Gathering Requirements from Remote Users

T. Leonard, V. Berzins, LUQI, and M. J. Holden

Luqi@cs.nps.navy.mil

Abstract

We describe a distributed requirements engineering environment using computer aided software engineering tools linked together through the Internet. We created this distributed requirements engineering environment using Microsoft's Personal Web Server (PWS), Microsoft's Open Database Connectivity (ODBC) technology, Netscape Communicator, Microsoft's Internet Explorer, Microsoft's Access97 database, and a set of PERL scripts that are executed by users of the environment to perform database operations. We show how we added basic security features to the Internet accessible database.

1. Introduction

This paper shows how Front Loaded Accurate Requirements Engineering (FLARE) Teams are a means to realize accurate requirements early in project development and a means to ensure that requirements are satisfied in the implementation domain by automated systems [1]. These Teams can use Internet technologies to enhance the effectiveness of the set of CASE tools [2] that they use to manage requirements. Ideally, the set of requirements should be managed throughout the evolution of the system to provide the rationale for the system's behavior [3].

We show how to extend formal and informal specifications with audio and video file representations of requirements [4,5]. This is valuable because video allows developers to quickly gain a conceptual understanding of the problem domain. breaks the mind-numbing monotony often experienced when reading formal textual specifications and graphical diagrams and effectively provides an abstract representation of objects found within specifications.

Additionally, we demonstrate how Internet technologies can help managers improve their task assignment methods by incorporating team members' assessments of the difficulty of implementing software components into the decision process. The products produced by the FLARE Teams are used to make this possible.

2. CASE environment enhancement using internet technologies

The number of computer aided software engineering tools and environments available to assist FLARE Teams is extensive. Queen's University in Kingston, Ontario publishes a partial CASE tool list that has nearly 400 tools listed [6]. Using a small subset of available CASE tools commonly used at the Naval Postgraduate School, we show how to enhance a CASE environment with Internet technologies.

Researchers at the Naval Postgraduate School have developed a CASE tool called Computer-aided Prototyping System (CAPS) [7]. This tool provides a capability to develop prototypes using the prototype system description language (PSDL) [8]. Once completed, CAPS promises to provide a robust environment that will facilitate the management of requirements throughout the life of a system.

By design, the prototypes produced using CAPS are demonstrated to users. Users evaluate the prototypes, and developers use the information obtained from the user's evaluation to refine the requirements of the software system [9]. CAPS would produce the best results if a developer personally presented a prototype to a user, but this would be expensive in terms of travel and set up time, especially if multiple meetings between a developer and user were needed. By using audio and video conferencing techniques over the Internet, similar to those described by Macedonia and Brutzman [10], we can remove this limitation. Additionally, the developer's ability to interact with the user at any time, provided the user has access to an Internet enabled computer with video conferencing capabilities, would enhance the engineering environment created by CAPS and tools similar to it. This enhancement would be achieved by allowing the developer to resolve ambiguous requirements with users while they are still actively involved in the process of developing a prototype or model. It also would reduce the cost of travel by

eliminating the requirement of having the developers and users co-located during the presentation of a new or changed prototype. Applications exist on the market that make this possible with a modest, under $1,000.00, investment in additional equipment and software [11,12].

The use of Internet video conferencing to augment a software-engineering environment is available today as are other Internet technologies providing comparable enhancements. One of these additional Internet technologies is "intelligent browsing" [13]. It is now possible for a Software Engineer to use intelligent agents to retrieve information from the Internet [14]. These tools can aid Software Engineers in their efforts to understand the problem domain and to find appropriate solutions to requirements in the implementation domain. Enhanced with intelligent agents and Internet video conferencing, this Software Engineering environment should improve significantly the production of quality software in a timely manner.

3. Augmentation of formal and informal specifications with video

Where appropriate, FLARE Teams will augment requirement specifications with video representations of the requirements. This can be helpful because certain requirements can be better understood by developers if they can observe users during the performance of the activities that generate the requirements [4]. For example, most software developers are not experts in infantry fighting procedures and have no concept of the actual tactics, techniques and procedures used by infantry forces to accomplish their assigned mission. This lack of understanding of the problem domain is further complicated by preconceived ideas formulated by software developers as they are exposed to the entertainment industry's dramatization of infantry soldiers and their fighting techniques. Problem domain objects and concepts such as foxholes, fields of fire, accuracy, timeliness and cover have very specific meanings to infantry soldiers. Typical Software Engineers do not necessarily share these meanings. Software Engineers can represent each of them with formal or informal methods. However, would a Software Engineer located in Silicon Valley, when given a formal or informal representation of the requirements elicited from this problem domain, be able to design a system that would satisfy the needs of the user without direct knowledge of the user's context? The requirements produced from this type of problem domain, a domain that is foreign to most Software Engineers, is an ideal situation to use video to augment requirements. In this section, we show how the addition of a type "video", similar to that of a textual comment, to formal and informal specification and design languages will increase developers understanding of the problem domain.

