Development of software support tools for social work professionals

Elliot Díaz
Computer Engineering
Othoniel Rodríguez, Ph.D.
Department of Computer Engineering
Polytechnic University of Puerto Rico

Abstract *— The objective of this project was to identify the general needs of social work professionals and use the software engineering process to develop these much needed software support tools. The primary needs identified were storage and fast retrieval of data, inventory management, schedule management, and security. The Waterfall Model of software development was used throughout the project. A detailed requirements and design analysis were performed. Microsoft's VB.NET programming environment and Oracle's MySQL database were chosen for the project. Unified Modeling Language's (UML) use cases, class diagrams, and relational model diagrams were used to aid in the development process. The process was documented following IBM's Rational Unified Process (RUP) documentation style. The source code was fully documented tested. The tests performed included configuration, performance, load, functional and database integrity testing. All the tests were successful in the sense that either no defects were found or the defects found were successfully removed.*

Key Terms *— Social work, software development, software engineering, Waterfall Model.*

INTRODUCTION

These Social work professionals have great need for software support tools. There are virtually no affordable or open-source software solutions to handle their client's cases. All social workers contacted before beginning this project have stated that they rely solely on the Microsoft's Word and Excel products to handle their needs. Additionally, many social worker offices also manage small inventories of items for their "clients", and most social workers use some form of appointment handling system. The problem deepens when you add that most social worker offices have old computers with limited resources (processor speed, memory size, hard disk size, etc). Following the software engineering process a software solution could be developed that could help social workers keep records of their cases, inventory and appointments, in a centralized database, and at the same time be light-weight enough to quickly perform its functionalities on an old computer with limited resources.

A social work center in Dorado, PR was selected to serve as example for this project. The center is called CEDEM (Centro de Desarrollo Humano de la Mujer y la Familia) and is a social work office focused on helping women and families in times of need.

SOFTWARE REQUIREMENTS

The Social Worker Assistance System's (SWAS) main objective is to store all data regarding the participants (the clients who require the center's assistance) that is currently only being kept in paper. The main objective or feature is storing the participant's data. The data would include: profile, background, family, situation, services required, and social worker's analysis. Additionally, it should be capable of quickly searching for a participant's records using a variety of fields, such as name, telephone number, and record number. The system should also be secured by a username/password security subsystem.

Some additional but not critical requirements include keeping a calendar of CEDEM's activities,

and keeping a database of CEDEM's inventory and how it is being managed.

SWAS requires one computer behaving as a server which must have the MySQL database installed and configured. A Linux server would be the recommendation, as MySQL database is native to Linux, but any Windows Server edition or Windows XP Professional or above could also be used. The server should have at least 512 MB memory and at least 1 GB of free hard drive. The client computers running the tool itself must be connected through a LAN connection to the server and would be required to have a Windows operating system with its version being Windows XP or above with the 2.0 .NET framework or above. The clients should have at least 512 MB of memory.

Some constraints that were taken into consideration for the project include the following. CEDEM only has available old computers with limited memory and Windows operating systems, so the software had to be lightweight and compatible with Windows XP. One computer must serve the database for the support tool and the database must be MySQL to keep CEDEM free of database licensing hassles. Most of CEDEM's employees only speak and read Spanish, so the software's text must be fully written in Spanish. The data must be secured, which means the SWAS and its database must be secured by a system of user/password authentication. The software will be delivered with its full code to CEDEM so that they may contact any programmer to further enhance the tool.

As a non-functional requirement, SWAS should never delete entries in the database; instead a system of expiration of entries should be used. Each entry to the database should expire the previous entry and insert a new entry. This could be achieved by creating a compound primary key that includes the date of the change. This would make analyzing changes to the database much easier for the DBA (database administrator), and would make it possible to recuperate data erroneously expired. Since the user-name is to be inserted with each change as specified in section 5.3 (Security Requirements), then on each database entry the DBA could easily view which users performed which changes. This simple requirement makes systems auditing much easier and reliable.

System Features

As mentioned before, the system features required for the project include: Participant Record Management, Security System, Inventory System, and Calendar System.

The Participant Record Management feature was a high priority feature that refers to the ability to store, retrieve, update, and print a participant's record. The functional requirements include:

- Creating a search query and fetch matching data from the database.
- Fetching complete participant data from the database.
- Creating a new participant record in the database.
- Updating participant data in the database.
- Displaying participant data in forms.
- Sanitation of user inputs before storing in the database.
- Alerting user of any errors.
- Creating/appending to log (text) file.
- Writing detailed error data to log file.

