REQUIREMENT-MANAGEMENT TOOL FOR SMALL-TO MEDIUM-SCALE SOFTWARE PROJECTS

By Jay P. Alegata

ABSTRACT

Software requirements consist of the product and behavioral properties of a software system. They are essential factors in building the right software product/s for clients and customers. Communication is very important during the development of a software to ensure that the needs and expectations of the client are satisfied. An agreement between the developer and client, regarding project deliverables are produced. The Requirement-Management Tool (RMT) was developed to manage the production of various requirements artifacts like Statement-of-User-Requirements (SUR), Software Requirements Specifications (SRS), and Acceptance Criteria for sign-off. Most of the problems in software development includes: keeping track of user requirements; but with the use of this software, a checklist is generated by which the system analyst can determine the status of user requirements. The RMT software utilizes the IEEE Software Engineering standards for documenting software requirements to formalize requirements artifacts. As one of the case tools for software development, it can significantly increase the consistency of requirements documentation and tracking of software projects. The development of RMT follows the object-oriented analysis and design methodology wherein individual components are derived from the requirements and use cases.

INTRODUCTION

Software requirements are essential factors in building software products for customers. In fact, it is defined by the Software Engineering Body of Knowledge (SWEBOK) as one area that software engineers could engage in as part of their professional practice of software development and software engineering. It covers both the functional and non-functional properties of the software system. During the course of development, developers conduct elicitation, analysis, specification, validation and management of these requirements. To ensure that both developer and client agree on the requirements, project deliverables are produced during this phase for approval. These deliverables consist of Statement-of-User Requirements and Acceptance Criteria (SUR), Software Requirements Specifications (SRS), and Acceptance Criteria for sign-off.

The current practice of software development shows that requirements are rarely documented by the developers because production of such documents is time-consuming. As a result, requirements problems plague the software industry. Most of these problems include: lack of consistency of software requirements artifacts resulting to poor traceability of user requirements to the various phases of software development, lack of control of requirements changes making it difficult for client to determine whether or not requirements are satisfied, and developers not being able to keep track of approved change requests due to invisibility of requirements process.

In this regard, requirements management tools such as TIGER PRO v1.13, TRUEreQ, CMS & RTS, MKS Requirements 2006 among others were developed to help developers in documenting user requirements for the software product. Commercially available tools flood the software market but small-to medium-sized development teams do not use them due to cost and usability constraints. Large companies justify its benefits like being able to track project progress and monitoring requirements changes. However, these tools cannot help elicit requirements from clients and can't replace the defined process of managing project requirements (Weigers, 1999). Most of the commercial software available for requirements can be linked with MS Word applications or with other requirements management tools.

Based on the analysis of Weigers, requirements tools made by Rational Corporation such as Rational Rose can be directly connected with the RequisitePro in defining requirements. Since these tools are generic, they can support and be tailored into the development process of a software organization with the aid of the vendors themselves (Weigers, 1999).

In software projects, different types of requirements for the software product are specified by the customer. These can include business requirements, functional requirements, hardware requirements, nonfunctional requirements, and quality requirements. Nowadays, there are various requirements management tools that can be used to facilitate or manage the documentation (http://www.paper-review.com/tools/rms/read.php). Some software companies have customized software tools for managing requirements depending on the complexity of their software projects to minimize delays of documenting unnecessary requirements.

The requirements management tool conceptual framework (Figure 1) is based on the integration of the features of commercially available requirements management tools, the current practices of software development course of software engineering students and the IEEE Software Engineering Standards for Documentation.