3.1. The new memory paradigm

In the use of video to augment the representation of requirements, we would like to have easy and quick access to it. The cost of existing magnetic storage has recently dropped to affordable levels, making storage of video on fast hard disk drives feasible. Figure 1 depicts the dramatic reduction in memory prices that have taken place during the ten-year period between 1987 and 1997. It is now economically feasible to augment requirement specifications with video that is retrievable by anyone in the software development group on demand. This capability is the crux of bringing the application domain to the design and implementation domains. Another storage technology, digital versatile disk (DVD), introduced to the masses in 1997, provides additional space to store audio and video files [15].

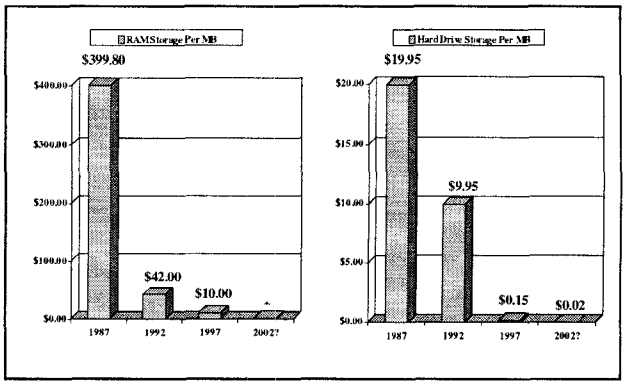

figure 1. Dropping memory prices. In 1987, the cost of secondary magnetic storage, hard drives, was about $20.00 a megabyte (mb) [16: p. 89], and primary memory, random access memory (ram), was about $400.00 a mb [17: p. 309]. In January of 1992 this dropped to about $10.00 a mb for hard drive storage and $42.00 a mb for ram [18: p. 356]. In January of 1997, both types of memory were at an all-time low. hard drive storage sold for about $.15 a mb [19: p. 152], and ram for about $10.00 a mb [20: p. 153]. The prices shown for the year 2002 are based on a 30% yearly reduction in memory prices [21]

The same technology that allows analyzing "a golf swing from up to nine different camera angles" [22] can be used to provide on-demand video to increase software and systems developers' understanding of the problem

domain. Adding a type "Video" to specification and design languages provides an easy way to incorporate the use of video into the software engineering process.

3.2. Using video with formal methods

We use the Spec Language [1] to illustrate how FLARE Teams would incorporate video into the development process. This technique requires the representation of the Spec model in HTML format [23] and the use of an Internet browser to access the model. When appropriate, we use comments in Spec models stating that a video clip is available and we hyperlink these comments to video clips. This is attractive because it allows the designer to quickly augment a model with video using comments. Figure 2 shows how to implement this method. When an engineer reads the specification using an Internet browser, the engineer can click on the comment to view a video clip of the object being defined.

```
DEFINITION bunker -- Concept for describing shelters. - - Click
here to view video clip
  INHERIT fortification - - The module "fortification" defines types
security and cover.
.
.
END
```

Figure 2. The bold text on the first line is linked to a video file describing a bunker.

4. Programmer input into the work tasking process

Once designers have identified modules that require implementation, management must produce a programmer work schedule [24]. The products produced by FLARE Teams enable programmers to gain a better understanding of problem domain concepts and objects. This increased understanding makes programmers' input into the module assignment process used by managers more valuable.

Each programmer knows their programming abilities and can estimate the time to complete a programming task. We show how managers can assign tasks to programmers based on these estimates. We use Internet technologies to produce an interactive module evaluation environment where each uncommitted programmer rates unassigned modules by perceived level of difficulty. This process allows managers to produce an optimal programmer work schedule, minimizing the cost of implementing all unassigned modules, measured in terms of time, where a shorter implementation time is better.

We have developed an Internet form that executes a PERL script located on a server to access a Microsoft Access database using Microsoft's ODBC technology to capture programmers' assessments of tasks (Figure 3). [25,26,27,28,29,30]

The method used to gather input requires that each programmer estimate the number of days it would take to implement the module listed on the form. Each programmer is required to repeat the process until they meet one of the following three criteria. They have identified a module that they can implement in minimal time, and the system schedules the programmer to implement it; the programmer has evaluated all modules that have not been scheduled; or management stops the process because they have determined a suitable working schedule.

Once the Send button is pressed, the input provided is automatically transferred to a central location where it is processed. The scheduling process can be automated using techniques similar to those of the Evolution Control System (ECS) [31], incorporating programmer input into the system. The modified ECS can enforce various policies ranging from scheduling a task immediately if a programmer estimates they can complete it in .5 days, to waiting until each programmer has evaluated all modules in an attempt to develop an optimal schedule.

