2006-227: A LIGHT-WEIGHT TOOL FOR TEACHING THE DEVELOPMENT AND EVALUATION OF REQUIREMENTS DOCUMENTS

Ben Garbers, University of Wisconsin-La Crosse

Ben Garbers has been working with IBM, Rochester, MN for 6 years where he had experience with software requirements gathering, design, development and testing. His technological expertise includes Java applications, dynamic web applications and artificial intelligent applications. Ben is a graduate student in the Master of Software Engineering program at the University of Wisconsin La Crosse. Currently he is a first line manager of an internal build tools department at IBM.

Kasi Periyasamy, University of Wisconsin-La Crosse

Kasi Periyasamy is a Professor in the Computer Science Department and the Program Director of the Master of Software Engineering (MSE) program at University of Wisconsin-La Crosse. His research interests are in Software Specifications, Requirements Engineering, Software Testing and Verification. He has published a lot of papers in Software Engineering area, and is the co-author of the book "Specification of Software Systems" published by Springer-Verlag, 1998. He is a member of the Association of the Computing Machinery (ACM).

A Light Weight Tool for Teaching the Development and Evaluation of Requirements Documents

Abstract

Writing correct and consistent software requirements specification (SRS) is one of the most important goals of a requirements engineering process. The SRS serves as the basis for subsequent design, testing and maintenance of the software product. The more errors and inconsistencies contained in an SRS, the more time and efforts are required to correct them at a later stage in the development process. Most SRS documents are manually typed using a word processor and hence the writer is responsible for ensuring the correctness and consistency of the document. Without adequate tool support, the manual construction and analysis of SRS documents is a tedious process and is error-prone. This paper describes an interactive tool developed at the University of Wisconsin-La Crosse that assists students preparing an SRS document based on the IEEE standard 8301998[1]. The tool provides an easy-to-use interface and the ability to create, edit, load and save SRS documents. In addition, it evaluates the requirements document based on criteria published by the Software Metrics program at the Software Assurance Technology Center, NASA[2]. A function-point metrics analyzer is also built into the tool so that the efforts required to complete the project specified in the document can be evaluated.

Introduction

A project-oriented course in Software Engineering generally requires the students to analyze the requirements for the problem and then write the software requirements specification (SRS) document. Since the SRS serves as the basis for design, testing and maintenance of the software product, the students are expected to follow some standard such as IEEE 830-1998[1] while developing the SRS. A sample functional requirement in IEEE standard format is shown in Figure 1.

Index:	ATM.2
Name:	Deposit
Purpose:	To deposit an amount into an account
Priority:	High
Input parameters:	***account number***, ***amount***
Output parameter:	None
Action:	Ensure that ***account number*** exists.
	Ensure that ***amount*** is greater than zero.
	Retrieve the account with ***account number***.
	Update the balance in the account by adding ***amount*** to it.
Exceptions:	***account number*** does not exist.
	amount is less than or equal to zero.
Remarks:	None

Figure 1: An IEEE 830 compliant functional requirement in an ATM system

Often, students use simple word processors to write the document and so are enmeshed with clerical details such as correctly specifying the section numbers, specifying unique index numbers for functional requirements and so on, and do not pay much attention to the contents of the document. With the result, the final document may be well-typed but will not have sufficient contents. In particular, the review of such documents shows that the students should spend more time in writing the contents of the document first than structuring the document according to the standard. Without adequate tool support, the manual construction and analysis of SRS documents is a tedious process and is error-prone. It is therefore decided to develop a tool to assist the students in developing the SRS. The students, when using this tool, will be prompted with various sections and paragraphs within the sections and the tool prints the final document in IEEE standard format. Thus the students are relieved in focusing on the clerical details of the document.