The Security System feature was also a high priority feature that refers to the ability to create, update, and remove SWAS users, and verify their username and password before granting access to the system and participant's data. The functional requirements include:

- Verifying the existence of a specific user-name in the database.
- Verifying the existence of a specific user/password combination in the database.
- Verifying that the current password is correct when trying to change it.
- Verifying that the new password and its confirmation are equal.
- Displaying user's data.
- Updating the user's data.

- Displaying a list of users.
- Displaying a list of users containing within their data specified criteria.
- Invalidating a specific user in the database.

The Inventory System feature was a medium priority feature that refers to the ability to create an inventory lot, and add or remove items to or from that lot. The feature would keep record of the amount of items in the lot, to whom they are given and who donates them to CEDEM. The functional requirements include:

- Displaying a list of inventory lots.
- Verifying the size of a particular lot.
- Creating a new lot category.
- Creating a new inventory lot.
- Recording lot transaction data.
- Decreasing the amount of items in a lot.
- Increasing the amount of items in a lot.
- Displaying a list of lots containing within their data specified criteria.

The Calendar System feature was also a medium priority feature that refers to the ability to schedule events at a date and time. The feature would also remind the user when the event is near, and if he/she tries to overlap different events. The functional requirements include the following:

- Displaying a list of all events for a user.
- Displaying a list of the user's current date's events in the main window.
- Displaying a list of events containing specific criteria.
- Automatically refreshing the list of events.
- Alerting the user when an event is imminent.
- Writing to a text file a list of events.
- Inserting an event into the database.
- Invalidating a specific event in the database.

SOFTWARE DESIGN

This section presents the architecture as a series of views; use case view, logical view, process view and deployment view. These views are based on an underlying Unified Modeling Language (UML).

Use-Case View

This section describes the set of use cases that represent some significant, central functionality. It also describes the set of use cases that have a substantial architectural coverage or that stress or illustrate a specific, delicate point of the architecture.

The participant record management system's use case (Figure 1) enables a user to create and edit participant records. At the moment of the record's creation the participant's profile containing his/her primary data must be entered, and optionally the user may enter the participant's family data, background, current situation, and desired services. These optional inputs may also be entered when editing the participant's record. The edit record option would always use the record searching feature of the application.

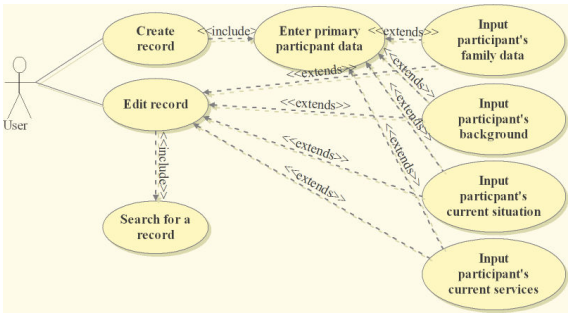

Figure 1
Participant Record Management Use Case

The security system's use case (Figure 2) enables a user to create another user, edit a user's data, which may include changing his/her password, and to delete another user account, which has as a requirement the activity of searching for the user to be removed.

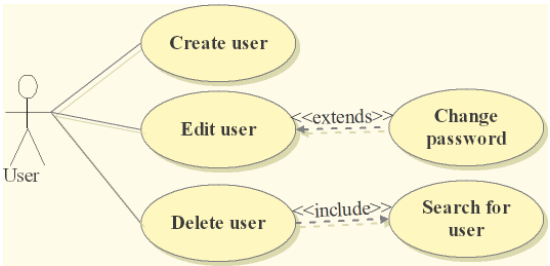

Figure 2
Security System Use Case

The inventory system's use case (Figure 3) enables the user to create an inventory lot, and deposit or withdraw items from an inventory lot. The lot creation function would optionally involve creating a new type or category of inventory (e.g. women's clothes, toddler's toys, etc.). Withdrawing and depositing items to lots would involve using a searching for a lot feature, and would both update the quantity of items in the lot.

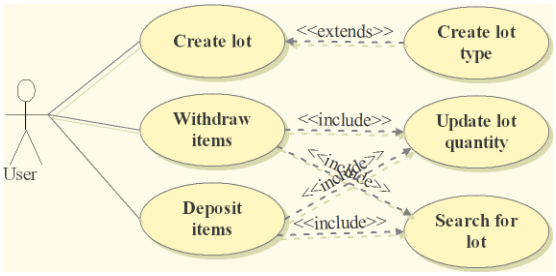

Figure 3
Inventory System Use Case

The calendar system's use case (Figure 4) enables the user to create and cancel events, and to view their current scheduled events. Of course, to cancel an event the searching feature must be used to first identify the event to be canceled.