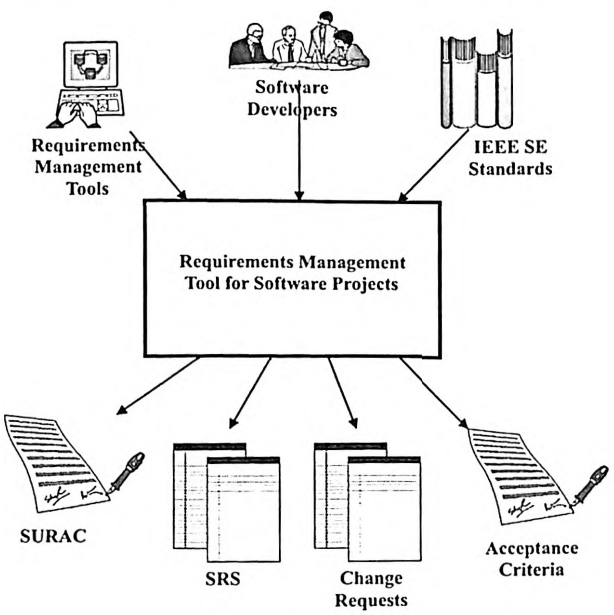

Figure 1. Conceptual Framework

These tools were the basis for defining major functionalities of the requirements management tool which focused on the major deliverables to make it easy to use, and any unnecessary features were excluded. Experiences in software development of Software Engineering students were made as basis for identifying the artifacts that were produced by the tool. In this way, navigating the requirements management tool was easy due to prioritized functionalities. The IEEE Standards for software requirements specification (IEEE Std. 830 1998) were integrated into the tool to establish compliance of requirements artifacts with the software engineering standards.

The requirements engineering process included elicitation, analysis, specification, validation and management of the user requirements in the development of a software product. However, the requirement management tool focused only on handling the major deliverables for the requirements including the generation of SUR, SRS and Acceptance criteria for sign-off. The SUR represented the list of user requirements to be developed for the client. The SRS focused on the detailed specification of the requirements but did not cater to modeling use case diagrams and UML representation of requirements. The acceptance criteria during deployment would be generated by the tool for client sign-off.

The requirements management tool is not a web-application tool but a stand-alone software application that could be used by software developers. It manages documentation of user requirements based on requirements types, use cases and generation of requirements artifacts. Since the bulk of requirements are unpredictable, individual requirements based on requirements types can be printed independently and integrated into the SUR. The acceptance criteria is subject for approval will be generated but the actual sign-off can be done manually. In addition to this, the requirements management tool defines the requirements engineering to be followed. It is the prerogative of the software developers to define their own process, but only the major documentation for requirements being catered. This tool is applicable to small-to medium-scale software projects where requirements artifacts are rarely produced.

Objectives of the study

The general objective of the study is to develop a requirements management tool for tracking, controlling and monitoring requirements artifacts of software development. Specifically, it aims to:

1. assist the project teams in documenting requirements and change requests for the software product through generation of major requirements documents such as SUR, SRS, and Acceptance Criteria;

2. manage customer's change requests for a software product and keep track of approved change requests through generation of reports of changes;

3. utilize the IEEE Software Engineering standards for documenting software requirements and its recommended practices; and

4. integrate best practices of software engineering in tracking and controlling artifacts of software requirements.

METHODOLOGY

Early in the development, the researcher had examined various literatures and fora that discussed their respective requirements process. Since software requirements could be elicited from senior software engineering students' projects, and literatures for requirements processes were available, the researcher utilized the modified waterfall model (Figure 2) for its development. Waterfall model was essential to the development of the project because the requirements were defined and changes to requirements were minimal. Although the client was not considered in the development, detailed requirements were made available to users to ensure product conformity.

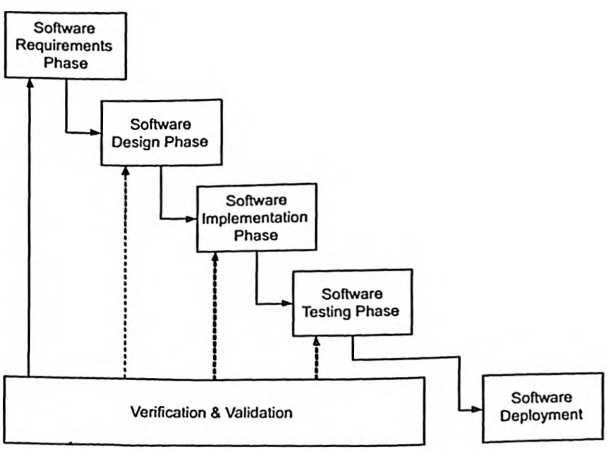

Figure 2. Modified water fall

During the development of RMT, the following five (5) major software development activities were performed: software requirements, software design, software implementation, software testing and software deployment. Additional activities like inspections and peer reviews were performed and focused mostly on the information flow between subsystem interfaces. Verification and validation activities were conducted by the researcher among colleagues and senior software engineering students to check its feasibility in their current software development project. These activities served as a defect prevention process of the Waterfall Model to minimize defects specifically in the prioritized features of the RMT. A summary report of each activity was documented to identify the major problems and inconsistencies of the software development activities.

Software Requirements Phase. Initially, requirements elicitation was performed among software engineering students with software projects. Informal interviews and observations of the process when meeting with clients in the development were conducted. As for requirements artifacts, the project workbooks of software engineering graduates were examined. Information about how software companies deal with requirements processes gathered from software engineering forum were found useful in defining the major features of RMT. To formalize defined requirements for RMT based on varied sources, a functional use case diagram was developed to describe the behavior and quality attributes of the software.