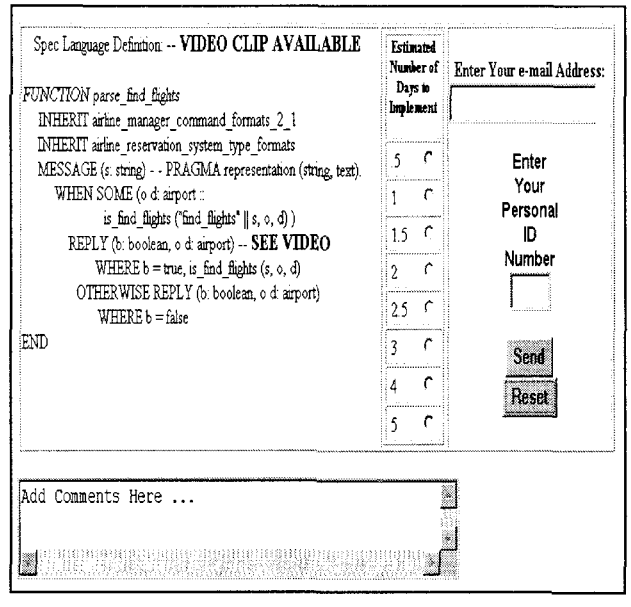

Figure 3. Form used to capture programmers' assessment of the time needed to implement a module. V. Berzins developed the spec language definition [1: p. 424].

This system can be used to assess schedule risk. Modules with a wide variance in programmer estimates are more likely to cause problems than the modules

where most of the programmers agree on the time it would take to implement. This system gives managers the ability to base decisions on this information. For example, programmers who continuously have a large variance between estimates and actual implementation time may require additional training on understanding specifications.

Incorporating programmers' assessments into the process also allows management to assign tasks to programmers in a way that takes advantage of each programmer's personal expertise and preferences. Each programmer has one or more classes of problems they can easily solve due to their accumulated experiences and habits. Incorporating their input into the module scheduling process would most likely increase their productivity because they would be assigned modules based on their completion time estimates. Additionally, this system can be utilized to focus recruiting efforts. Consider a situation in which the entire set of programmers rated tasks C, D, and E as taking the maximum allowable time to implement. Management might focus more of their recruiting efforts on finding individuals that rate these tasks as taking less time to implement, thereby increasing the efficiency of the entire organization and minimizing costs.

5. FLARE: a requirements engineering environment

We stated that traditional software engineering environments could be easily enhanced using Internet technologies. We offer evidence for this by introducing a CASE tool called FLARE that uses the technologies presented in Section II to create a distributed requirements engineering environment. FLARE is designed to enhance the software development process by offering a means to inexpensively manage requirements and facilitate communication of requirement related issues between all interested parties in the software development process. The FLARE Team discussed in Section I would use this tool.

5.1. FLARE's components

FLARE is composed of the following programs that use the Internet to exchange information. This coupling produces a synergistic effect by combining the distinct features of each program to produce a requirements engineering environment.

5.1.1. Microsoft's Access 97. An inexpensive database designed to function on Windows 95 or Windows NT, this database makes it relatively easy to manipulate the requirements engineering information entered into the FLARE environment. It also produces reports in HTML format, enabling users of FLARE to easily publish information that has been manipulated by database methods to the Internet. [32]

5.1.2. An access database file. This database file contains the tables, queries, forms, reports, and macros that constitute the management aspects of FLARE. [33]

5.1.3. A set of PERL scripts running on a PWS. Users of the environment access the environment's database by executing PERL scripts located on an Internet server. The PERL scripts are called using FLARE's user interface (Figure 4) that is accessed from an Internet browser.

5.1.4. A set of JavaScript enhanced HTML Files. When accessed with an Internet browser, this set of files creates the user interface for the FLARE environment. The essential elements of these files are embedded JavaScript [34] and Forms [35]. JavaScript enables the pull-down menus found in the user interface to function. The ability to input and transmit information is made possible by using Forms embedded in FLARE's HTML files.

5.1.5. A JavaScript enabled Internet browser. The browser is the shell that the user interface of FLARE runs in. It must be JavaScript enabled to allow the pull-down menus to operate. We used Microsoft's Internet Explorer [32] and Netscape's Communicator [36] to test the user interface.

5.1.6. Microsoft's PWS. We chose this web server to use in the environment because Microsoft freely distributes it.Microsoft's PWS.

5.2. FLARE's user interface

Figure 4 shows the initial user interface of FLARE. Each of the four pull-down menus represents a phase in the software life cycle. Each menu shares the "Mission Needs Statement" option. We chose to include this in each phase to emphasize the needs of the customer. The arrow between the "REQUIREMENTS" and "DESIGN" menus symbolizes communication between the two phases in the form of requirements specifications. The arrow between the "DESIGN" and "IMPLEMENTATION" phases symbolizes communication between the phases in the form of a formal or informal design specification. The arrow between the

"MAINTENANCE" and "REQUIREMENTS" phases symbolizes the transition caused by new or changing requirements. The "TESTING" icon in the center of the interface with arrows radiating in the four directions symbolizes the testing that must be built into each phase of the development cycle.

