation to my advisor, Dr. Kasi Periyasamy for his guidance, encouragement and continuous support through the development of this project. The extensive knowledge, vision, and creative thinking of Dr. Kasi have been the source of inspiration for me. His patience in discussion and reviewing encouraged me to constantly improve the user experience delivered in this application.

I would like to thank the rest of my professors in the Computer Science Department at the University of Wisconsin-La Crosse for everything I learnt from you.

Finally, I would be remiss if I did not express my most sincere gratitude to my husband Gijo Philip, and children Amy and Emily, for their continual patience, extended sacrifices, understanding, and encouragement during the tenure of my studies.
# Table of Contents

Abstract .............................................................................................................................. iii  
Acknowledgements .......................................................................................................... iv  
Table of Contents ............................................................................................................ v  
List of Figures .................................................................................................................. vi  
Glossary ............................................................................................................................. vii  
1. Introduction ..................................................................................................................1  
   1.1 Background Information ..........................................................................................1  
   1.2 Motivation .............................................................................................................2  
2. Requirements .............................................................................................................2  
   2.1 Use Case Model ..................................................................................................... Error! Bookmark not defined.  
   2.2 Use Case Narratives Documents .......................................................................... 6  
   2.3 Cost Estimation Using Use Case Points .............................................................. 8  
3. Life Cycle Model Used ............................................................................................. 10  
4. Architectural Design ................................................................................................. 10  
   4.1 Use Case Diagram ................................................................................................. 10  
   4.2 Class Diagram ...................................................................................................... 12  
   4.3 Components of the Tool ...................................................................................... 12  
   4.4 Tool Support for Cost Estimation ....................................................................... 16  
5. Implementation & Testing ........................................................................................ 18  
6. Conclusion .................................................................................................................. 20  
7. Future Expansion ...................................................................................................... 20  
8. References .................................................................................................................. 21  
Appendix .......................................................................................................................... 22
List of Figures

Figure 1: Structure of Functiona1 requirements document Error! Bookmark not defined.
Figure 2: Use case diagram for a simple ATM System ...................................................... 4
Figure 3: Use case narrative for Withdraw ................................................................. 5
Figure 4: Document template for use case narratives ................................... 6
Figure 5: Use case diagram ................................................................................. 11
Figure 6: Class diagram ..................................................................................... 12
Figure 7: Main screen of the Requirements Editor ........................................ 13
Figure 8: Adding and Editing Assumptions .................................................. 14
Figure 9: Uploading Use Case Diagrams.......................................................... 15
Figure 10: Adding and Editing Use Cases ......................................................... 16
Figure 11: Main screen for Use Case Point calculation ........................................ 17
Figure 12: Technical Factors in the Calculation of Use Case Points ........ 18
Figure 13: Preamble Tab.................................................................................. 22
Figure 14: Title Page Tab .............................................................................. 23
Figure 15: Introduction Tab ............................................................................ 23
Figure 16: Assumption Tab ............................................................................ 24
Figure 17: Use Case Diagram Tab ............................................................... 24
Figure 18: Use Case-Use Case Definition Tab .............................................. 25
Figure 19: Use Case-Pre Condition Tab .......................................................... 25
Figure 20: Use Case-Post Condition Tab ......................................................... 26
Figure 21: Use Case-Successful Scenario Tab ............................................... 26
Figure 22: Use Case-Includes/Excludes Tab .................................................... 27
Figure 23: Use Case-Exceptions Tab ............................................................... 27
Figure 24: Use Case-Additional Remarks ...................................................... 28
Figure 25: Reference Tab .............................................................................. 28
Figure 26: Actor Cassification Tab ................................................................. 29
Figure 27: Use Case Cassification Tab ............................................................ 29
Figure 28: Use Case Parameters Tab .............................................................. 30
Figure 29: Environmental Factors Tab ............................................................ 30
Figure 30: Technical Factors Tab ................................................................. 31
Figure 31: Calculate Efforts Tab ................................................................... 31
Glossary

**IEEE**
IEEE is the world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity. IEEE and its members inspire a global community through IEEE's highly cited publications, conferences, technology standards, and professional and educational activities.

**UML**
The Unified Modeling Language™ UML is a visual language for specifying, constructing, and documenting the artifacts of software-intensive systems. Complex software designs difficult for you to describe with text alone can readily be conveyed through diagrams using UML.

