A SOFTWARE TOOL FOR REQUIREMENTS SPECIFICATION
On using the STORM Environment to Create SRS Documents

Sergiu Dascalu*, Eric Fritzinger*
*Department of Computer Science and Engineering, University of Nevada, Reno,USA

Kendra Cooper**, Narayan Debnath***
**Dept. of Computer Science, University of Texas at Dalla
***Dept. of Computer Science, Winona State University,USA

Keywords: Software tool, requirements specification, use case modelling, SRS document, UML, CASE.

Abstract: STORM, presented in this paper, is a UML-based software engineering tool designed for the purpose of automating as much of the requirements specification phase as possible. The main idea of the STORM approach is to combine adequate requirements writing with robust use case modelling in order to expedite the process leading up to the actual design of the software. This paper presents a description of our approach to software requirements specification as well as an overview of STORM's design concepts, organizing principles, and modes of operation. Also included are examples of the tool's use, a comparison between STORM and similar CASE tools, and a discussion of needed features for software environments that support text aspects of requirements and use case modelling.

1 INTRODUCTION

Specification of software is a difficult and complex activity that requires substantial effort on the part of the requirements engineer (Endres and Rombach, 2003). There are many Computer Aided Software Engineering (CASE) tools that help with the development of software, but they rarely have much support for text descriptions of requirements, use cases and scenarios. Existing tools place less emphasis on text aspects of requirements modelling and Unified Modelling Language (UML) tools are largely diagrammatic editors (Booch, 2002; OMG UML, 2007).

We have created STORM, the Supporting Tool for the Organization of Requirements Modelling, to assist requirements engineers in the specification phase of the software life cycle. STORM is a UML software tool that incorporates requirements management and use case creation as outlined in (Arlow and Neustadt, 2002). This is done in an effort to help the engineer through the specification of the software. One of the ways that STORM assists the requirements engineer is in the automated generation of several artefacts of the software specification.

Although initiated in an academic environment, STORM answers real needs for speeding up the specification process by supporting the semi-automated generation of the Software Requirements Specification (SRS) document (Sommerville, 2006).

STORM has been introduced in a 2006 paper (Dascalu, 2006) that described its main concepts and provided excerpts from its software model and prototype in action. Since then, STORM has undergone further development and has evolved in terms of new features, operational capabilities, and reliability. The current paper provides an overview of STORM's functionality and highlights its new features available. Furthermore, the way we organize the SRS is used as the structure of the "target document" to be generated by STORM based on the requirements engineer's and system analyst's input on requirements, actors, and use cases.

This paper's contributions are as follows. First, we provide a description of a new operational tool for requirements specification that answers real software specification needs. Because UML contains no modelling constructs for expressing functional

Dascalu S., Fritzinger E., Cooper K. and Debnath N. (2007).
A SOFTWARE TOOL FOR REQUIREMENTS SPECIFICATION - On using the STORM Environment to Create SRS Documents.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 319-326
DOI: 10.5220/0001345303190326

and non-functional requirements there is lack of tools that support the process of defining and managing such requirements. Here is where the novelty of STORM comes in place, as a solution to fill in this existing gap in the landscape of CASE tools. Second, we identify and discuss a set of needed yet typically overlooked features of supporting tools in requirements and use case modelling. Third, through the tool described in this paper we propose an initial component in a suite of tools aimed at automating software modelling and documentation. For example, in future work, within the traditional (waterfall-like) timeline of the software process, before using STORM a new component tool would extract an outline of requirements from a project concept. At STORM level, specification patterns and evolved reusable requirements libraries could be supported by a new component of the tool suite (or by a future version of STORM). After using STORM, the artefacts it creates could be fed into a new component tool that would guide and support the automated generation of design models such as interaction diagrams, state-chart diagrams, and class diagrams (Liu et al, 2003).

The remaining of this paper is organized as follows: Section 2 provides related background information and further describes our motivation for creating STORM; Section 3 presents details of the SRS document we have used as target document to be generated by STORM; Section 4 overviews the tool's functionality and highlights its new features and capabilities; Section 5 provides examples of using STORM for requirements specification and use case modelling; Section 6 contains a comparison with related work; finally, Section 7 outlines directions of future work and presents our conclusions.

