A Requirements Engineering Tool Based on Use Cases

Jenny Gijo and Kasi Periyasamy
Computer Science Department
University of Wisconsin-La Crosse
La Crosse, WI 54601
{gijo.jenn,kperiyasamy}@uwlax.edu

Abstract

The outcome of a requirements engineering process is a structured document that describes in detail all requirements of a software system. IEEE recommends to use a well-defined format for writing requirements so that software designers and developers can easily understand the requirements. When 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 IEEE recommended format is independent of the design methodology, deriving an OO design from functional requirements itself is a challenge. The shift in programming paradigm made some researchers to develop use case based requirements. The focus of this paper is on 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.

1 Requirements Engineering Tools

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.

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.

This paper describes the development of a tool that helps writing use case based requirements document. The structure of the document is chosen to be somewhat similar to the one recommended by IEEE for requirements specifications. The aim of this project is to develop a simple and easy-to-use requirements tool that can be used in a classroom setting. 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.

1.1 Cost Estimation

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 [6] proposed a use case based approach for cost estimation. Since then, a lot of researchers have extended Karner's approach including one of the authors of this paper. The tool implements the cost estimation algorithm that is described in [6]. Details of this algorithm are given in a later section.

2 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 1 shows a use case diagram for a simple ATM system involving only four transactions.

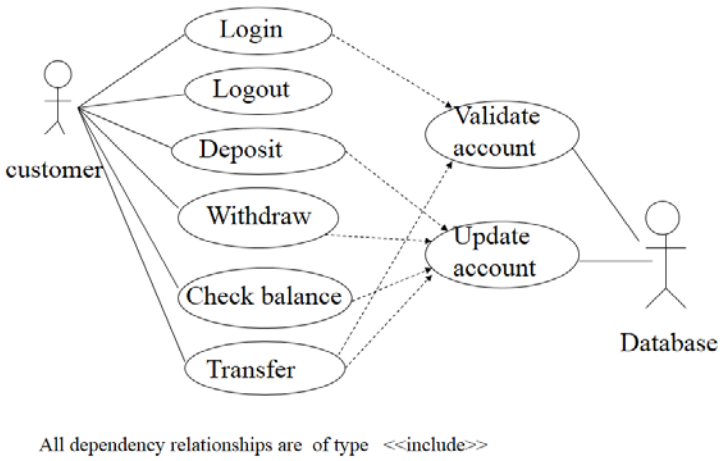

Figure 1. Use Case Diagram for a simple ATM System

As seen in Figure 1, 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 2. There is no standard or recommended format for a use case narrative; the one shown in Figure 2 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.

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

2.1 The Structure of 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 3 has been proposed for the use case narratives document. This structure is somewhat similar to the one recommended by IEEE and so users of IEEE template will have a smooth transition to the use case based document.

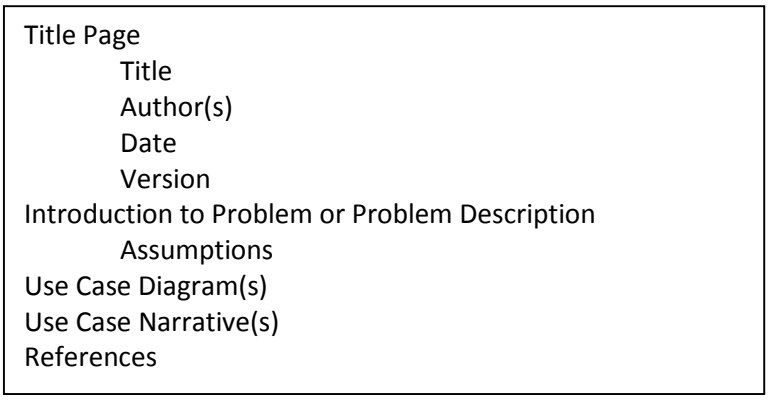

Figure 3. Document Template for Use Case Narratives

3 Use Case Based Requirements Editor

The design of the use case based requirements editor mimicked the requirements editor based on 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 4 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 3. 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’). Thus, 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