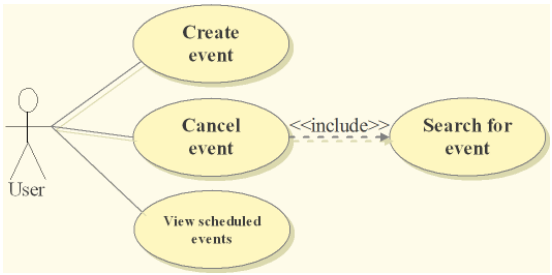

Figure 4
Calendar System Use Case

Logical View

This section presents a description of the logical view of the design. Here are described the most important classes, and their organization in service packages and subsystems.

SWAS's logical view consists of three main packages: the user interface management package, the input and output management package, and the database management package (See Figure 5).

The user interface management package contains all the forms the users will actually view and interact with. These would include the forms for login, user creation, maintenance and deletion, participant record creation, participant background, family data, current situation and needed services data, inventory lot creation, item deposit and withdrawal, event creation and cancellation, and calendar view. This package would also include the simple logical code pertaining to the individual form's functionality.

The input and output management package contains classes with all the functions required to interact with the client's file-system to export data and write the log files.

The database management package contains classes with all the functions required to interact with the database server. This would include opening and closing connections, creating and running update, insert, and select queries in the database and returning the responses in an appropriate format.

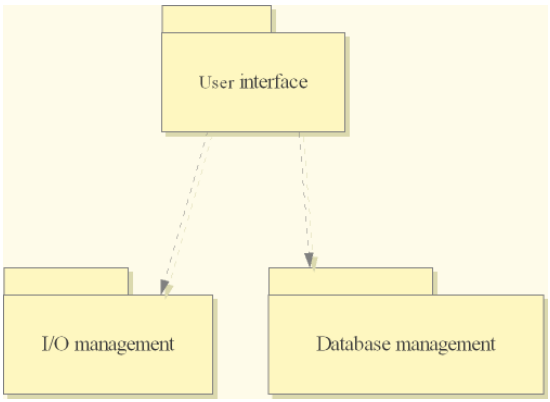

Figure 5
Package Diagram

Process View

This section describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations.

In SWAS's case, the application will be using a simple client/server configuration, in which the server will store the data and the client will perform all of the logical processes. As specified in the Software Requirements, the client is required to be lightweight, and is not expected to execute strenuous operations, so the client would be configured to process its commands on a single

thread, one process at a time. The server however would be configured to process all the database transactions on a multithreaded level, as to allow on the very least 10 clients to process transactions concurrently. The internal processes of the server would be automatically handled by the MySQL database.

Deployment View

This section describes the physical network configuration, including the processes that would run on each node.

In the Deployment diagram depicted in Figure 6, can be viewed the two types of possible connections. The network would be a simple client/server configuration, over an office local area network, or alternatively over the Internet. The interface would initiate a connection to the Database Server through the Database Management Package. This package would then initiate communications with the Database Server through a TCP/IP connection. For the connection to work over the Internet, the Database Server would simply require a Static IP address over the Internet. The internal processes carried out in either an Client PC or External Client PC would be identical.

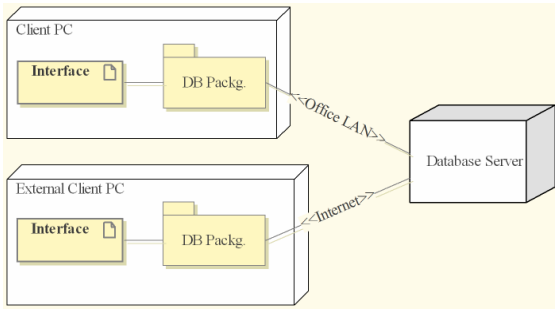

Figure 6
Deployment Diagram

Size and Performance

The MySQL database has been reported to having performed on benchmarks several thousands of transactions per minute, quite more than will be necessary for the operation of SWAS. On the other hand, CEDEM's best candidate for a server computer is a Windows XP Professional machine, which only allows 10 concurrent connections. Seems like a small number of connections, but it shouldn't be a problem since they do not plan on using the software over the Internet and the center has less than 10 other computers available and there are only a few employees working on the center at any one time. The software is designed in a minimalist fashion, only performing the functions stipulated by the Software Requirements Specifications. It does not have any additional graphical requirements, and does not perform multiple transactions concurrently on any single client. Overall SWAS would perform far beyond the needs specified in the Software Requirements.