2 BACKGROUND AND MOTIVATION

Three of this paper's authors regularly teach, at different universities, undergraduate and graduate courses in software engineering. Typically, these courses include group projects that involve teams of 3 to 6 students each (Dascalu, 2005). At senior and graduate level courses we aim to emulate as much as possible an industrial environment in which students apply sound software engineering practices and develop in collaboration relatively complex software-intensive projects. As part of the educational and development process, the main phases of the software process (Pressman, 2006; Sommerville, 2006) are covered and student groups are required to take their projects from the successive phases as project concept definition, requirements specification, analysis, design, implementation, integration, testing, and delivery. Due to time constraints, in semester-long courses only development is covered while in year-long courses evolution is also involved (in the 2 or 3 iterations of the project).

What struck us from the beginning our teaching of these courses (about five years ago) was the lack of CASE tools for capturing requirements as well as textual descriptions of use cases and scenarios. Tools such as Rational Rose (IBM Rational, 2007) and MagicDraw (Magic Draw, 2007) have excellent capabilities for drawing diagrams and even generate code from software models but not sufficient facilities for the above – hence, the students had to write their project's SRS using regular text editors such as Microsoft Word or vi.

To speed up the generation of the SRS, we started about three years ago the development of STORM, a tool which has the goal of supporting requirements engineering and use case modelling and assist software engineers in generating the SRS for their software-intensive projects.

Our work on STORM has been initially materialized in a 2005 version of the tool (Dascalu, 2006). Further work has been done continuously since then on improving STORM and a second, 2006 version has been developed as part of a Master thesis in Computer Science successfully defended in June 2006 by one of this paper's authors (Fritzinger, 2006). This version, with its significant new enhancements, is discussed in this paper. Currently, work involving a couple of graduate students is ongoing on developing of a web-based version of STORM that will be likely made publicly available, as open source software, around the end of 2007.

Beyond its immediate scope and utility, we believe the STORM project is a worthwhile endeavour for several reasons. First, it is at the core of our research interests, focused on software specification. In fact, in terms of research STORM provides an excellent groundwork for investigating the principles of Model-Driven Development (MDD) and advancing work in this area (Stahl et al, 2006). Second, the tool's potential for application in practical, real-world applications is high (we are working to achieve this goal). Third, it constitutes a powerful educational tool that not only illustrates the significance of supporting tools in software engineering but also serves the requirements

specification and (partially) analysis phases of the software process. Furthermore, it helps the students create software models and generate documentation (the SRS) in a more effective way. In other words, STORM is placed at the confluence of *methods* (ways of performing software process activities that it supports), *models* (that it helps create), and *tools* (itself being a representative of this category), three fundamental pillars of software engineering.

3 OUR SPECIFIC SRS

Before looking at STORM's current features it is necessary to briefly describe the typical structure of the SRS used in our projects. As shown in Table 1, this structure has been kept to several minimal but essential sections in order to be suitable for a rather short, streamlined software engineering process applied in projects involving 3 to 6 developers over a period of 4 or 8 months.

Table 1: Structure of the target SRS document.

Section #	Section contents
0	Table of contents
1	Abstract / executive summary
2	Introduction
3	Requirements modelling: - functional requirements; - non-functional requirements. (prioritized on 3 levels)
4	Use case modelling: - use case diagram; - use cases using templates and text descriptions; - sample scenarios (primary scenario and most important secondary scenarios)
5	Requirements traceability matrix
6	Initial snapshots of the user interface
7	Glossary of terms
8	Resources and references
9	Contributions of team members
10	Appendices

Notably, the most substantial items in this SRS structure (sections 3, 4, and 5, which account for about 70% of the SRS size) are covered in STORM, as described next in the paper. All other items, such as the glossary of terms and the initial snapshots of the software system's planned user interface can be appended by the requirements engineer in the document file generated by STORM. Also, future needed extensions to the SRS will be implemented in STORM–diverse software modelling practices, including details of extended SRS documents are planned to be covered by new versions of the tool.

4 OVERVIEW OF STORM

STORM's main goal was to create a user friendly CASE tool capable of improved handling of text aspects of requirements specification and use case modelling, an area mostly neglected by the currently available CASE tools. Primarily, STORM provides support for managing requirements, use cases, use case diagrams, scenarios, and traceability matrix. This section briefly describes how each of these aspects is handled within the STORM environment.

4.1 Main Functionality

Requirements management. STORM supports software requirement management as outlined by (Arlow and Neustadt, 2002). The requirement statements are broken down into functional and non-functional requirements, and are sequentially numbered, e.g., R1, R2, R3, etc. This method allows the creation of the traceability matrix to be less tedious than it would be with other methods of requirements writing.