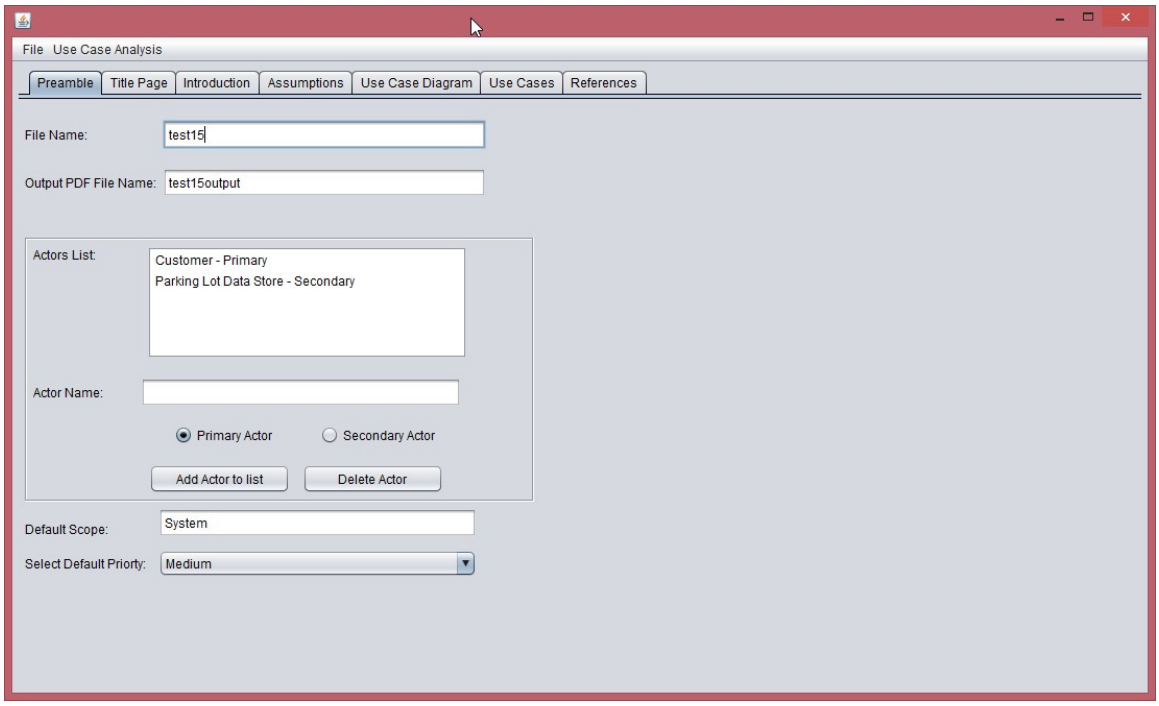

Figure 4. Main Screen of the Requirements Editor

frequently. So it is preferable to keep assumptions under a separate tab. Figure 5 shows an expanded view of the ‘Assumptions’ tab.