Software Design Phase. Based on the requirements specifications, high-level modules that defined the major subsystems were identified. Each subsystem had to ensure that certain requirements were categorized while encapsulating the behavior of the whole system. Since object-oriented design was used, all identified classes and objects were modeled using the Unified Modeling Language (UML). Additional diagrams defined by UML were incorporated in the design as the need arises. The development of graphical user interface (GUI) screenshots helped in providing visual representation of the system. It helped validate the requirements specifications with the resulting system.

Software Implementation Phase. The software implementation phase focused on the transformation of the design into a formal object oriented programming language. Since object-orientation was used,

JAVA programming language was selected for implementation. In addition, Java syntax provides the developer the tools that can be used to examine functionality. The suitability of the programming language and other technological decisions are done in the software design phase. In order to provide readability of source codes, a coding standard was developed wherein every function component was indented during coding.

Software Testing Phase. The RMT is subjected to software testing to check its functionality. Only major test cases were done to ascertain that defects were found prior to the deployment of the system. In the development of this tool, a test strategy was developed to ensure that possible risks of the system are covered. The researcher conducted unit/module testing, integration testing, system testing, and acceptance testing.

Software Deployment. The software deployment phase is the formal turn-over of the requirement management tool to the En109 Robotics Laboratory for trial use. In cases where improvements in features shall be requested, the researcher is responsible for initiating enhancements of the requirements management tool.

RESULTS AND DISCUSSION

The Software industry flooded the market with requirements management tools. However, developers are still looking for a better tool that produces essential requirements documentation. The requirements management tool focuses on the needs of clients and developers in the requirements phase. The tool produced SUR, SRS, Change Requests, and Acceptance Criteria which were subjected to assessment of both parties (client and developer) for utilization. These are the key deliverables in the requirements phase and it can be used for obtaining signed-off from clients. The work context of the requirements management tool is shown on Figure 3.

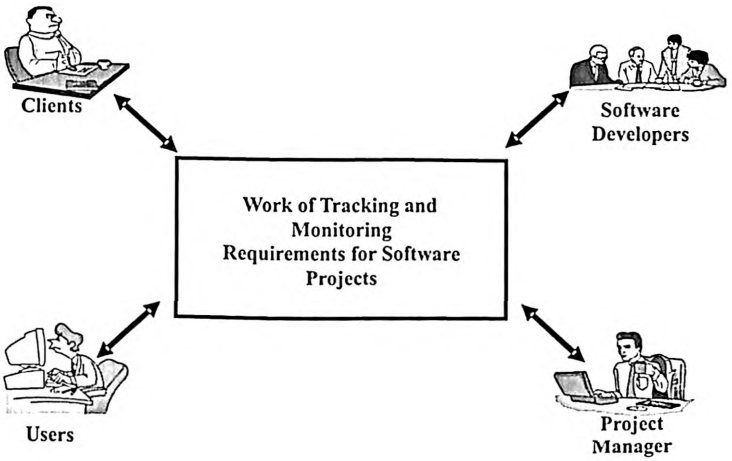

Figure 3. Work Context Diagram

The requirements management tool catered to software developers, project managers, clients, and users involved in software development. During the course of development, high-level requirements are documented; clients were requested to sign-off to signify their approval. The developers then elicited requirements from clients and users for specifications. These specifications were in the form of use cases and scenarios to identify and understand the behavior of the system. As the development goes on, clients or users request for changes in some of the specifications.

During deployment, the client was asked to test the system for conformance using the acceptance criteria defined by the given requirements. After all requirements are satisfied, the client sign-offto signify acceptance of the product. The process of documenting these activities is managed by the requirements management tool.

Use cases are derived from the business events identified in the work context diagram. They are units of work that describes the overall behavior of the system.