**XML**
Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards. The design goals of XML emphasize simplicity, generality, and usability over the Internet.
1. Introduction

The importance of requirements engineering in software development has been cited in many publications including in IEEE standards such as the IEEE Recommended Practice for Software Requirements Specification [1]. As evident from these publications, the structure of the requirements document plays a key role in ensuring that requirements are described in a coherent manner. This further enables software developers validate the requirements. As a result, tool support for developing requirements documents becomes necessary in order to help requirements specialists focus on writing requirements rather than concentrating on the structure of the document and the format of requirements.

1.1 Background Information

In the past, the IEEE recommended format [1] has been used in many projects for writing requirements specifications. Its popularity comes from the structure of document and the format of requirements presentation that is easier to read and understand, and perform analysis such as cost estimation and testability. Recently, the use case model is used for capturing requirements. Use case model is a part of UML (Unified Modeling Language) [2], a standardized design notation for the design and development of object-oriented software. Its simplicity in terms of notations and expressiveness makes it easier to use both by customers and developers. A use case diagram depicts use cases (similar to functional requirements) in a diagrammatic way but the details of such use cases are written in a separate document called Use Case Narratives document. Unlike the IEEE recommended format, there is no standard for writing use case narratives. However, most developers use some sort of a structure similar to that of an IEEE recommended format. Consequently, the use case narratives document becomes the requirements document.
1.2 Motivation

Use Case narratives are a required part of a Use Case document for the understanding of a Use Case diagram. Because of the formatting requirements of a use case document, it takes more time to write one. Consequently, the use case developer tends to spend more time in formatting rather than focusing on the use cases themselves. It is therefore evident that tool support is necessary for developing a structured use case document. Such a tool must take care of formatting considerations so that the use case developer will focus on the application domain. This is the motivation of the current work – developing a tool that assists a user develop a structured use case diagram and the associated narratives.

The rest of this manuscript describes the design and development of a tool for use case based requirements engineering.

2. Requirements

Requirements engineering is an important aspect of software development. There are many projects which have failed leading to huge losses because requirements engineering was not performed in the beginning. There is a tendency in the software development workplace to start coding as soon as possible. It is only late in the project that realization takes place that the project was not moving in the right direction. This is because requirements engineering was not done well before the start of the project.

There are 2 types of requirements- functional and non-functional requirements. Functional requirements are those that define the expected services and scope of the system. Non-functional requirements define the look and feel, performance, security and other factors. The main focus for this manuscript is the functional requirements.

Requirements are collected with the help of interviews with the users or stakeholders, research, brainstorming, etc. IEEE had a standard format for the writing the functional
requirement documents as shown in Figure 1. Each functionality of the software was explained with the help of a structure description.

Each functional requirement is given in the following format:

- **Index:**
- **Name:**
- **Purpose:**
- **Input parameters:**
- **Action:**
- **Output parameters:**
- **Exceptions:**
- **Remarks:**
- **Cross-references:**

**Figure 1. Structure of Functional requirements document**

A tool was developed for the generation of the functional requirements document by Ben Garbers and Dr. Kasi Periyasamy[7] at University of Wisconsin La Crosse. It was aimed for project managers to easily create a structured functional requirements document. This tool was made available to the public and received good response. Later, with the importance of Object-Oriented Programming, the requirements document was created using the Use Case Model. The Use Case Diagram by itself does not give sufficient information to the requirements of a project. The Use Case Narratives document was created with a structure similar to one suggested by IEEE for the functional requirements document. This document gives a clear explanation of the use case model and avoids confusion among the developers and stakeholders. The tool explained in this manuscript aids in generation of the use case narratives document in a structured format. This helps the user concentrate on the requirements and development of the use case model rather than the formatting of the narratives document. The following sections explain the Use Case Model and the Use Case Narratives documents.

### 2.1 Use Case Model
A use case is defined to be “a set of actions performed by a system, which yields an observable result that is of value for one or more stakeholders.”[2] An actor is someone or something that interacts with the system. A use case diagram typically describes the interactions between actors and use cases of the system. The relationships between use cases and between actors are also depicted in a use case diagram. Figure 2 shows a use case diagram for a simple ATM system involving only four transactions.

Figure 2. Use Case Diagram for a simple ATM System

As seen in Figure 2, the use case diagram itself does not provide all details of interactions or relationships between its elements. For example, when a customer invokes the ‘Withdraw’ use case, the customer needs to provide some input parameters and get a confirmation from the system whether or not the action is successful. In case of failure, the customer must get a detailed message explaining the reasons of the failure. The UML notation for use cases does not include any such additional information. It is up to the designer to include additional documentation that supports the use case diagram. This additional documentation, called Use Case Narrative, is expected to include details of all elements in the use case diagram. The use case narrative for ‘Withdraw’ is shown in Figure 3. There is no standard or recommended format for a use case narrative; the one shown in
Figure 3 is created to resemble the format of a functional requirement recommended by IEEE, but with few modifications. For example, the narrative includes pre and post-conditions as well as the relationships between actors and other use cases.