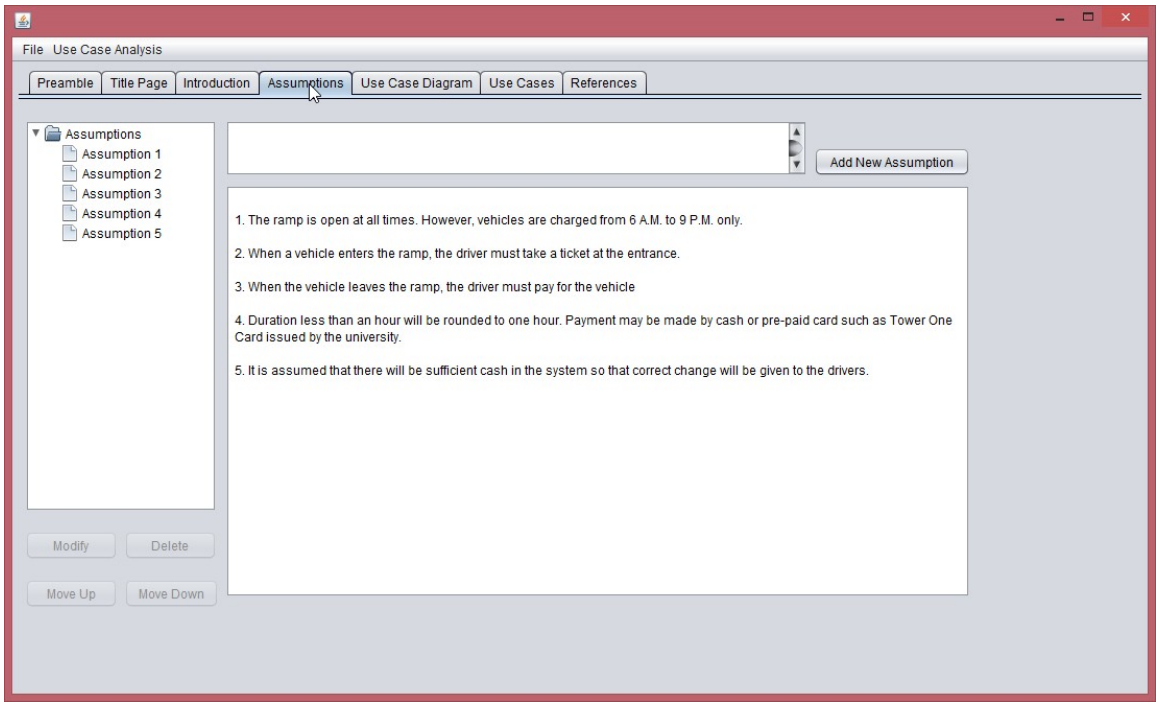

Figure 5. Adding and Editing Assumptions

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 6.