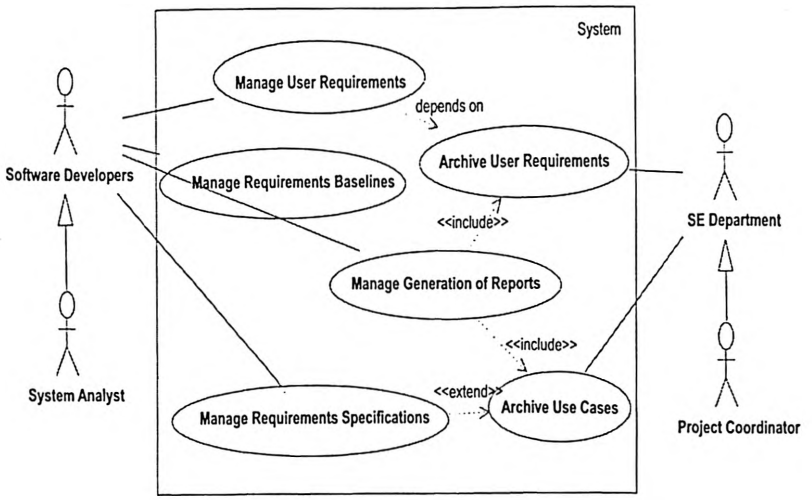

Figure 4. General Use Case Diagram

The general use case diagram shown in Fig. 4 described the overall behavior and functionality of the requirements tool. It managed user requirements supplied by users to the software developers. The system analyst, who is also one of the developers, managed the requirements specifications and kept track of completed requirements. These requirements were archived, together with the generated use cases that represented the work of the system. As a deliverable for the requirements tool, it generated reports intended for the client, customers, and project coordinator for review. The project coordinator monitored the completion of requirements and examined the reports submitted by the development team. The project coordinator, a representative of the SE Department, managed the execution of software development projects.

The major features of the Requirements Management Tool were defined by the menus provided in Figure 5. The essential features for managing user requirements and artifacts were considered.

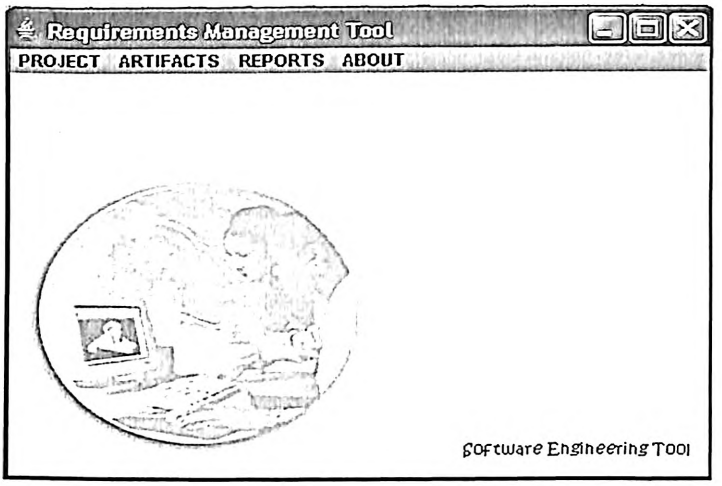

Figure 5. RMT Main GUI

The RMT Main GUI consisted of four (4) menu options, namely: Project, Artifacts, Reports, and About; that facilitated major interactions with the user. The Project Menu defined the project details for a software project where requirements would be tracked. The Artifacts Menu facilitated interaction with the user for details pertaining to user requirements, use cases, and requesting for change. It allowed the user to add, edit/view, and delete those artifacts within the database. The Reports Menu dealt with the list of reports that would be produced and major documents such as SUR, SRS, and Acceptance Criteria. The About Menu showed the additional information about the RMT system that the user needs to know.

Architectural Design

The development of the requirements management tool mainly focused on utilizing object-oriented development. A high-level view of the design is shown on Figure 6:

During decomposition, the logical packages are isolated from the package that defined the graphical user interfaces. This ensured high cohesion and low coupling of components that enhances performance of information flow. The user is the main driver for every transactions of RMT and the data is accessed through defined interfaces.

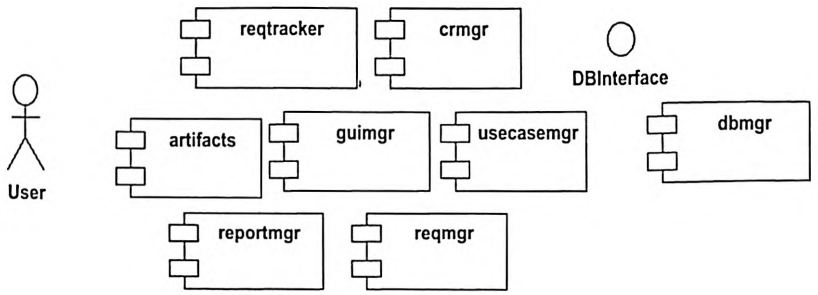

Figure 6. RMT High-Level Design

Design and Implementation Issues

The Requirements Management Tool is developed to manage the production of various requirements artifacts like SUR, SRS, and Acceptance criteria for sign-off. It allowed the user to provide the list of user requirements that would be incorporated in a software project. The user requirements types consisted of functional, non-functional, technical, organizational, and documentation requirements. Each requirement is transformed into use cases to derive the detailed requirements of the system. The development of RMT followed the object-oriented analysis and design methodology in which individual components are derived from the requirements and use cases. It is modeled using the Unified Modeling Language (UML) that represented the high-level design of components, its functions, and information flows. Individual classes are defined per component to simulate the functionality.