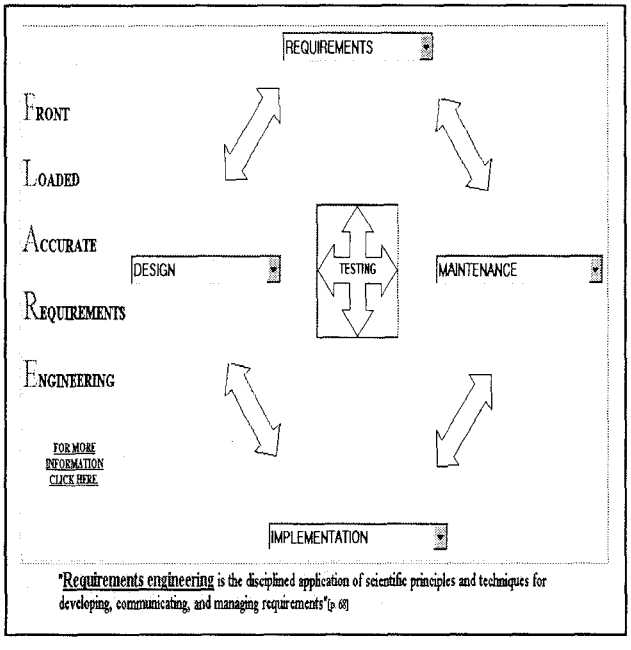

Figure 4. FLARE user interface.

We describe the functionality of each menu option in the following subsections.

5.2.1. Requirements pull-down menu. Enter Requirement: This option allows a Software Engineer to

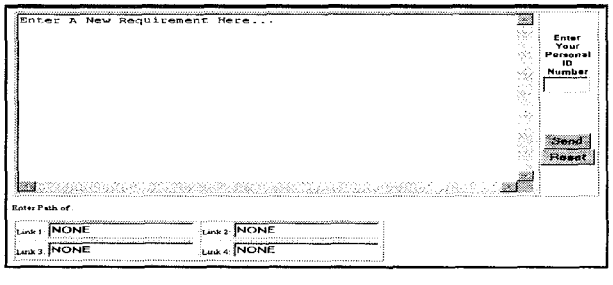

Figure 5. Requirements entry form

input a new requirement into the FLARE environment. Figure 5 shows the format of the form. The four link fields at the bottom of the form are provided to allow FLARE Team members to include logical links to video or other file representations of requirements. If an engineer entered a requirement that contained the problem domain object "foxhole", and the engineer had a 30 second video clip of a foxhole, then the engineer could add a comment in the requirement indicating that a video file of a foxhole was available at link 1. We choose to label these fields 'links" because the FLARE Team could use any file type to augment the requirements.

View Requirements: This option allows engineers to view all approved requirements. Figure 6 shows the HTML page that the database generates automatically when given the "Save as HTML" command found on the menu bar of the Access database.

Ask a Req. Question: This option allows a way to input questions about requirements into the FLARE

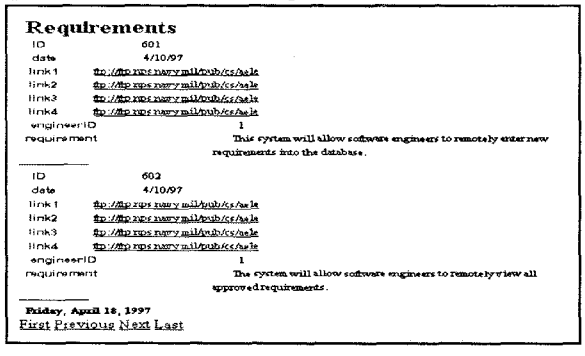

Figure 6. Database generated html Page.

system. It is similar in appearance to the form in Figure 5. Upon activation of the send button, the question is automatically imported into the database where a FLARE Team member using the form shown in Figure 7 can answer it.

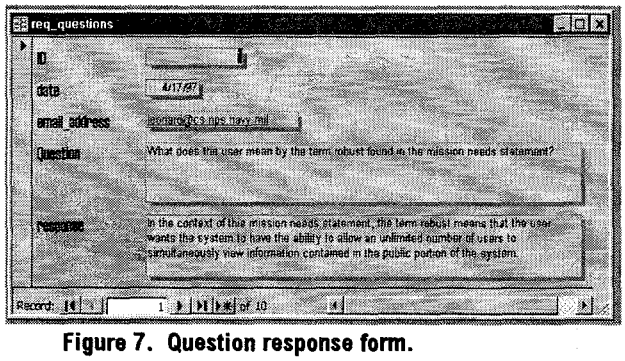

Figure 7. Question response form.

View Requirements Questions: This option allows users to view questions that have been asked along with the answers provided by engineers using the form shown in Figure 7.