Quality

The design conforms to the quality requirements stipulated in the Software Requirements in the following manner:

- The software's text is to be written in Spanish. English was used in this text to be uniform in the written communication.
- The files to be written by the Output Layer through the Input/Output Management Package are to be comma delimited text files. On machines with Microsoft Office installed, these files are configured to be opened by Microsoft Excel. On any machine these files can be edited by Notepad, WordPad or any other text editor.
- The Operational Layer applies a flexible configuration of the data entered to the User Interface Management Package. The only data required for a participant record to be opened is the main participant's profile data, and only a small set of the participant's profile data is required. The software does not require any additional participant's data to be entered in any particular order.

TEST PLAN

The test plan supports as its main objectives identifying existing project information and the software components that should be tested, listing the recommended Requirements for Test (high

level), recommending and describing the testing strategies to be employed, identifying the required resources and provide an estimate of the test efforts, and finally listing the deliverable elements. The following functional requirements were identified as targets for testing: database test, functional test, user interface test, load test, security and access control test, and configuration test.

Database testing serves to verify that SWAS's user credentials can be entered and retrieved, that the participant's data can be entered, retrieved and updated, that the categories can be entered and displayed, that the inventory lots can be entered, and updated, that the schedule can be entered, and displayed, and that changes to the participant's data can be audited.

Functional testing serves to verify that users can create a participant profile and locate it afterwards, that users can export the participant's data to files that can be opened by text editors such as Microsoft's Word or Excel, that users can create an inventory lot and locate it afterwards to perform deposits and withdrawals, and that users can create scheduled events, and the events would be displayed in a manner that alerts the user.

User interface testing serves to verify proper navigation through all use cases, and that each UI panel can be easily understood.

Load testing serves to verify the response time when loaded with 10 logged users, and when the 10 users access the database simultaneously.

Security and access control testing serves to verify that non-authorized users cannot access the system's data, that users can be created and removed from the system, that users can change their passwords, and that a list of current user's can be accessed.

Configuration testing serves to verify proper functional operation using the Operating Systems specified in the Software Requirements: Windows XP SP3 or Windows Vista SP2 32 and 64-bit.

Test Strategy

The test strategy presents a recommended approach for testing the software application. Previously was described what would be tested; now is described how it will be tested.

The objective of the data and database integrity testing was to ensure the database access methods and processes function properly without data corruption. The test involved manually invoking each database access method and process, seeding each with valid and invalid data (or requests for data). It would also involve inspecting the database to ensure the data has been populated as intended, and reviewing the returned data to ensure that the correct data was received. The test would be completed when all database access methods and processes functioned as designed without any data corruption.

The objective of function testing was to ensure proper application navigation, data entry, processing and retrieval. The test involved executing each function using valid and invalid data to verify the expected results with valid data, an the appropriate error with invalid data. The test would be completed when all functions have been executed and all the identified defects were addressed.

The objective user interface testing was be to verify that navigation through the target of the test properly reflects the requirements, including window to window, field to field and access methods (tab key and mouse movements). Another objective was to verify that object characteristics such as menus, size, position, state, and focus conform to standards. The test involved navigating and verifying the objects in each application window. The test would be completed when each window was properly verified and within acceptable standards.

The objective of load testing was to verify the performance behavior of designated functionalities under normal anticipated workload and worse anticipated workload. The test would be performed by simulating execution of developed functions increasing the number of transactions or the number of iterations of each transaction. It would be performed on one and on multiple client machines. The test would be completed when successful

completion of the functions is attained without any failures and within expected time.

The objective of the security and access control testing would be to verify that only users with permission to access the application are permitted to access them. The test would be performed by attempting to gain access to the application with permitted users and non-permitted users. The test would be completed when only the users permitted to access the application can access it.

The objective of the configuration testing was to verify that the client application functions correctly on the prescribed client PCs. The test would be performed by performing several functions that interact with the PC's applications. The test would be completed when the functions combining the application and the PC's applications are completed without failure.

TEST RESULTS

The tests defined in SWAS's Test Plan where performed according to the strategy indicated. All the tests were successfully completed in terms of the functionalities and test requirements defined. In most cases, no defects were found, and in the cases where defects were found, they were promptly corrected.

The data and database integrity tests were performed to verify the system's security, the participant's data, the categories, the inventory, and the scheduling data integrity. The tests involved manually invoking, inserting and updating the data including acceptable and unacceptable values. All the tests performed were successful in the sense that all unacceptable data triggered appropriate error messages, protecting the integrity of the data; whilst acceptable data was successfully inserted to the database.