One of the other functions within this context is the ability to import and export requirements from/to a particular file format. Currently, STORM is compatible with the RUT (RUT, 2007), a requirements tool created by other authors in parallel with STORM. This feature is especially important when dealing with large projects that may borrow functionality from previous projects.

Actor management. The actors in STORM can be associated with use cases and can even inherit each other. An inheriting actor is automatically associated with all of the use cases associated with the inherited actor.

Use case management. Use case management in STORM was designed from the beginning to allow for automated generation of the use case diagram as well as of the primary and secondary scenarios. The goal of the use case form (template) is to structure everything so that it can do so. The interface changes as necessary to accommodate sequential actions, branching, and looping. One can even "collapse" particular branches to make the viewing of the use case easier.

Some of the functions included are the ability to inherit, extend, and include other use cases, denoted by the <<inherit>>, <<extend>>, and <<include>> stereotypes, respectively.

Misuse case management. Misuse cases (Alexander, 2003) are use cases that attempt to destabilize the system and cause havoc among its internal workings. These misuse cases add two new stereotypes for use case creation: <<threatens>> and <<mitigates>>. The idea is to find weak areas in the system and attempt to find use cases to mitigate the potential threats to the system.

Requirements tracing. The traceability matrix provides a mapping of requirements to use cases. In the form of a table, it shows what requirements are captured in what use cases. It is a good tool for tracing and verification, as well for showing at a glance which use cases are critical to the success of the system.

Scenario generation. STORM's scenario generation facility automatically creates the scenarios, both primary and secondary, based on the data entered via the use case template. This automated generation is done by taking all branches as separate scenarios and compiling them into a list. The user specifies the primary scenario on the use case form via radio buttons located at each branch.

Use case diagram generation. The use case diagram is also automatically generated based on the information input in the actor forms and use case form. The user can click and drag all of the individual components of the diagram in order to position them to the user's liking. In order to keep a clean look to the diagram all connections and system boundaries will auto-adjust to the movements. The only objects that need to be moved are the actors and use cases.

SRS document export. In addition to the features listed here, STORM can also export the software requirements specification document to either HTML or rich-text format (RTF). This is major feature, developed in an effort to allow the requirements engineer to generate and show the specification outside of STORM's interface and allows for a printable version of the specification.

4.2 Summary of New Features

As compared with the initial version of the environment, the last standalone edition of STORM provides the following major additions and enhancements:

- Requirements reuse via import and export facilities;
- Capability to generate the use case diagram from use cases;
- Capability to generate scenarios from use cases;
- Facility to generate SRS as an RTF document;
- Facility to generate SRS as an HTML document;
- Further tested software, packaged for use, applied in student projects.

5 EXAMPLES OF USE

In this section, samples of STORM uses are provided to illustrate the environment's main functions described in the previous section. Figure 1 shows the main menu of STORM, allowing the requirements engineer to select any part of the specification he or she might want to work on.

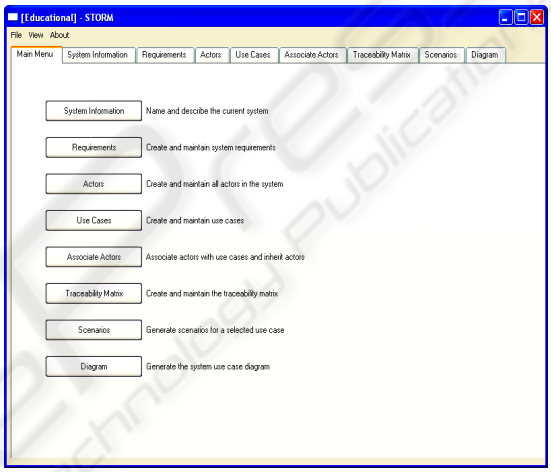

Figure 1: Main user interface of the STORM tool.

Figure 2 displays the requirements entry form, which in essence allows the user to specify if the requirement is functional or non-functional and select its priority.