The authors developed a tool called *Napkins* that would assist the students in developing IEEE compliant requirements document. Using this tool, one would be able (i) to develop a SRS based on a simplified version of IEEE standard format, (ii) to evaluate the SRS based on criteria published by the Software Metrics program at the Software Assurance Technology Center, NASA[2], and (iii) to evaluate the time to complete the project based on the requirements entered using the tool. This last evaluation uses function point metrics. The Napkins tool can therefore be used by the students in a Requirements Engineering course as well as by those in a Software Metrics course.

Napkins - Design Details

Figure 2 shows the architecture of the Napkins tool. The tool consists of three major components – the requirements editor, metrics analyzer and the requirements document evaluator.

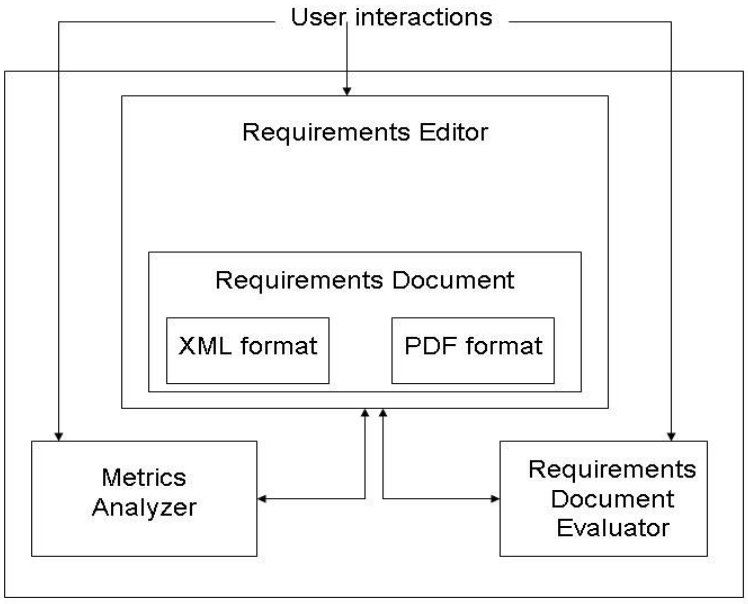


Figure 2: Architecture of the Napkins Tool

The editor provides several windows for the user to input various sections of the requirements document. Internally, the tool stores the document in XML format so that it can be interchanged with other tools. In addition, the tool also provides the option to generate a PDF file of the requirements document for printing and publishing. More details of the requirements editor are given in a subsequent section.

The purpose of the metrics analyzer is to estimate the project deadline using function points metrics[3]. Information needed to compute the metrics are to be manually extracted from the requirements document and then use the metrics analyzer just like a calculator. In addition, the user will be able to interact with the metrics analyzer directly to provide additional information such as adjustment factors to get a more accurate estimation of the deadline.

The requirements document evaluator scans through the requirement document and displays some primitive metrics for evaluating the document. The user then evaluates the quality of the document based on the guidelines provided by NASA Software Metrics Program[2]. This evaluator is also discussed in detail in a later section.

Requirements Editor

Figure 3 shows the starting screen of the requirements editor. All the three components of the Napkins tool can be accessed through the menu shown on top of the screen. The left pane provides the various sections of a IEEE compliant requirements document. A user of the tool will be able to navigate and input the contents of various sections using the left pane. As an example, Figure 3 shows how a user can input a new functional requirement.