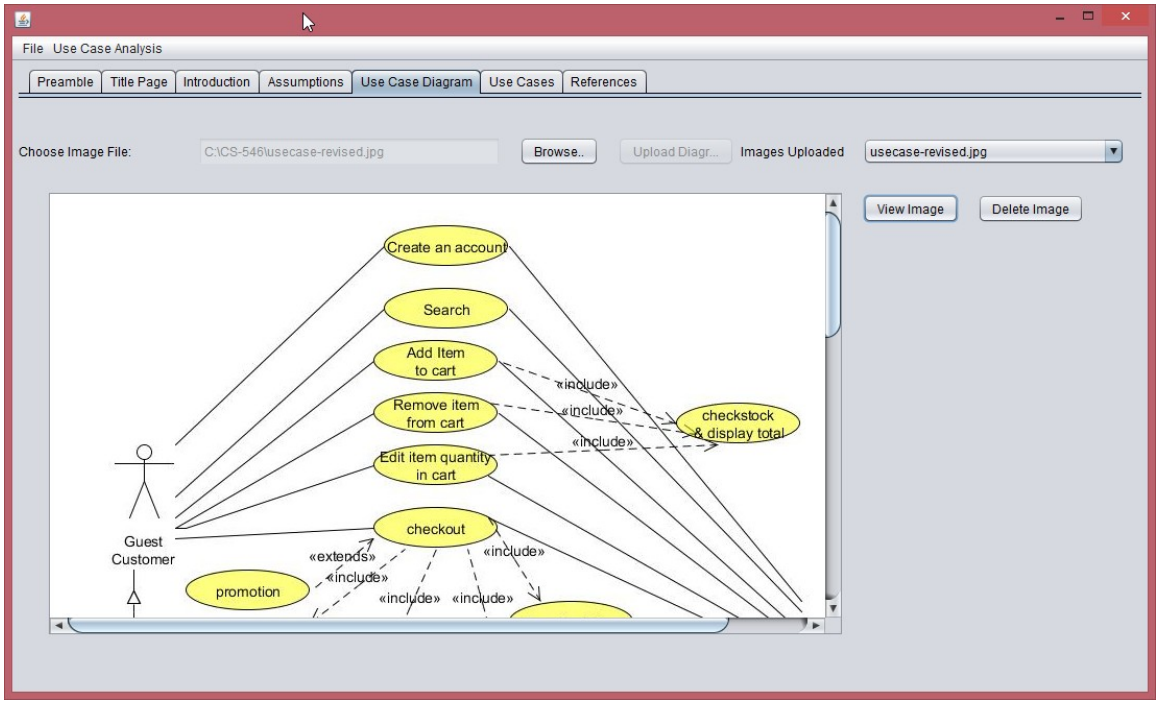

Figure 6. Uploading Use Case Diagrams

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 7 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.

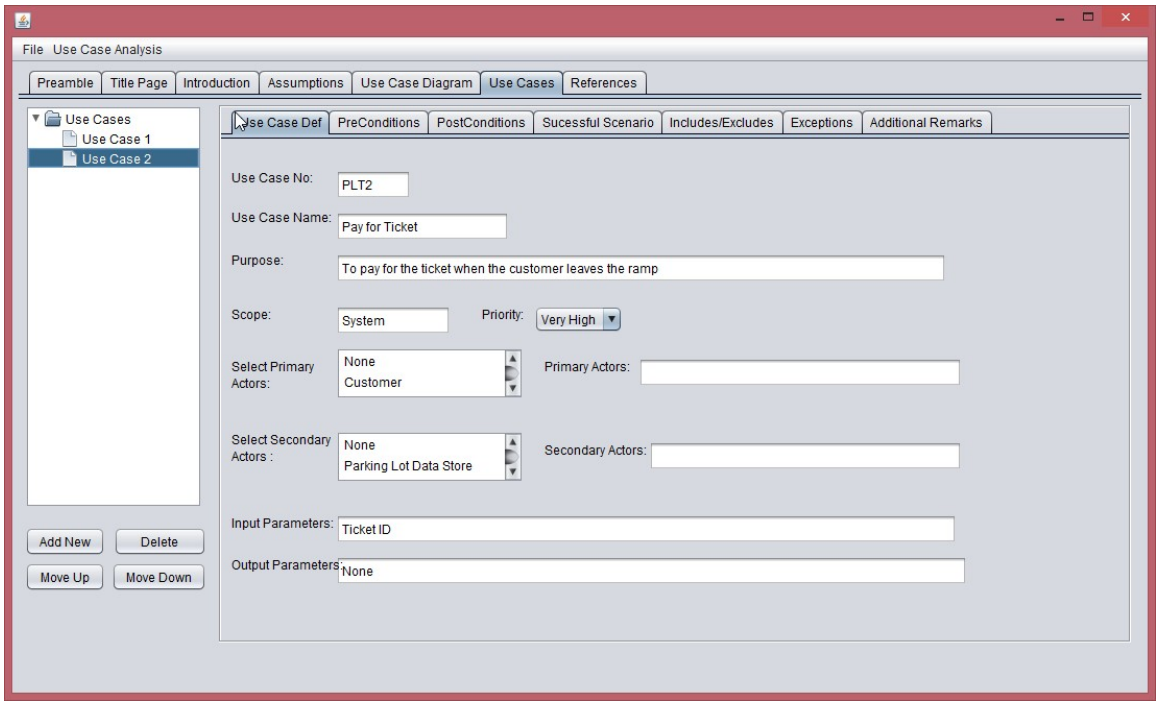

Figure 7. Adding and editing use cases.

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.

The tab ‘References’ is self-explanatory and is not further described in this paper.