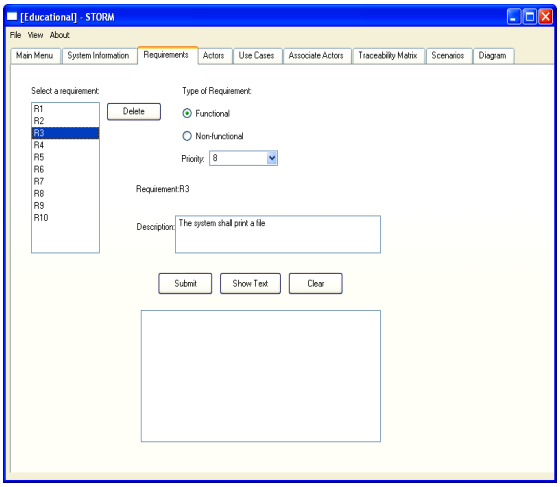

Figure 2: STORM requirements form.

Figure 3 presents the use case form (template), which allows the user to input use cases. A use case template contains preconditions, post-conditions, main flow of events, and alternative flow of events. Each statement in the main flow of events can be a basic statement, an if-statement, or a while-statement (loop).

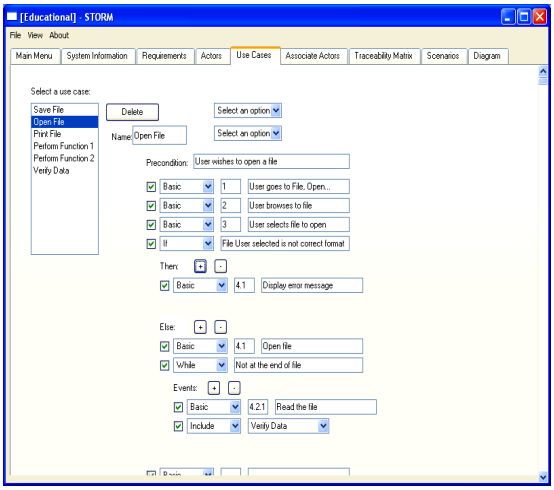

Figure 3: STORM use case form.

Figure 4 displays the associate-actors-to-use cases form. This form allows the developer to associate actors with use cases in addition to allowing the designer to make actors inherit other actors. This form allows the automated generation of connections between actors and other components of the use case diagram.

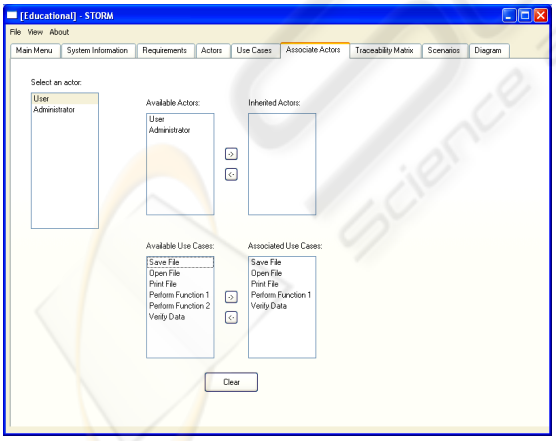

Figure 4: STORM form for associating actors with use cases.

Figure 5 shows how from the use case created in Figure 3 several scenarios were generated based on the information entered into the use case template. The primary and secondary scenarios are outlined, and the steps are shown as they would appear in an execution of the use case. One thing to note is how the loop is handled in the primary scenario. In general, it is difficult if not impossible to determine how many loops it would take to complete a task, so at this moment this aspect of scenario generation is handled only superficially in STORM.

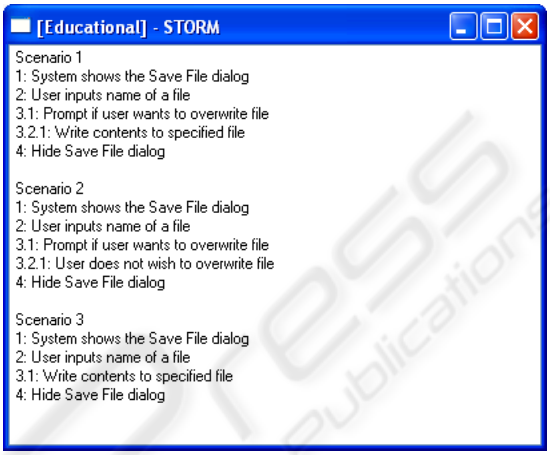

Figure 5: Example of generated scenarios in STORM.