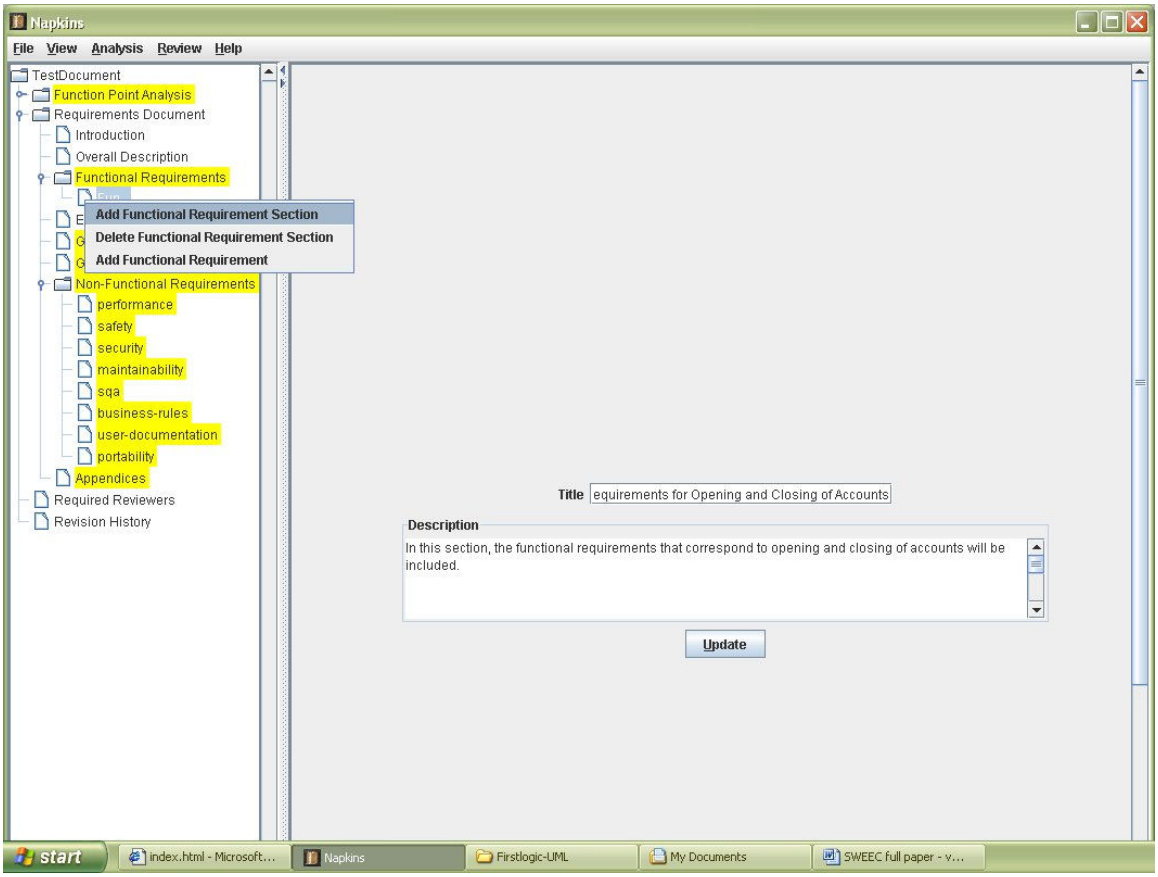


Figure 3: The Requirements Editor

The requirements editor enables a user to create a new requirements document, to save it in XML format, to save it in PDF format, to load a previously created requirements document and to edit a requirements document. All these functionalities can be invoked through the 'File' menu option. The editor also supports uploading of images so that graphical user interface requirements can be included in the document. The highlighted sections on the left pane include additional functionalities through the 'right click' of mouse button. For example, Figure 3 shows how a user can add a new functional requirement by right clicking on the Functional Requirements section first and then clicking on the 'Add Functional Requirement'.

Currently, the editor supports a simplified version of IEEE standards format for requirements specification. This includes functional requirements, graphical user interface requirements and non-functional requirements. Even though the screen shows only a handful of non-functional requirements, this list can be expanded by the user. The requirements can also be divided into several subsections, thus allowing modular specification of the requirements.