<table>
<thead>
<tr>
<th>Use Case #:</th>
<th>ATM-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case name:</td>
<td>Withdraw</td>
</tr>
<tr>
<td>Purpose:</td>
<td>To withdraw a positive amount from an account.</td>
</tr>
<tr>
<td>Scope:</td>
<td>System</td>
</tr>
<tr>
<td>Priority:</td>
<td>Very high</td>
</tr>
<tr>
<td>Primary actors:</td>
<td>Customer</td>
</tr>
<tr>
<td>Secondary actors</td>
<td>None</td>
</tr>
<tr>
<td>Input parameters</td>
<td>Amount to be withdrawn</td>
</tr>
<tr>
<td>Output parameters</td>
<td>None</td>
</tr>
<tr>
<td>Precondition:</td>
<td>User must have logged into an account.</td>
</tr>
<tr>
<td></td>
<td>Amount must be numeric and must be positive.</td>
</tr>
<tr>
<td></td>
<td>Amount must be less than or equal to the available balance in the account.</td>
</tr>
<tr>
<td>Post-condition:</td>
<td>If successful, the given amount is subtracted from the balance of the account.</td>
</tr>
</tbody>
</table>

Successful scenario:
1. User requests for withdrawal and provides a positive amount.
2. User request and amount parameters are validated.
3. If the parameters are valid, the current balance is updated by subtracting the amount from the balance. The updated account is stored back into the database.

Is this use case extended? No
  If yes, what are the extensions?

Does this use case extend another use? No
  If yes, what are they?

Does this use case include another use case? Yes
  If yes, what are they? Update account

Is this use case included in another use case? No
  If yes, what are they?

Exceptions: User did not login into any account.
  Format error in amount parameter.
  Amount is zero or negative.
  Amount is more than the available balance in the account.

Additional remarks: None

Figure 3. Use Case Narrative for Withdraw

2.2 Use Case Narratives Document
In addition to supporting a thorough understanding of the use cases of the system, it was decided to make the use case narratives document serve as a requirements document for the system. In this regard, the structure shown in Figure 4 has been proposed for the use case narratives document. This structure is somewhat similar to the one recommended by IEEE and so users of the IEEE template will have a smooth transition to the use case based document.

The Use Case Narratives document consists of the following:

i. **Title Page**
   The title page consists of the title of the project followed by the name of the author of the document, the date and the version of the document.

ii. **Introduction / Problem Description**
   This section starts with a fresh page. The problem description which is an overall description of what is the requirement of the project is described here.

iii. **Assumptions**
There may be various factors to the project that are not well defined. So the project manager will consider some assumptions. The assumptions are detailed in this section.

iv. **Use Case Diagram**

The use case diagrams come in this section. If it is a large project with a lot of functionality in it, there may be more than one use case diagram.

v. **Use Case Narratives**

Each use case in the use case diagram is expanded using the use case narratives. The use case narratives make the understanding of the use case clear with the input and output parameters, pre and post conditions, exceptions, etc. It helps in avoiding ambiguity in the use case. Figure 3 shows the sections of the use case narratives.

Each use case has the use case number, use case name, purpose, scope, priority, primary actors, secondary actors, input parameters, output parameters, preconditions, postconditions, successful scenario, inclusions, extensions, exceptions and additional remarks.

vi. **Reference**

The use case narratives document ends with the references.
2.3 Cost Estimation using Use Case Points

In addition to capturing requirements in a structured format, the tool also provides support for estimating cost of development using use case points. Cost estimation using use case points was coined based on function point cost estimation model for procedural software products [5]. Karner [5] proposed a use case based approach for cost estimation. Since then, a lot of researchers have extended Karner’s approach. Dr. Kasi Periyasamy and Aditi Ghode of University of Wisconsin - La Crosse also extended this approach. The tool implements the cost estimation algorithm that is described by Periyasamy and Ghode[6].