In the software construction, JAVA is used to formalize the individual classes and packages. Computer Aided Software Engineering Tools (CASE) is used such as BlueJ for the development of classes and NetBeans IDE for developing the graphical user interfaces (GUI) of the software. The database used is mySQL because it is free and easy to manipulate in creating tables. Jasper Reports for Java is used to create the individual report of the requirements tools such as requirements lists, artifacts reports, and other requirements-related documents.

During testing, minimal defects are found because of the help of CASE tools for development. It automatically created a warning of

defects prior to compilation and even made some notes on how defects could be fixed as in the case of NetBeans in which try-catch and fix imports could be generated automatically. Most of the defects are found during unit tests due to the inclusions of third-part application programmer's interface (API). The RMT tool is tested by some of the software engineering students with software projects and by colleagues within the SE department.

CONCLUSION AND RECOMMENDATIONS

In the light of the advancement of technology in providing comfort and utilization of various software applications to address the problem of requirements process and documentation, the Requirements Management Tool (RMT) software is developed. The Requirements Management Tool (RMT) is intended for the software engineering students in their software development projects. It caters to the documentation of user requirements as well as the production of major requirements artifacts like SUR, SRS and Acceptance Criteria. It addresses the problems of developers to make templates in documenting these requirements in a formalized format. As with the RMT, requirements artifacts are uniformly developed and follow the industry standard. Most of the problems in software development include keeping track of user requirements, but through this software, a checklist can be created in which the system analyst can determine the requirements status.

The RMT software utilizes the IEEE Software Engineering standards for documenting software requirements and it is recommended to address the need of formalizing the requirements artifacts. This document can assist project teams and make necessary change requests wherever if suggested by clients during the development. It can generate a list of requirements changes and determine whether those requirements changes are approved or not. The generation of reports for change request can give the developer the confidence that requirements changes are followed.

Requirements are essential in the development of a software product. Developers are now finding ways to improve the requirements process. The RMT software, as one of the CASE tools, can significantly increase the consistency of requirements documentation, and assist the tracking especially of the software project of software engineering students. However, software

developers can still improve the RMT software to address their specific needs in the requirements phase. Listed below are some of the recommendations for the RMT software:

1. complex generation of requirements list in which changes made in the requirements artifacts can be documented;
2. graphical representation of how requirements evolve during the development phase;
3. flexible documentation of requirements artifacts such as SUR, SRS and Acceptance Criteria;
4. software should be able to handle parallel multiple projects without corrupting data entities;
5. automated conversion of high-level requirements into use cases and use cases into detailed requirements; and,
6. since Java has limited or need-to-discover-GUI-components, there is still a need to make improvements on the look and-feel (L&F) of individual modules.

ACKNOWLEDGMENT

First of all, I want to thank my wife, Dorothy, for the love and support she has given me all these years. She has given me the strength to face the challenges of life.

I want to thank my family especially my mother who had been very supportive eversince. Thanks Nanay; you are the one who has taught me to be the way I am today. I also want to thank Auntie Elsie for her prayers and guidance in everything that I do.

I want to thank my friends and colleagues for giving me advice and also, a great thank you to all others who showed their interest in my interest and offered help.

Most of all, I want to thank God for all the blessings, good health, and the knowledge He has given me to overcome every difficulties that I've been through. He changed my views in life and gave me the strength to go on.

REFERENCES

SEEC Forum. *Division of information technology engineering and the environment.* Retrieved August 16, 2006, from http://www.seec forum.unisa.edu.au/SEECTools.html.

TRUEreQ *Object oriented product management.* Retrieved August 16, 2006, from http://www.truereq.com.

Change management and requirements tracing system. Retrieved August 16, 2006, from http://www.bandwood.com/cms_ exec_summary.htm.

MKS Requirements 2006. Retrieved August 8, 2006, from http://www.mks.com/products/rm.

International council on system engineering. Retrieved August 2, 2006, from http://www.paper-review.com/tools/rms/read.php.

Weigers, Karl E. (1999). Automating requirements management. *Software development magazine.*

Kuhn, S. (1998). The software design studio: An exploration. *IEEE Software*, March/April 1998.