Metrics Analyzer

The metrics analyzer computes the estimated time to complete the project whose requirements are described by the current requirements document. The computation is based on function point metrics. Function point metrics is used to evaluate the complexity of a software product; it uses

function points as the complexity factor instead of the traditional Lines Of Code (LOC). There are two steps in the calculation of the number of function points. The first step is to compute the unadjusted function points for each requirement in the document. We have used the method from Dreger[3] for this calculation. Accordingly, each functional requirement is classified as simple, average or complex. Each classification has been given a specific number (for example, simple may be 3, average may be 5 and complex may be 7). These numbers may vary depending on further classification of the functional requirement. Accordingly, for each functional requirement we first identify its business category (Input, Output, Inquiry, File or Interface). We then analyze the number of data references, file references and logical record format relationships for the functional requirement. Depending on this analysis, its unadjusted function point value changes. For example, the following table shows the function point value for the functional requirement belonging to business category 'Output'.

	Data items referenced		
Files referenced	**1 to 5**	**6 to 19**	**20 or more**
0 or 1	Simple (4)	Simple (4)	Average (5)
2 or 3	Simple (4)	Average (5)	Complex (7)
4 or more	Average (5)	Complex (7)	Complex (7)

More details regarding these unadjusted function point values can be found in Dreger's[3] book. Once the unadjusted function points are calculated, the second step is to fine-tune them using 14 adjustment factors. These are tabulated below.

Data communications	Online update
Distributed data or processing	Complex processing
Performance objectives	Reusability
Heavily used configuration	Conversion and installation ease
Transaction rate	Operational ease
Online data entry	Multiple site use
End-user efficiency	Facilitate change

The adjustment factors can be entered by clicking on the 'Function point analysis' icon on the left pane and entering the values on the window on the right side. The results of function point analysis can be viewed through the 'View' menu option. They are also displayed when the user presses the 'Calculate' button at the bottom of the screen on Function Point Analysis. However, this option only shows the final value while the 'View' option provides all details of the calculations.

In order to assist the users to provide the right input for unadjusted function point, a 'Help' button is provided on the Function Point Analysis screen. This will open a separate window explaining the calculation of unadjusted function points.

Apart from the function point analysis, the tool also provides another option to estimate the completion time of the project using DeMarco's model[4]. While entering the details of each functional requirement, the user is asked to provide a weight for the functional requirement to determine its size. The following table gives the weights and their associated numeric values.

Function type	Weight
Compose or decompose data	0.8
Update information	0.5
Analyze data and take action	1.0
Evaluate input data	0.8
Check for internal consistency	1.0
Text manipulation	1.0
Synchronize interactions with users	1.5
Generate output	1.0
Perform simple calculations	0.7
Perform complex calculations	2.0

These weights for the functional requirements are used to calculate the sizing parameter as follows:

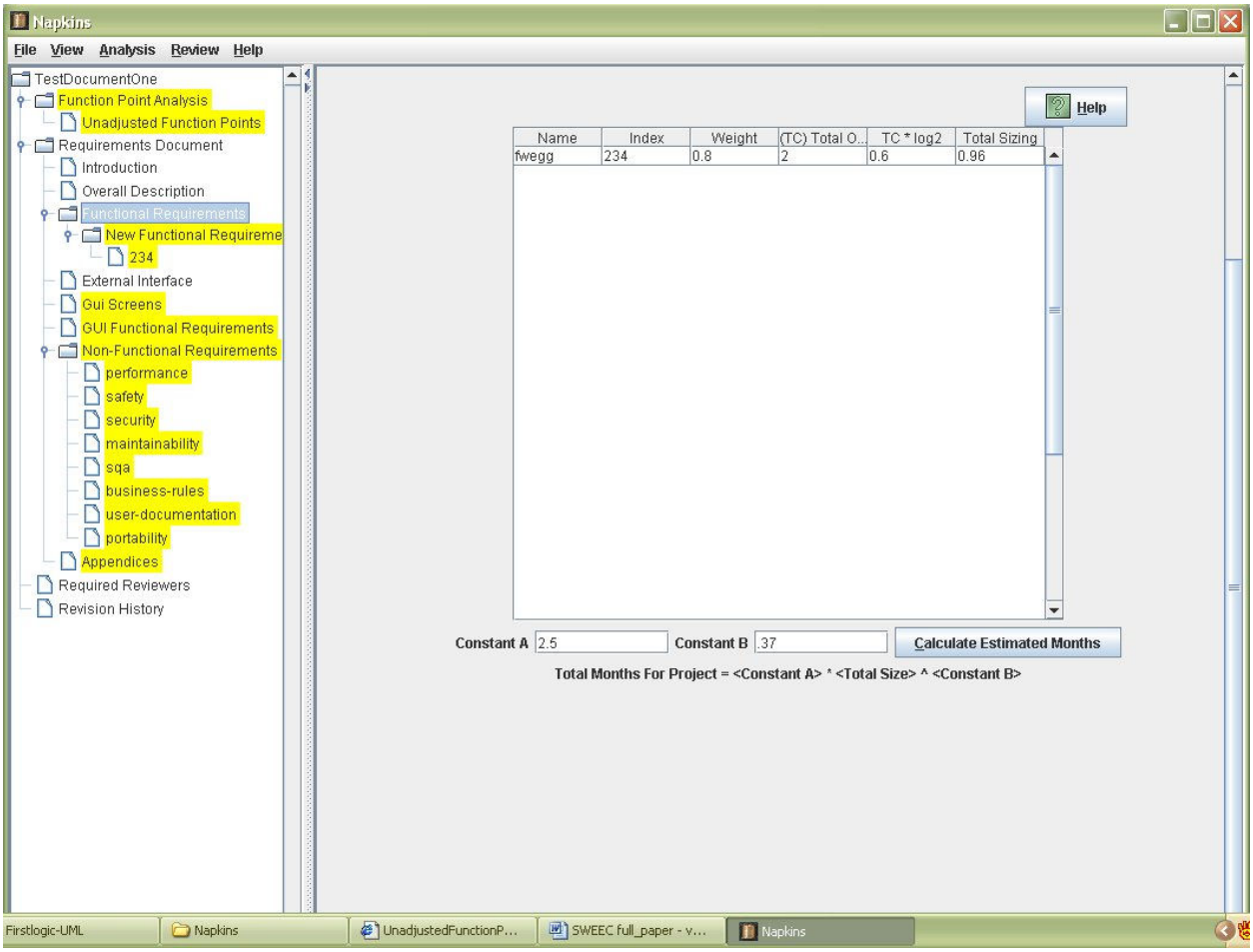
Total Sizing = Weight * <Total inputs and outputs> * (log2 * <Total inputs and outputs>)

Finally, the estimated time to complete the project is calculated as

Estimated time = Constant A * Total Sizing ^ Constant B

where 'Constant A' and 'Constant B' are to be defined by the designer depending on the project.

The results of this calculation are shown on the first screen on Function Point Analysis (see the screen shot below).