Figure 6 contains a snapshot of the generated use case diagram. Here, the user can click and drag the components of the diagram to improve its graphical presentation and make it more "structured". This diagram was generated based on the information garnered from the use case form, the actors form, and the associate-actors-to-use cases form. All connections are created through the UML stereotypes (e.g., <<extend>> and <<include>>) specified on the associations made.

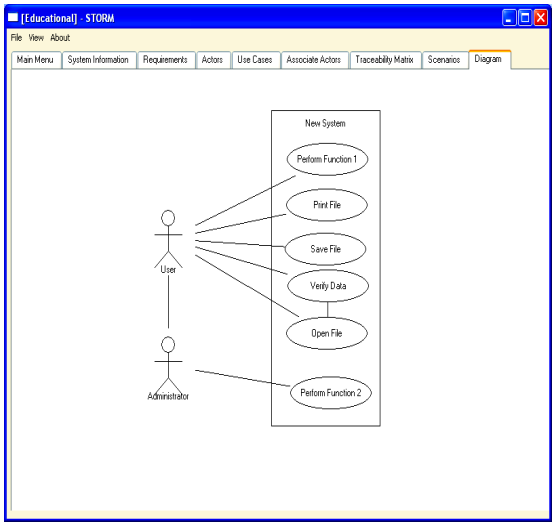

Figure 6: Example of generated use case diagram in STORM.

Finally, Figure 7 presents an excerpt of the generated SRS document as an RTF (Word) file. Due to space limitations, other features of STORM such as generation of the SRS as an html file or reuse of requirements using import and selection facilities, are not illustrated in the paper.

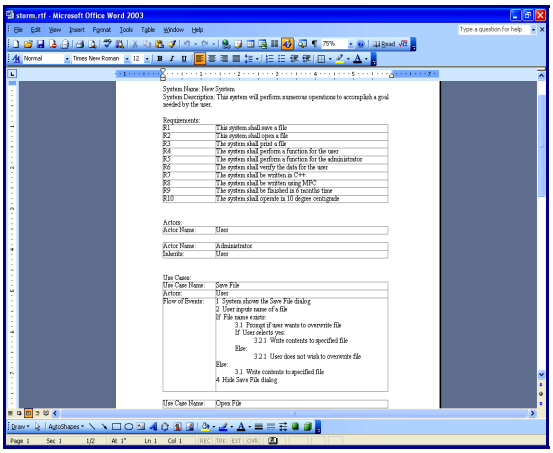

Figure 7: Excerpt from a generated SRS document as an RTF file in STORM.

6 COMPARISON WITH RELATED WORK

While STORM also incorporates support for graphical representations (use case diagrams), the defining feature of the tool is its ability to handle the text aspects of requirements specification and use case modelling. There are very few programs or add-ons on the market that support text in the way that STORM does, and none of them provide all features that STORM has to offer.

Among the closest related tools, DOORS is a requirements tool capable of handling very large commercial projects (DOORS, 2007). Scenario Plus is an add-on for DOORS which adds support for use cases and a use case diagram (Scenario Plus, 2007). Together they provide a powerful combination for managing and tracking requirements. This combination also allows for dealing with misuse cases and exporting generated documents to Microsoft Word. The main difference that sets STORM apart is the automatic generation of scenarios from use case templates, which STORM supports, but DOORS does not. Furthermore, STORM allows use case and scenario descriptions using the practical templates proposed by Arlow and Neustadt (2002), while DOORS does not.

The Open Source Requirements Management tool is a powerful open source requirements tool (OSRMT, 2006). It is available for download on sourceforge.net and has many features similar to DOORS'. This tool allows for prioritization of requirements and includes a traceability matrix but lacks support for use cases, scenarios, and use case diagrams. It is a good open source solution for requirements management, but compared to STORM it does not handle the use case modelling of a software system.

EventStudio is a CASE tool mainly concerned with the graphical representations of a system (EventStudio, 2006). It incorporates a scripting language called FDL that allows for dynamic generation of diagrams, including automatic generation of scenarios. Further, EventStudio can create images that can be scaled to different paper sizes and can export documents. However, it does not handle text aspects of requirements, use cases, and scenarios.

RUT is a use case tool developed by NASA's Software Assurance Technology Center (RUT, 2007). It is a web-based system that allows users to create projects, actors, and use cases much in the same way that STORM does. It is based on PHP and MySQL and performs text analysis to look for keywords that may add ambiguity to the use case text. RUT is a much focused tool that handles use cases very specifically. Nevertheless, STORM has a broader scope, providing support for managing requirements, use case diagrams, use cases, and traceability matrix.