4 Cost Estimation using Use Case Points

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 which 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 which 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 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 * Ti), for all 1 <= i <= 13. Similarly, the contribution of environmental factors towards the cost estimation is calculated by the formula EF = 1.4 + (-0.03 * Ei), for all 1 <= I <= 8. The final step is the calculation of the Adjusted Use Case Points (AUCP) which is computed as AUCP = UUCP + TCF + 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 which 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 in [6].

4.1 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 8 shows the main screen of use case point calculation. This screen is displayed

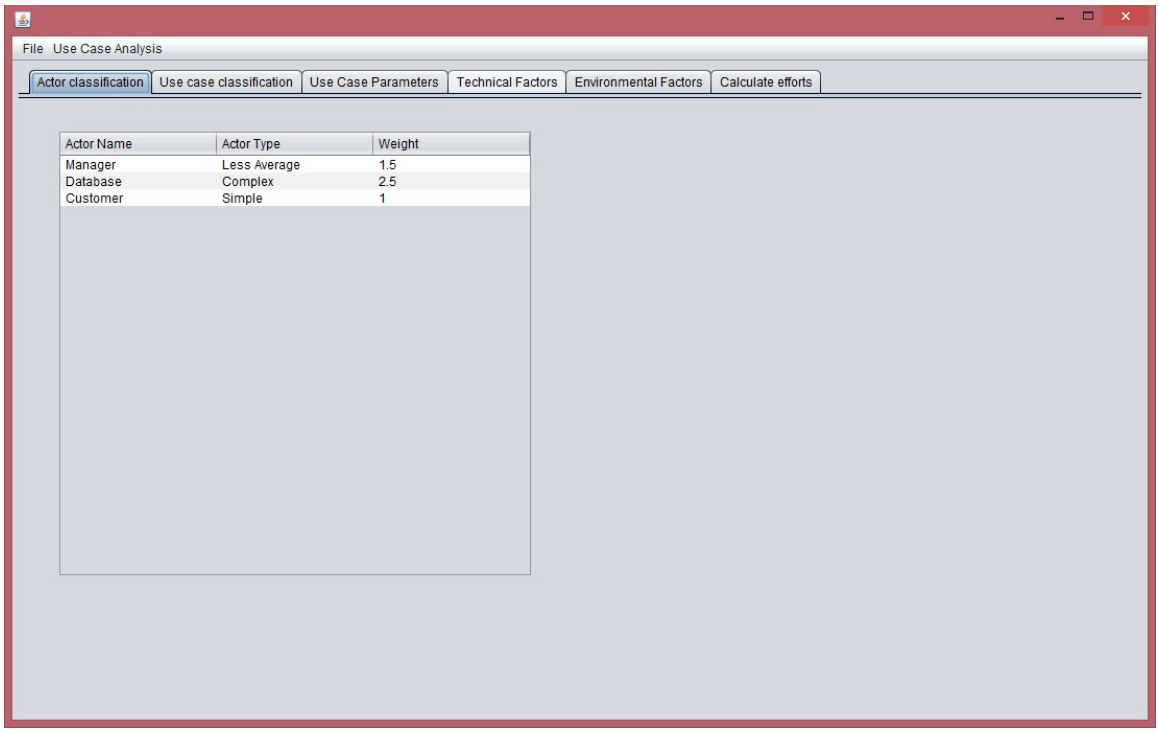

Figure 8. Main Screen for Use Case Point Calculation.

when a user selects the use case analysis option from the menu bar on the top. As seen in Figure 8, 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 assignment 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 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.