The functional tests were performed to verify the system's functionalities in the areas of creating and locating a participant's profile, exporting a participant's profile to an external test file, opening the text file with Word or Excel, creating and finding inventory lots, depositing and withdrawing items from an inventory lot, and scheduling events. The tests involved executing each function with valid and invalid data to ensure that the appropriate actions or error messages where activated by the software. All the tests performed were successful in the sense that all unacceptable data triggered appropriate error messages; whilst acceptable data was successfully inputted to the system. All the functionalities performed as expected.

The user interface tests were performed to verify that the system's user interface panels could be easily understood. The tests involved navigating through each user interface panel verifying that they properly reflected the requirements of the particular use case, and that all the elements within the panels conformed to a set of standards. The tests performed were successful in the sense that the user interface panels reflected properly the functionalities they addressed. The few defects found were related to the aesthetics of the panels and were corrected in order to maintain a standard look and feel throughout the application.

The load tests were performed to verify that the system's performance under normal and worse possible anticipated workloads. The tests involved opening at least 10 instances of the application in both host and client computers and performing various transactional functions that access the database. The objective was to verify that the response time was acceptable under normal and worse possible anticipated workloads. The tests performed were successful in the sense that, as expected, the response time was normally under one second for each transaction performed. The test was performed over a wireless network with a maximum 54Mb/s speed, which is in fact less than the recommended LAN connection for the software. For the purpose of this project and this test the worse possible workload was just 10 concurrent users, but further tests are recommended with at least 50 concurrent instances over a 100Mb/s LAN connection.

The security and access control tests were performed to verify that non-authorized users could not access the system, that users could be created

and eliminated from the system, that users could change their passwords, and that a list of all the current users could be accessed. The tests involved creating and deleting several users, attempting to access the system with and without proper credentials, changing the passwords of several users and confirming the new password by exiting and re-entering the system, and producing and exporting a list of users to an external format. The tests performed were successful in the sense that users were successfully created and removed from the system, their data and passwords was successfully updated and validated upon exiting and re-entering the system, a list of current users was created and exported to a comma-delimited file, and finally upon attempting to access the system with and without the proper credentials, access was granted only with the proper credentials.

The configuration tests were performed to verify that the client could be installed and operated on Windows XP, Windows Vista 32-bit and Windows Vista 64-bit. The tests involved installing the client software on several clients with the indicated operating systems, and performing all the application's functions. The tests performed were successful in the sense that the client was installed and configured to run on several machines with the specified operating systems, and all the functions were verified to be running appropriately.

CONCLUSION

The problem specified at the beginning of this article was to be solved by using the software engineering process to develop a solution capable of storing, updating and retrieving social work cases, inventory data and appointment data in a centralized database, and at the same time being capable of performing these functionalities on a computer with limited resources (processor speed, memory size, hard drive size, etc). Throughout this article is demonstrated the software engineering process being carried out for this purpose. The software's requirements and a design capable of fulfilling the requirements are here summarized. The design's development can be seen in the source code, and in the database's schema, which were not included for practical purposes. A test plan was developed that would verify that the software could, in fact, fulfill all of the requirements. Finally, all the tests performed on the software were successful in either proving that the software fulfilled all the requirements or brought to the surface any deficiencies which were immediately resolved. The only possible conclusion is that the problem as it was defined has been completely and successfully resolved.

REFERENCES

[1] Connolly, T. and Begg, C., *Database Systems: A Practical Approach to Design, Implementation, and Management*, Third Edition, Addison Wesley, 2001.

[2] Horstmann, C., *Object-Oriented Design and Patterns*, Second Edition, Wiley, 2005.

[3] Miller, R., "Practical UML: A Hands-On Introduction for Developers", http://edn.embarcadero.com/article/31863, 2009.

[4] MySQL, "MySQL 5.1 Reference Manual", http://dev.mysql.com/doc/refman/5.1/en/index.html, 2009.

[5] Pratt, P. and Last, M., *A Guide to SQL*, Eighth Edition, Course Technology, 2008.

[6] Rational, "Rational Unified Process: Best Practices for Software Development Teams", *Rational Software White Paper*, Rev 11/01.

[7] Rational Software Corporation, http://www.ts.mah.se/RUP/RationalUnifiedProcess/wordtmpl/index.htm, 2009.

[8] Shneiderman, B. and Plaisant, C., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Fifth Edition, Addison Wesley, 2009.