We focus on intelligent assistance for organizing questions and answers rather than on intelligent agents for answering questions directly because the domain addressed by the questions has not been accurately formalized - otherwise there would be no need for a requirements determination process. The desired assistance is in locating similar previous questions and the associated answers derived from human expert

sources. Our initial approach for providing this assistance is to use a concept similarity lattice to define a distance between questions. This enables computing the nearest neighbors of a given question among the questions previously asked. We are exploring the use of a domain-specific core vocabulary to organize the questions more accurately than would be possible using a generic lexicon. The core vocabulary corresponds to the jargon of the problem domain. Locating repeated words in documents from the application area and filtering out noise words such as articles and prepositions can identify the core vocabulary. This vocabulary must be formalized into a concept hierarchy by knowledge engineers.

Mission Needs Statement: This option allows engineers to review the mission needs statement that prompted the development of the software system.

5.2.2. Design pull-down Menu. Enter Specification: This option allows a Software Engineer to input a specification that satisfies a requirement. Figure 8 shows the format of the form used. Note the fields labeled "Requirement ID." These fields facilitate requirements management. When a specification is entered into the system, the engineer should also enter the requirements that are associated with the specification.

The link fields on the form allow engineers to enter informal design specifications into the FLARE environment. An engineer would enter a comment in the text area of the Specification Entry Form indicating that

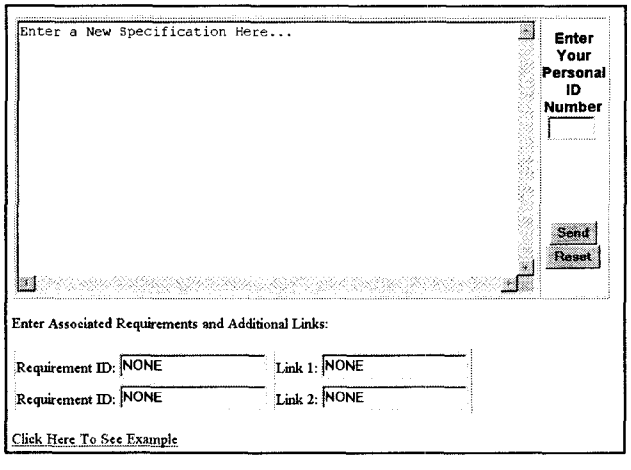

Figure 8. Specification entry form.

a hyperlink to a graphical model or specification exists. An informal specification would likely be in the form of a graphical model such as those found in the Unified Modeling Language [37].

Remaining Menu Options: The menu options View Specifications, Ask a Specification Question, View Specification Questions, and Mission Needs Statement are very similar to those found in subsection B-1 above and do not require further explanation.

5.2.3. Implement pull-down menu. Enter Estimates: The functionality of this option is thoroughly described in Section IV.

Remaining Menu Options: The menu options View Specifications, View Requirements, Ask an Implementation Question, View Implementation Questions, and Mission Needs Statement are very similar to those found in Subsection B-1 above and do not require further explanation.

5.2.4. Maintenance Pull-Down Menu. Enter a Bug Report: This option allows a Software Engineer to input an error found in the implementation into the flare system.

Enter Change Request: This option allows a Software Engineer to input a new or changed requirement into the FLARE environment.

Remaining Menu Options: The menu options View Bug Reports, View Change Requests, Ask a Maintenance Question, View Maintenance Questions, and Mission Needs Statement are very similar to those found in subsection B-1 above and do not require further explanation.

5.3. FLARE database

The database portion of FLARE's environment is implemented with Microsoft's Access database. The conceptual schema for this database is shown in the

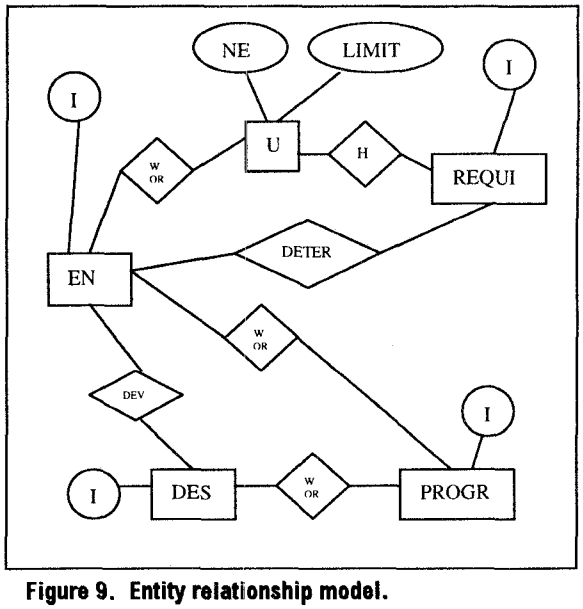

Figure 9. Entity relationship model.

entity relationship model [38] in Figure 9. The data

requirements of the FLARE system are the storage of user needs and limitations, the storage of the required software system capabilities needed by users to solve their problems and the storage of implementation domain information. The required implementation domain information consists of storage of engineer, programmer and design information. FLARE draws a distinction between programmers and engineers because they perform completely different functions and have different responsibilities. This structure supports FLARE Teams by providing a means to manage the information gathered during requirements elicitation. Users have needs, and FLARE Teams are responsible for determining the requirements of software systems that will satisfy these needs. Programmers and other engineers will use the products produced by FLARE Teams stored in the FLARE database to achieve their respective tasks.