Diagramming editors represent an important category of CASE tools that includes Microsoft's Visio (Microsoft Visio, 2003), MetaMill (MetaMill, 2007), and IBM's Rational Rose (IBM Rational Rose, 2007). These tools provide excellent support for diagrams and visual aids to assist in the development of software. They are some of the most popular CASE tools on the market. The difference between them and STORM is significant in that STORM's main focus is on the text aspects of software specification while the diagramming tools only truly support its visual aspects. While the visual aspects are very important, some specification depths are lost when only dealing with diagrams.

As a example of this class, Microsoft Visio is a diagramming tool among the most accessible and popular on the market. It handles only diagrams, and so is quite powerful when it comes to making graphical representations of a software system. However, there is little emphasis on text for specification of the system. There are workarounds for the lack of text support, for example use case objects having a "comments" field which can be used to list the steps in that use case, but this has no

Table 2: Comparison with similar CASE tools.

CRITERIA → TOOL ↓	Requirements modelling	Use case text description	Use case diagram generation	Automated scenario generation	Misuse case support	Export to document format	Full diagram capabilities
STORM	√	√	√	√	√	√	×
DOORS + SP	√	√	√	×	√	√	×
OSRM	√	×	×	×	×	√	×
EventStudio	×	×	×	√	×	√	√
RUT	×	√	×	×	×	×	×
MS Visio	×	×	×	×	×	×	√

functionality other than that for textual design. STORM was designed mostly for text aspects of requirements specification, which significantly distinguishes it from Visio.

The above are summarized in Table 1, which contains a chart comparing STORM to the CASE tools mentioned in this section.

7 FUTURE WORK AND CONCLUSIONS

There has been much accomplished in the STORM project to create a CASE tool capable of handling the text aspects of requirements and use case modelling. However, there is still more work that can be done to make the tool more complete and robust. The capabilities of STORM currently are sufficient to specify a system, but there is plenty of room for expanding it into other phases of the software process. In the long run, STORM could be the first component of a suite of tools capable of handling most of the software engineering process.

On a shorter timeframe, one major improvement to the environment can be to allow for the exporting and importing of more file formats. In particular, it will be useful to import and export to the XMI special file format, which is an XML-based diagram format that many of the commercial CASE tools support. Exporting and importing entities using this format would allow for more interoperability between STORM and other CASE tools.

Another enhancement is to allow for more robust scenario generation. As it stands, the scenario generation is handled almost completely by the tool. Allowing the user to pick which paths (i.e. prune the scenario tree) of a use case to generate related scenarios would be useful. Further, allowing the user to have control over loops in a use case which are rather superficially handled at present is another item of future work.

Collaboration can be a great asset, especially when it comes to designing software. One of the ways STORM could be applied to this is to create a web-based version, one that could be used to work with people all over the world in the specification and the design of a software system. In fact, a web-based version of STORM has already been started, with the related goal of using the open source software paradigm to allow for a repository of specifications that other requirements engineers could browse and use in software development.

There are yet more areas of STORM development. The following is a list of some more additions to the tool that can increase its utility:

- Enhanced requirements description and reuse, for example by supporting additional templates for requirements definition;
- More customization features to adapt STORM utilization to specific user preferences (e.g., choices of font types, font size, colors, etc.);
- Expanded scope of the tool by including support for more diagrams, particularly for analysis and design (e.g., class diagrams, activity diagrams, interaction diagrams) as well as for code generation;
- New features for integrating other SRS document figures and data (e.g. screenshots of the user interface, glossary of terms, and references).

STORM has a strong enough base in UML to be expanded in many different ways. Being text-oriented in nature, the tool would be complemented by having additional diagrams added to its functionality.

The goal of the STORM project was to create a CASE tool for the specific purpose of supporting the requirements and specification phases of software engineering (Dascalu, 2006). It has been designed as a tool to be used in an academic setting, in particular for supporting the way specification is done at the University of Nevada, Reno, USA. STORM is using

techniques garnered mostly from (Arlow and Neustadt, 2002) book, along with some further modifications from this paper's authors, who has also relied on some of Ian Sommerville's software engineering approaches and guidelines (Sommerville, 2006). There are few text-oriented CASE tools because generally the primary focus is on diagrams, charts, and pictures. The diagrams are important, but how the system functions is more important, which is often described also in the text of the specification, not exclusively in its diagrams. The need for text-oriented CASE tools is clear, but they are not as prevalent in the industry as diagramming tools. It has been shown, though, that reducing errors earlier in the software process will reduce time and cost of the project as a whole (Lauesen, 2002; Endres and Rombach, 2003). That is why it is important to specify the system with more than just diagrams.