An important feature of the tool is its support for cost estimation using use case points. At the core, the methodology for cost estimation is based on how much coding effort is required for implementing various constructs as described in the use case document. Using this concept, a weight is assigned to each construct in the document. The calculation depends on five factors: (1) actors and their classifications, (2) use cases and their classifications, (3) use case narratives, (4) technical factors, and (5) environment factors. The steps for calculating the estimated cost are given below:

1. **Assign weight to actors**: Classify actors into seven categories: Very Simple, Simple, Less Average, Average, Complex, Very Complex and Most Complex. The classification depends on the number of associations the actor has. After the classification, each actor is assigned a weight based on the classification. Accordingly, each very simple actor is assigned a weight of 0.5, the weight is increased by 0.5 towards the other end of the classification spectrum. At the end of this step, the unadjusted actor weight is calculated by summing up the weights for each actor.

2. **Assign weights to use cases**: This step is very similar to the previous step except that it is based on use cases. Like actors, use cases are also classified based on the number of associations but the classification has only four categories: Simple, Average, Complex and Most Complex. The weights for these classifications are 0.5, 1, 2 and 3 respectively. The conclusion of this step ends up in unadjusted use case weight that is the sum of weights of all use cases.
3. **Assign weights to use case narrative parameters**: In the third step, each use case narrative is analyzed and a weight is assigned for each important construct in the use case narrative. Accordingly, every input parameter is assigned a weight of 0.1, every output parameter is assigned a weight of 0.1, every predicate in a pre- or post-condition is assigned a weight of 0.1, every action described in the successful scenario is assigned a weight of 0.2, and finally every exception is assigned a weight of 0.1. As a result, every use case narrative has its own weight that is the sum of all weights assigned to these constructs. The unadjusted use case narrative weight becomes the sum of weights of all use case narratives.

4. **Assign weights to technical factors**: In this step, the user will be prompted with 13 technical factors and asked to select one or more technical factors that closely describe the type of technical work involved in the implementation of the use cases. These technical factors are taken from Karner’s first document on use case points [5]. As given in his document, each selection of a technical factor is associated with a predefined weight; see [6] for details.

5. **Assign weights to environmental factors**: Like technical factors, this step tries to assign environment factors that closely fit the intended product. These environment factors also chosen from Karner’s model and there are eight environment factors.

6. **Calculate use case points**: After collecting the weights from various sections of the document, the final step first involves the calculation of unadjusted use case points by adding weights due to actors, weights due to use cases and weights due to use case narratives. The sum of all these weights is called Unadjusted Use Case Points (UUCP). The contribution of technical factors towards the cost estimation calculation is given by the formula

\[
TCF = 0.6 + (0.01 \times Ti), \text{ for all } 1 \leq i \leq 13
\]

Similarly, the contribution of environmental factors towards the cost estimation is calculated by the formula

\[
EF = 1.4 + (-0.03 \times Ei), \text{ for all } 1 \leq I \leq 8.
\]
The final step is the calculation of the Adjusted Use Case Points (AUCP) which is computed as

$$\text{AUCP} = \text{UUCP} + \text{TCF} + \text{EF}.$$  

Notice that this summation gives the estimated cost in terms of use case points only. The actual efforts (say in man months) should be calculated by multiplying UUCP by a factor N that represents the number of man hours expected to implement one use case point. Practical evidences show that values of N range from 5 to 20.

Complete details of the use case point calculation with case study can be found as described by Periyasamy and Ghode [6].

3. Life Cycle Model Used

The project was developed using a combination of Incremental and Agile approaches. The requirements of the project were collected at the beginning of the project. The use case model and class diagram were created at the start of the project. The implementation of the project was done with an agile approach. The Agile Life Cycle model was used to get constant customer feedback. The project was done in small phases achieving to complete each user story demonstrate it, make the necessary changes and then moving on to the next phase. When the initial research for the project was done, I had initially decided to use Jasper for the pdf generation. But about a year later I found that a new package was available for pdf generation that was more convenient to use. In today’s fast changing environment, the requirements and technology keep changing so fast that the Agile Life Cycle Model is apt for software development to keep at par with the latest technology and changing requirements.

4. Architectural Design

4.1 Use Case Diagram
The basic functionality of the software was put down in the form of a Use Case Diagram. The main actors were the User and File System. The Use cases were

1. Creating a new file
2. Opening a file
3. Saving a file
4. Generating the PDF
5. Calculating EUCP

The use case diagram is shown in Figure 5.
4.2 Class Diagram

The classes for this project are the file, titlepage, usecase, usecasedef, preamble and actor. The class diagram is shown below with all the attributes.