5.3.1. Scheduling Algorithm. We developed and implemented a scheduling algo-rithm that automatically assigns tasks to programmers. The algorithm uses programmer's estimates (see Section IV) of the difficulty of translating a design specification module into a programming language implementation. The algorithm uses a greedy strategy [39:p.329]. It picks the lowest estimated time to implement a module and assigns that module to the programmer who made the estimate. The algorithm does not produce optimal results in all cases, yet provides a close approximation; this is acceptable given that the heuristic in which the algorithm determines a schedule is based on imprecise estimates.

6. Security

The initial distributed Internet database application was totally insecure. It was composed of freely available programs that were downloaded from the Internet. We were pleased with the availability of programs that allowed us to quickly create the distributed database environment, but recognize that in the context of security, we are completely at the mercy of the people who developed the components. The products could contain Trojan Horses or undiscovered viruses.

Three of the five components we used have binary files that potentially could contain viruses or Trojan Horses. The PERL library files are too large for us to ensure that they do not contain back doors. The Microsoft products were free of cost, but the licenses specifically state that it is a violation of the license agreements to reverse engineer the code. This makes it impossible for us to make any claims about the security of the system regardless of the security methods that we implement. The application's insecurity made it less than optimal for any purpose other than using it as a teaching or experimentation tool. This section details the steps taken to implement a basic level of security features in the system making it usable by a broader class of people.

We introduced basic security features into the distributed database environment, addressing the four major areas of network security: "Secrecy or Confidentially; Accuracy or integrity; Authenticity; and Availability." [40: p. 202]

6.1. Securing the database

This section shows how we added basic security features to the distributed database environment. The features that we added will keep most unauthorized people from adding, deleting and modifying database records. The features that accomplish this are: identification and authentication; execute, read and write restrictions placed on the server; restricted ftp services, and physical security of the server that has the distributed database located on it. [41]

6.1.1. Identification and authentication. We used a simple database table to implement identification and authentication. In order for users to add, delete, or modify records they must now enter a proper "userID and password" pair. Users must possess a valid userID and password pair to access the database. The userIDs and passwords entered by users should be sent over the Internet encrypted. (Figure 10)

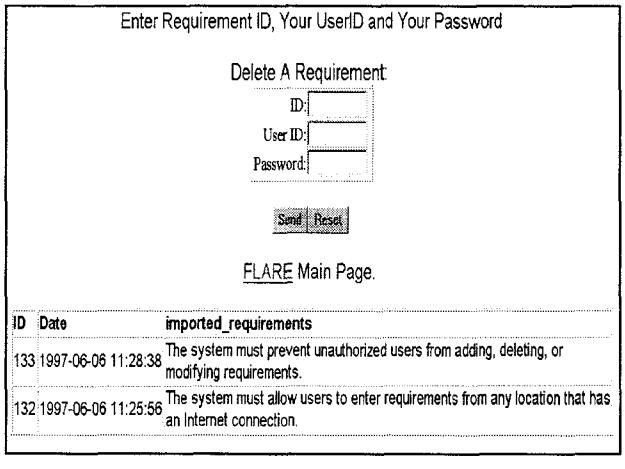

Figure 10. Requirement deletion form with identification and authentication features.

6.1.2. Execute, read and write restrictions. The PWS allows administrators to configure the permissions on directories. We use this to restrict what files users of our

Internet database can read, write, and execute. We use the same procedures Microsoft recommends for users of Active Server Pages.

We placed all HTML files used to provide a portion of the database's user interface in a directory that is readable but does not allow users to write files or execute programs. The directory is readable because users of the system need to read the HTML files in their Internet browsers to display our pages.

We placed all the PERL scripts in a directory that is executable but does not allow users to read or write files. Since the directory is executable, users can execute the PERL scripts that perform the database queries. Read permissions are denied because users do not need to view the scripts to perform database operations. If users were to read the PERL scripts, they could possibly find ways to gain unauthorized access to the data stored in the database or system files.

6.1.3. Restricted FTP services. The PWS has an FTP server. We configured it so that users are not allowed to access the directories containing the database, HTML and PERL script files.

6.1.4. Physical security of the database Server. The computer which stores the environment's files offers no built in security because it is running on Microsoft's Windows 95 operating system. We have placed the computer behind locked doors with limited access. This provides a reasonable level of assurance that unauthorized people will not be able to modify the files stored on the computer. It does not prevent personnel who routinely work in the area from making unauthorized modifications to the environment's files.