In summary, the primary contributions of the STORM environment consist of the handling of textual aspects of requirements and use cases modelling, including via requirements reuse, as well as of the automated facilities of the tool. In particular, the generation and the exporting of a STORM project (SRS document) to rich-text format and HTML are useful from a practical point of view.

We see STORM as a promising component of a set of future interrelated tools that will incorporate MDD principles and will get closer to the ideal "software factory," where practically in no time desires (software requirements) are expressed by users and running code to achieve them is generated by the factory's suite of tools. From a general and realistic perspective on software development, this is still largely a utopian thought, but perhaps only for a limited time. In the past, advances in computing, including in software engineering, proved capable to exceed our expectations.

REFERENCES

Alexander, I., 2003. Misuse cases: use cases with hostile intent. In *IEEE Software* **20**(5): 58–66.

Arlow J. and Neustadt, I., 2002. *UML and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley.

Dascalu, S., Fritzinger, E., Debnath, N., and Akinwale, O., 2006. STORM: Software Tool for the Organization of Requirements Modeling, in *Procs. of the 6th IEEE Intl. Conf. on Electro/Information Technology (EIT-2006)*, IEEE Computer Society Press, pp. 250-255.

Dascalu, S., Varol, Y., Harris, F., and Westphal, B., 2005. Computer Science capstone course Senior Projects: From project idea to prototype implementation, *Procs. of the IEEE FIE-2005 Frontiers in Education Conference*, pp. S3J/1-6.

Fritzinger, E., 2006. STORM: Software Tool for the Organization of Requirements Modeling, Master thesis, University of Nevada, Reno, USA.

Booch, G., Rumbaugh, J., and Jacobson, I., 2002. *The Unified Modeling Language: User Manual*, Addison-Wesley, 2nd edition.

DOORS, 2007. Telelogic's DOORS. Requirements management traceability solutions. Available online as of March 31, 2007 at http://www.telelogic.com/products/doorsers/index.cfm

Endres, A., Rombach, D., 2003. *A Handbook of Software and Systems Engineering*, Addison-Wesley.

EventStudio, 2006. EventHelix, EventStudio System Designer 2.5 – sequence diagram based system design. Available online as of July 5, 2006 at http://www eventhelix.com/EventStudio/.

IBM Rational Rose, 2007. Available online as of April 11, 2007 at http://www-306.ibm.com/software/rational/

Lauesen, S., 2002. *Software Requirements: Styles and Techniques*, Addison-Wesley.

Liu, D., Subramaniam, K., Far, B.H., and Eberlein, A., 2003. Automating transition from use cases to class model. In *Procs. of the 2003 Canadian Conference on Electrical and Computer Engineering*, pp. 831-834.

Metamill Software, 2007, Metamill – The UML CASE tool, model UML diagrams, and engineer Java, C, C++, and C# code," Available online as of April 1, 2007 at http://www.metamill.com/

Microsoft Visio, 2003. Microsoft Office Online: Visio 2003 Home Page. Available online as of June 1, 2006 at http://www.visio.com/

OMG UML, 2007. Object Management Group – UML, Available online as of April 11, 2007 at www.omg.org

OSRMT, 2006. Requirements Management Tool. News. Available online as of July 1, 2006 at http://www.osrmt.com/

Pressman, R.S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 6th edition.

RUT, 2007. Requirements Use Case Tool. NASA. Assurance tools and techniques. Available online as of February 19, 2007 at http://satc.gsfc.nasa.gov/

Scenario Plus, 2007. Scenario Plus – templates & tools for scenario-based requirements engineering. Available online as of March 20, 2007 at http://www.scenarioplus. org.uk/index.html

Sommerville, I. 2006. *Software Engineering*, Addison-Wesley, 8th edition.

SourceForge, 2006. SourceForge.net: Use Case Editor. Available online as of July 1, 2006 at: http://sourceforge.net/projects/uced

Stahl, T., Voelter, M., Czarnecki, K., 2006. *Model-Driven Software Development: Technology, Engineering*, Management, Wiley & Sons.