![Class Diagram](image)

**Figure 6. Class Diagram**

4.3 Components of the Tool

The design of the use case based requirements editor mimicked the requirements editor based on the IEEE recommended practices format [7], also developed at the University of Wisconsin-La Crosse. The focus of both editors is on providing interactive dialogs for entering requirements and estimating cost of development based on the requirements. The IEEE format based editor used Function Point approach for cost estimation while the use
case based editor uses Use Case Point approach. Besides, the structure of the document output by each editor is also different.

![Figure 7. Main Screen of the Requirements Editor](image)

Figure 7 shows the main screen of the use case based requirements editor. Like any other editor, the main screen provides functionalities for file operations such as “Open”, “Save” and “Close”. The tabs indicate various sections of a requirements document as given in Figure 7. In addition, the ‘Preamble’ tab lets the user input all actors (external entities) along with their classifications (primary or secondary). This information will be shared (and used) by all use cases. The tool also validates actors’ information and ensures uniqueness of actor names. Notice that the requirements editor saves the current document in an XML file (selected under ‘File name’) and produces a formatted output in PDF (selected under ‘Output PDF File name’). The XML file can be used by other tools as well. The ‘Title Page’ and ‘Introduction’ tabs are meant for providing title information and problem description respectively.
The fourth tab in main screen provides dialogs for introducing and editing assumptions. There are two reasons why we selected a separate tab for providing assumptions: First, adding and editing assumptions require more space and so it may clutter the space if it is included with other tabs such as ‘Introduction’. Second, assumptions may be edited as the requirements are written but introduction and preamble information might not change frequently. So it is preferable to keep assumptions under a separate tab. Figure 8 shows an expanded view of the ‘Assumptions’ tab.

![Figure 8. Adding and Editing Assumptions](image)

The two panels on the right side are reserved for adding/editing assumptions (the one on the top) and displaying assumptions (the one at the bottom). Numbers are added automatically to the assumptions for cross-references. They are also adjusted automatically when assumptions are deleted or moved. To facilitate easy editing including moving of assumptions, a tree view of assumptions is provided on the left panel. The design of this tab and the activities were impacted by the feedback obtained from users of the IEEE format based editor [7], developed earlier.
The most important sections of the requirements are the use cases and the use case diagram. A user of the tool can upload any number of use case diagrams as shown in Figure 9.

![Figure 9. Uploading Use Case Diagrams](image)

Uploaded diagrams can be selected individually for viewing. They can also be deleted at any time. When saving the requirements, the image files are stored in the same directory where the XML file is saved thus saving a local copy of each image. This way, all diagrams pertinent to a current requirements document are all kept in the same folder. The user need not navigate to the original location where the image was first selected.

The dialog for entering use cases is designed to be somewhat similar to that of adding assumptions. This improves the usability of the tool so that the user will have one consistent view. Figure 10 shows the dialog and associated activities for a use case. Each use case is viewed under the tab ‘Use Cases’ when selected from the tree view in the left panel. The
association of actors with a use case is selected from the dropdown box of actors; these actors are already entered under ‘Preamble’ tab.

Figure 10. Adding and editing use cases.

There are additional tabs for a use case as shown in Figure 7 that correspond to the various components of a use case. These tabs provide a similarly structured dialog as in Assumptions. The use case number is ensured to be unique across the entire document.

### 4.4 Tool support for Cost Estimation

The use case document editor has a built-in calculator for cost estimation using use case points. It extracts all information regarding actors, use cases, associations between actors and use cases, and use case narratives from the document created by the user. For technical factors and environmental factors, users will be prompted for selections.

Figure 11 shows the main screen of use case point calculation. This screen is displayed
when a user selects the use case analysis option from the menu bar on the top. As seen in Figure 11, the extracted information on actors, use cases and use case parameters are displayed when the user selects the appropriate tab. Notice that the extracted information is not editable. It is for display purpose only for cross checking the calculation. Figure 9 shows the screen for selecting technical factors. The user will be prompted to select the technical factors that closely fit the product. The assignments of weights to the selected factors are not shown to the user since the user does not need to know those details. When the user selects the environmental factors tab, a similar screen appears.

The final tab in this section displays the computed AUCP values. If the user does not select any technical or environmental factor, then the final calculation will not be displayed. The tool thus forces the user to go through the technical and environmental factors before looking at the final calculation. Technically, the methodology does not require any technical or environmental factor in which case AUCP will become UUCP. That is, there

---

**Figure 11. Main Screen for Use Case Point Calculation.**
is no adjustment done in the calculation. However, practically, every software system belongs to at least one or more of the technical factors and one or more of the environmental factors and hence the tool forces the user to select them before completing the calculation.