6.2. Additional threats and vulnerabilities

6.2.1. Denial of service attacks. The system does not have any mechanisms to deter or prevent a denial of service attack. Our environment is particularly susceptible to this type of attack. It is not behind a firewall, and it uses the PWS, which has a 64 simultaneous user limit.

6.2.2. Normal hardware failures. The Internet database is stored on magnetic hard drives. A hard drive failure would cause a catastrophic data loss. As usual, backups will mitigate this loss.

6.2.3. Accidental data loss. The database allows authorized users to delete a record accidentally. In addition, the computer that runs the Internet database is used for other purposes than just as a server for our Internet database. Other users of the computer could accidentally delete the database file.

6.3. Countermeasures

We find the most important countermeasures to the threats listed above are an aggressive backup policy and user training. The data stored in the database is the most important element of the distributed environment. Continuous backups using the replication features found in Access are warranted. This countermeasure provides a degree of protection for the other vulnerabilities we listed. User training is an important countermeasure because the system currently allows any user with a valid userID and password pair to delete or modify database records. Users of the system must be trained to limit deletions and modifications, and to take extra care when doing so.

Countermeasures to denial of service attacks are expensive. Firewalls are a way to counter this type of attack [40,41]. Users of the environment should determine when such countermeasures would be cost effective to implement.

7. Conclusions and future work

Matching related statements has other practical and important applications in software engineering, including comparing incoming problem reports to previously received ones to determine whether the reported problem is new or has been reported previously.

As a decision support aid for managers, software engineers and other stakeholders in software development projects, this tool can significantly reduce time spent in preparing and delivering answers to routine questions as well as improve the correlation between stakeholder identified requirements and final production software. The tool and the approach are not limited to the domain of software engineering - any endeavor with frequent interaction between producer and consumer can benefit from its use.

We developed the initial version of a CASE tool FLARE quickly (2 months). FLARE is a requirements engineering environment composed of commercial off the shelf (COTS) software tools tied together by the Internet. Our experience indicates that tools of this kind can be useful for collecting requirements from stakeholders at a variety of physical locations.

FLARE's requirements tracing features could be improved. Even though each requirement receives a unique identification number, FLARE does not automatically track this requirement throughout the

development process, rather it relies on engineers "tagging" each new product with the appropriate requirement identification number. Automation of this tagging process would eliminate possibilities of entering incorrect tags, and could potentially allow engineers to look at any object produced in the development process and extract the associated requirement information.

Microsoft's Access database provides the ability to automatically generate HTML files based on the information submitted by the software engineering team. These files are crude. A more sophisticated HTML file generator would be useful.

The greedy strategy used by the module assignment algorithm does not guarantee an optimal solution although it is good enough for initial use. Better algorithms should be explored.

More refined models of who is authorized to modify or delete which parts of the database would help to further improve security with respect to data integrity and protection from data losses.

A demonstration of the FLARE tool can be downloaded from or run directly on the World-Wide Web at URL: http://web.nps.navy.mil/~aeleonar/Welcome.html

8. References

[1] Valetto, G. and Kaiser, G., "Enveloping Sophisticated Tools into Computer-Aided Software Engineering Environments," in Proceedings 7th International Workshop on Computer-Aided Software Engineering, IEEE Computer Soc. Press, Los Alamitos, Calif., 1995, pp. 40-48.

[2] Berzins, V. and Luqi, Software Engineering with Abstractions, Addison-Wesley Publishing Company, Reading, MA, 1991.

[3] Ramesh, B., Powers, T., Stubbs, C. and Edwards, M. "Implementing Requirements traceability: A Case Study," in Proceedings Second IEEE International Symposium on Requirements Engineering, IEEE Computer Soc. Press, Los Alamitos, Calif., 1995, pp. 89-95.

[4] Brun-Cottan F. and Wall, P., "Using Video to Re-Present the User," Communications of the ACM, Vol. 38, No. 5, May 1995, pp. 61-71.

[5] Kaiya, H., Saeki, M. and Ochimizu, K., "Design of a Hyper Media Tool to support Requirements Elicitation Meetings," in Proceedings 7th International Workshop on Computer-Aided Software Engineering, IEEE Computer Soc. Press, Los Alamitos, Calif., 1995, pp. 250-259.

[6] CASE tool index, Queen's University in Kingston, Ontario, available from http://www.qucis.queensu.ca/Software-Engineering/tools.html Internet; accessed 22 March 1997.

[7] Luqi, Ketabchi, M., "A Computer-Aided Prototyping System," IEEE Computer Technology Series, Computer-Aided Software Engineering (CASE), Editor: E. Chikofsky, 1988, pp. 89-95.

[8] Luqi, Berzins, V. and Yeh, R., "A Prototyping Language for Real-Time Software," IEEE Transactions on Software Engineering, Vol. 14, No. 10, October 1988, pp. 1409-1423.

[9] Luqi, "Software Evolution Through Rapid Prototyping," IEEE Computer, May 1989, pp. 13-25.