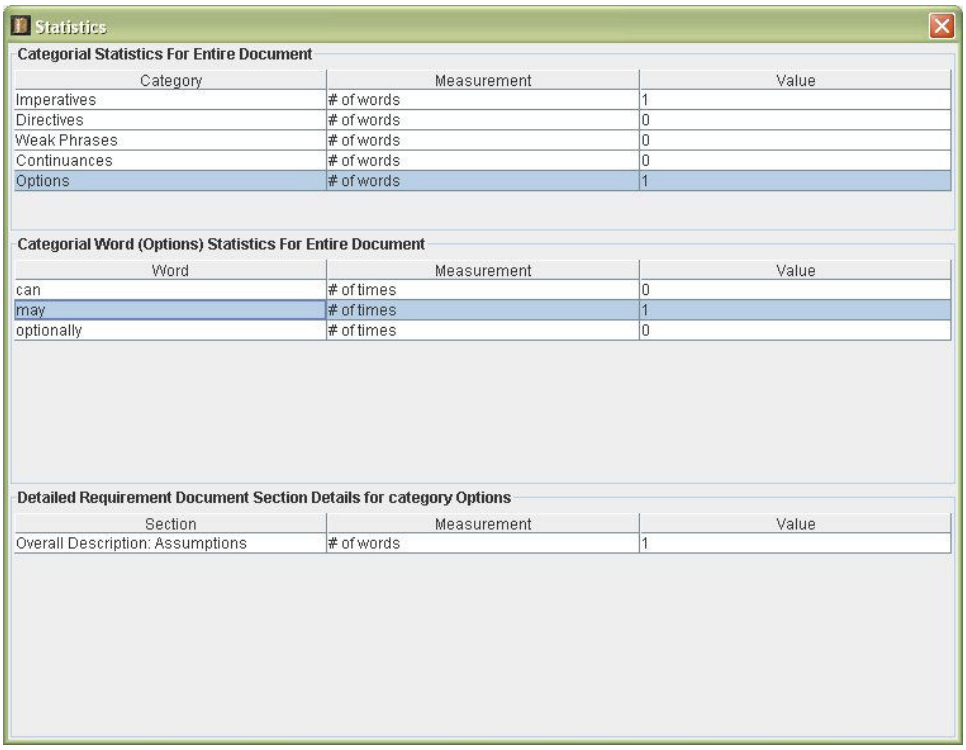
Requirements Document Evaluator

There are very few publications available on evaluating the quality of requirements documents. The authors found that the evaluation criteria published by the Software Metrics program at NASA seem to be appropriate for teaching purposes. In this work, the requirements document is first scanned for the number of words appearing in five different categories. The categories and some possible words in each category are tabulated below:

Category	Words in this category
Imperatives	Shall, should, must, will, are applicable to
Directives	Figure, Table, For Example, Note
Weak phrases	Adequate, as appropriate, as applicable
Continuances	Below, as follows, listed, following
Options	Can, may, might, optionally

Once the number of words in each category is extracted, the evaluator can determine whether it is appropriate to have, say, N number of words in category K. Since this decision depends on the

application domain, it is not possible to automate this process. Currently, the Napkins tool includes a mechanism to extract the words in each category and display them indicating their count and the sections in which they appear. Further, a user can also add or delete specific words in each category. The user can invoke the requirements document evaluation component through the 'Analysis' menu option. A sample screen shot of this component is shown below:



Major advantages

The major advantages of the Napkins tool are summarized below:

- It enables a user to develop an IEEE compliant requirements document focusing on the contents of the document and not worrying about the clerical details.
- It allows the user to estimate the time taken to implement the product using function points metrics. In addition, the user will also be able to configure the parameters for the function point analysis. The tool also provides another option to estimate the completion time using DeMarco's model. With the two options together, not only the students learn about metrics but also will be able to compare the two techniques.
- The tool provides a count of primitive metrics to evaluate the quality of the requirements document.
- The requirements document can be saved in PDF format for printing, viewing and publishing. It internally saves the document in XML format. This allows other tools to process the document.
- Using the 'Review' menu option, a user will be able to add a review of the requirements document (generally done by a reviewer other than the requirements writer). These reviews are stored within the tool for later analysis.

Limitations

The Napkins tool is currently used by the students in Software Engineering courses at the undergraduate and at the graduate level. The requirements editor is more or less complete even though it has a simplified version of IEEE standard format. This is because the tool is designed primarily for the students in Software Engineering courses. If the tool is planned to be used for actual software development, it needs to be extended to include other sections based on IEEE standards. The authors are planning to develop the extended version in the near future.

The metrics analyzer provides the mechanism to compute the function points more accurately by implementing the adjustment factor. However, the user is expected to manually input the five parameters to compute the unadjusted function points and the 14 adjustment factors to compute the final count of function points. It is possible to extract the five parameters for unadjusted function points directly from the requirements document provided that the user identifies them while writing the document. For example, the number of internal files can be extracted if the user indicates them while typing in the input and output parameters for functional requirements. The authors are planning to add such mechanism in the next version of this tool.