![Figure 12. Technical Factors in the Calculation of Use Case Points.](image)

5. Implementation & Testing

This software is implemented in Java using Java Swing for the User Interface. The details are stored in the XML file structure using XML SAX parsers library. The PDF file is written using the iText pdf library. Testing the application consisted primarily of functional testing completed during the development. Since the application involves a lot of GUI operations, we mainly focused on user event test cases. Several test cases were executed both by the developer and demonstrated to the customer.
The implementation started by first defining all the main classes. Then the concentration was on the GUI design. The GUI was designed on one form with a tabbed structure. This was to make it easy for the user without having to navigate through different forms. All the information that was required to be entered was sectioned by separate tabs. In the case of ‘Assumptions’, ‘Preconditions’, ‘Postconditions’, etc. there were options given to order, edit and delete them with ‘Move Up’, ‘Move Down’, ‘Modify’ and ‘Delete’ buttons. Once the GUI was completed and approved, the event handlers were written. We included an auto save option that will write the data to the file on each tab click. This avoids loss of data entered. The user does not have to keep saving his work.

After the completion of the event handling and writing data to the file, the next task was reading from the XML file. The user could choose the XML file. The data would be read from the XML file and the GUI screen would be filled with the corresponding data from the file. The PDF file was generated next. This was achieved using the ItextPdf libraries. All alignment of the text, headings and sub-heading was done using the format followed in UWL for the use case documentation. The use case diagrams are also placed in the documents.

The additional feature of the software was the Cost Estimations using Adjusted Use Case Points (AUCP). The weights for the actors and the use case were directly extracted from the use case narratives and displayed for the user. The Environment variables and Technical Factors were listed for the user to select according to the complexity of the project. Only after this selection is made the Adjusted Use Case Points is displayed. After the estimated hours for each use case it entered, the software will display the estimated hours for the project.

6. Conclusion
This manuscript describes the development and use of an editor for developing use case based requirements documents. The structure of the document is similar to the one recommended by the IEEE and is also similar to most commonly used use case documents. The main advantages of the tool are its simplicity, user-friendly interface and automatic calculation of cost estimation using use case points. Consequently, this tool is not as complex as IBM Requirements Composer tool [3] or the Requirements Capture tool provided by Visual Paradigm [4]. Some of the features of the tool are listed below:

- Easy-to-use GUI. Every item in the document is entered through dialog boxes and formatting of the final document is done by the tool.
- Most entries are validated while entering.
- The tool uses XML to store use case narratives and hence it is relatively easier to process the requirements using another tool.
- Use case diagrams (more than one) can be embedded into the document.

Though the tool was primarily designed as an educational tool to support requirements engineering and cost estimation, it can be used by real applications as well.

This tool was presented at the Midwest Instruction and Computing Symposium (MICS) on April 25, 2014 at Verona, Wisconsin[10].

7. Future Expansion

For future expansion, the software could have an in built tool for drawing the use case diagram. It could also have more options for formatting, selection of fonts, changing the order of the sections.

8. References


http://www.visual-paradigm.com/product/vpuml/features/sysml.jsp


9. Appendix

9.1 Screenshots

Figure 13. Preamble tab
Figure 14. Title Page tab

Figure 15. Introduction tab
Figure 16. Assumptions tab

Figure 17. Use case diagram tab
Figure 18. Use case - Use case definition tab

Figure 19. Use case - Precondition tab
Figure 20. Use case – Postcondition tab

Figure 21. Use case – Successful Scenario tab
Figure 22. Use case - Includes/Excludes tab

Figure 23. Use case – Exceptions tab
Figure 24. Use case – Additional Remarks

Figure 25. References tab
Figure 26. Actor classification tab

Figure 27. Use case classification tab
Figure 28. Use case parameters tab

Figure 29. Environmental Factors tab
Figure 30. Technical Factors tab

Select Technical Factors:
- Distributed System
- Response or throughput objectives
- End user efficiency (Online)
- Complex internal processing
- Code must be reusable
- Easy to install
- Easy to use
- Portable
- Easy to change
- Concurrent
- Includes special security features
- Provides direct access to third party software
- Special user training facility is required

Figure 31. Calculate efforts tab

Efforts in Man Hours

Un-adjusted UUCP = 8.5
Environmental Factors = 1.33
Technical Factors = 0.82

Adjusted UCP = 8.44

Select Number of Hours estimated for each UUCP

Efforts in Man Hours: 42.109

Calculate Efforts