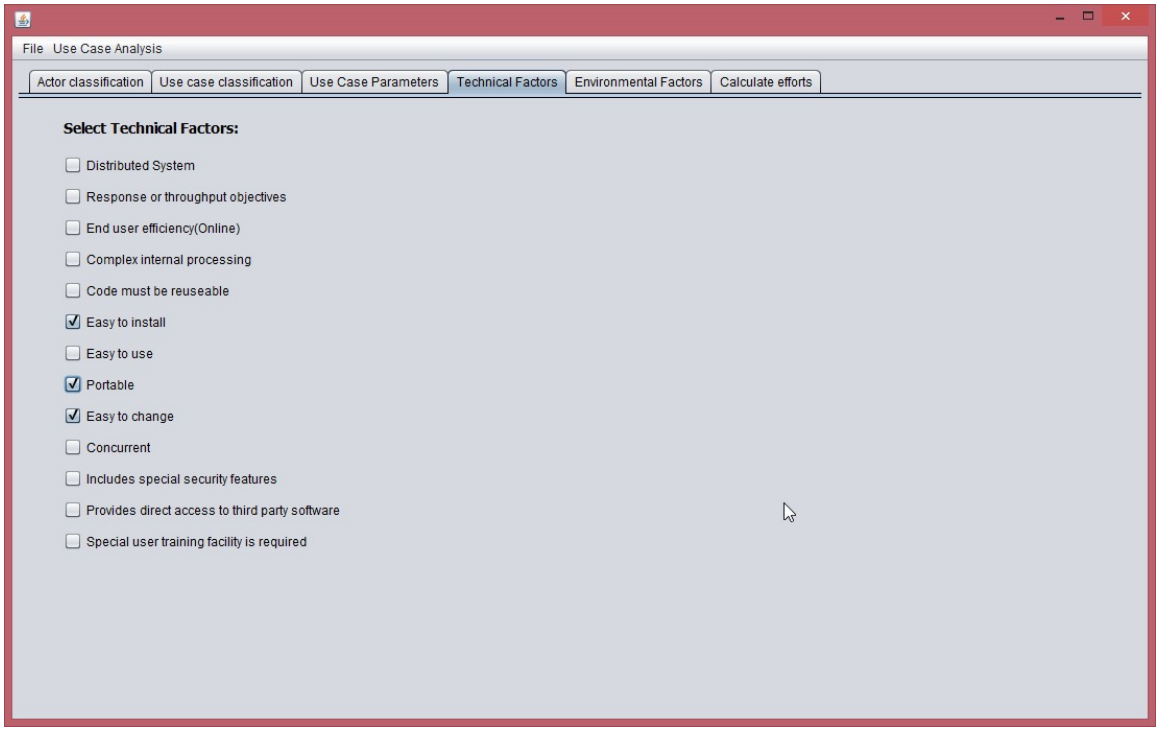

Figure 9. Technical Factors in the Calculation of Use Case Points.

5 Conclusion

This paper describes the development and use of an editor for developing use case based requirements document. The structure of the document is similar to the one recommended by 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. The tool has been developed primarily for educational purposes and is meant for students writing use case based requirements. However, it can also be used by any real application.

References

[1] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998, June 1998.
[2] Hans-Erik Eriksson *et al*, *UML 2 Toolkit*, Wiley Publishing Inc., 2004.
[3] Rational Requirements Composer developed by IBM, 2012. http://www-03.ibm.com/software/products/en/rrc
[4] Requirements Capturing Tool developed by Visual Paradigm, 2013. http://www.visual-paradigm.com/product/vpuml/features/sysml.jsp

[5] G. Karner, Metrics for Objectory, Diploma Thesis, University of Linkoping, Sweden, No. LiTHIDA-Ex-9344:21, December 1993.
[6] K. Periyasamy and A. Ghode, "Cost Estimation Using Extended Use Case Point (e- UCP) Model", International Conference on Computational Intelligence and Software Engineering (CiSE 2009), Wuhan, China, December 2009.
[7] B. Garbers and K. Periyasamy, "A Light-weight Tool for the Development and Evaluation of Requirements Documents", in the Proceedings of the Annual Conference of American Society of Engineering Education (ASEE), Chicago, IL, June 2006. Also presented in the Global Colloquium of ASEE, Rio de Janeiro, Brazil, Oct 2006.