The requirements document evaluator simply counts the number of words in each category. The assessment of the quality of the requirements document based on the counts of words in each category is not implemented in the current version. This is because NASA has not published the details of such assessment. Moreover, such assessment depends on the application domain and hence is not possible to be automated. The authors are investigating the possibility of providing guidelines for such assessment but it requires extensive research in this area.

Evaluation

The tool was used by a small group of students and some people at IBM, Rochester, MN. The overall comments about the tool and its usage were positive and encouraging. The small group of students who evaluated this tool was part of the Software Engineering course; these students used the requirements editor more than the metrics analyzer. From the students' perspective, the user interface needs some improvements and some features such as cutting and pasting text from one portion to another portion of the requirements document must be improved. Another concern was the lack of on-the-fly help facility. Overall, the students were satisfied with the functionalities of the tool. Some comments from the students are included below:

- It is easy to understand this tool.
- No need of remembering the requirements document format.
- The GUI elements are not aligned in some windows. Layout needs to be changed.
- Help functionality gives an overview of the tool. It would be great if help can be in detail and interactive instead of reading long pages of text.
- Converting to pdf file make it platform independent.

In contrast to the students' population, the evaluators from IBM mostly commented on the metrics analyzer. The use of metrics analyzer and requirements document evaluator were appreciated. The major concern is the lack of help facility. Currently, the tool provides online help which is an overview of the entire tool. Some of the comments from IBM evaluators are listed below:

- I like the fact that it will keep track of project estimation using different theories such as DeMarcos Model.
- Help text needs to be written on how to successfully use the review comments process.
- Was not apparent that you have to right click to add functional requirements.
- The estimation of time is apparent on how it is calculated when adding functional requirements.

All evaluators suggested that the tool must provide tool tips and short on-the-fly help facilities.

The authors have planned to improve the tool based on the first set of evaluations. In addition, the authors are going to make the tool available for public in order to get more feedback. Currently, the authors are demonstrating the tool to other institutions in the same region. Three of these institutions have agreed to use the tool in their software engineering courses and send feedback to the authors next year.

Conclusion

This paper describes the Napkins developed by the authors for developing a software requirements document. The document's structure complies with IEEE standard 830-1998. The tool also provides a metrics analyzer for computing function point metrics and the time taken to complete the project based on the requirements. In addition, the tool also includes another component to extract primitive metrics for assessing the quality of the requirements document. This assessment is based on the guidelines provided by the Software Metrics Program at NASA. The limitations and continuing work on the tool are briefly addressed.

Another requirements editor called *Requirements Compiler* (RC)[5] is also in development at the authors' institution. Even though RC provides somewhat similar mechanisms to develop an IEEE compliant requirements specification document, the method and the interface are quite different from those of Napkins. Further, the purpose of RC is to assist in automatic derivation of an object-oriented design from the requirements document. RC also provides some additional validations such as missing sections so that the user will be able to write a truly IEEE compliant requirements specification. In order to facilitate document exchange between the two editors, both of them use XML format to store the contents. The designers of both the editors are working cooperatively to support this document exchange.

The authors are also developing a management activities tracking tool which helps project managers assign management tasks to team members and keep tracking the individual assignments. The Napkins tool is linked to the management activities tracking tool and hence the requirements specification, requirements review and deadline estimation can all be used within the management activities tracking tool.

Bibliography

1. IEEE Recommended Practices for Software Requirements Specification, IEEE Standard 830-1998, 1998.
2. http://satc.gsfc.nasa.gov/metrics/index.html

3. Brian Dreger, “Function Point Analysis”, Prentice Hall Advanced Reference Series, 1989.
4. Stephen Kan, “Metrics and Models in Software Quality Engineering”, Addison-Wesley, 2003.
5. Tom Harron, “Requirements Compiler”, final manuscript in preparation for the Master of Software Engineering Degree, Department of Computer Science, University of Wisconsin-La Crosse, 2006.