[10] Macedonia M. and Brutzman, D., "MBone Provides Audio and Video Across the Internet," available from: ftp://taurus.cs.nps.navy.mil/pub/mbmg/mbone.html Internet; accessed 22 March 1997.

[11] Progressive Networks, "Real Audio and Video," available from: http://www.real.com/rvnba.html Internet; accessed 22 March 1997.

[12] Intel, "Internet Video Phone with Proshare Technology," available from: http://connectedpc.com/iaweb/cpc/iivphone/index.htm Internet; accessed 22 March 1997.

[13] O'Leary, D., "The Internet, Intranets, and the AI Renaissance," Computer, January 1997, pp. 71-78.

[14] Autonomy Corporation, "Autonomy Agents," available from: http://www.agentware.com Internet: accessed 23 March 1997.

[15] Poor, A., "DVD and CD-ROM: 21st Century Storage," PC Magazine Online, available from: http://www.pcmag.com/features/cdrom/_open.htm Internet: accessed 23 March 1997.

[16] Computer Mail Order, advertisement, Byte, McGraw-Hill, Peterborough, NH January 1987.

[17] Turner Hall Publishing, advertisement, Byte, McGraw-Hill, Peterborough, NH January 1987.

[18] Nevada Computer, advertisement, Byte, McGraw-Hill, Peterborough, NH January 1992.

[19] Computerlane, advertisement, Byte, McGraw-Hill, Peterborough, NH January 1997.

[20] First Source International, advertisement, Byte, McGraw-Hill, Peterborough, NH January 1997.

[21] Crothers, B., "Memory Prices Creep Back Up," CNET, Inc., available from: http://www.news.com/News/Item/0,4,8426,00.html Internet: accessed 12 May 1997, quoting Handy from Dataquest, available from: http://www.dataquest.com Internet: accessed 12 May 1997.

[22] Toshiba Corp., "A revolution is coming and it will change everything you think about Home Entertainment," available from http://www.toshiba.com/tacp/SD/javahome.html Internet: accessed 23 March 1997.

[23] NCSA, "A Beginner's Guide to HTML," available from: http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimerAll.html Internet: accessed 23 March 1997.

[24] Luqi, "A Graph Model for Software Evolution," in IEEE Computer Society Press tutorial, Software Merging and Slicing, collected by V. Berzins, May 1995, pp. 202-212.

[25]University of Kansas, "An Instantaneous Introduction to CGI Scripts and HTML Forms," available from: http://www.cc.ukans.edu/info/forms/forms-intro.html Internet: accessed 23 March 1997.

[26] Perl Institute, Online Manual, available from http://www.perl.org/CPAN/doc/manual/html/pod/index.html Internet: accessed 26 May 1997.

[27] ActiveWare, Release Download Page, available from http://www.activeware.com/Download/download.htm Internet: accessed 27 May 1997.

[28] Microsoft Corporation, Access developer web site, available from http://www.microsoft.com/accessdev/docs/bapp97/chapters/ba21_6.htm Internet: accessed 27 May 1997.

[29] Microsoft Corp., Open Database Connectivity, available from http://www.microsoft.com/ODBC/download/DMDownload.htm Internet: accessed 26 May 1997.

[30] Roth, D., Module Description Page, available from http://www.roth.net/odbc/odbc.html#ModDesc Internet: access 28 May 1997.

[31] Badr, S. and Luqi, "Automation Support for Concurrent Software Engineering," Proceeding of the 6th International Conference on Software Engineering and Knowledge Engineering, Jurmala, Latvia, June 1994, pp. 46-53.

[32] Microsoft Corporation, available from: http://www.microsoft.com Internet: accessed 1March 1997.

[33] Leonard, A., "Flare Download Site," available from: http://www.cs.nps.navy.mil/misc/flare/Readme.html Internet: accessed 14 April 1997.

[34] Netscape Corporation, "JavaScript Authoring Guide," available from: http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/index.html Internet: accessed 16 April 1997.

[35] Web Communications, "WWW Fill-Out Forms," available form: http://www.webcom.com/~webcom/html/tutor/forms Internet: accessed 16 April 1997.

[36] Netscape Corporation, available from: http://home.netscape.com Internet: accessed 16 April 1997.

[37] Rational Rose Inc., Unified Modeling Language version 1.0, available from http://www.rational.com/ot/uml/1.0/index.html Internet: accessed 3 April 1997.

[38] Elmasri, R. and Navathe, S. B., Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company, Redwood City, CA 1994.

[39] Cormen, T. H., Leiserson, C. E. and Rivest, R. L., Introduction to Algorithms, MIT Press, Cambridge, 1990.

[40] Russel, D. and Gangemi, G. T. Sr., Computer Security Basics, O'Reilly & Associates, Inc., Sebastopol, CA 1991.

[41] Garfinkel, S. and Spafford, G., Proactical Unix & Internet Security, O'Reilly & Associates, Inc., Sebastopol, CA 1996.