The Automated Testing Tool

By: Jeff Williams, Rebekah Michael

A Draft Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment for
the Bachelor's Degree
of Information Engineering Technology

University of Cincinnati
College of Applied Science

June 1, 2006

The Automated Testing Tool

By: Jeff Williams, Rebekah Michael

A Draft Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment for
the Bachelor's Degree
of Information Engineering Technology

University of Cincinnati
College of Applied Science

June 1, 2006

Rebekah Michael	Date
Jeff Williams	Date
Professor McMahon	Date
Patrick C. Kumpf, Ed.D. Interim Department Head	Date

Table of Contents

List of Figures

Abstract

CS/CAPP is a software product developed by CIMx, Inc., a small company in Milford, Oh. CS/CAPP is an Oracle Forms 6i application running on an Oracle 9i application database that customers use to plan out the materials and construction processes of anything from aircraft engines to military helicopters. Because of the high profile customers like Boeing and Rocketdyne and the sensitive and mission critical use of this software, it is very important for CIMx to rigorously test the software so that it is of the highest quality.

Testing includes regression testing older functionality to ensure that it has not been broken by any new functionality or updates to existing features that may be introduced in each patch. These static regression tests take approximately 4 weeks of 3 testers' time before each release. Our project team decided that an automated testing tool could execute these regression tests, so we designed the 'CS/CAPP Automated Testing Tool'. This tool is capable of automatically covering all core competency regression tests for CS/CAPP, grouped by functionality. Through the Automated Testing Tool (ATT) the Quality Assurance department chooses a single test or group of tests to run. The ATT then performs the test(s) and presents the results to the user upon completion of testing. It also records the results in an Oracle 9i database, providing an archive to view or print reports on demand.

The CS/CAPP Automated Testing Tool will save CIMx approximately 12 weeks of testing time every year and will ensure a consistently high quality product.

Automated Testing Tool Problem Statement

CIMx is a small company based in Milford, Ohio that creates software for industries to use in constructing aircraft engines, military helicopters, and even components of the space shuttle. Because of the nature of the products designed with this software, CIMx's products must be rigorously tested and incredibly dependable. Lives are literally at stake.

Testing the software occurs at the end of each phase of software's iterative development process. This testing takes a great deal of time and effort, and must be done with every product update and for every new product. In addition to the time and effort involved, there are mistakes or human error on the part of the testers that might let problems with the software slip through.

In short, CIMx needs a faster, more uniform, dependable way to quality test. Our project team believes we can address these needs by creating a program that automatically tests the core functionality of the CIMx products. This software, in the process of testing, is required to perform all the duties associated with QA. These duties include accuracy, and recording or at least reporting bugs found during the automated testing.

As defined in our interview with Kristin McClane of the CIMx company[6], there are four basic requirements for our product.

Finding Errors – First of all, the primary focus of the product will be to discover errors in the software it is designed to test. This will primarily consist of establishing a baseline of data found after a particular test, and comparing it to data found in future product revisions for that same test. This error discovery must perform at or above the levels of current testing practices.

Testing Speed – The product must perform its tests quickly, ideally faster than the testing is currently performed, and above all else, it must perform the tests to conform to the iteration schedule of test cycles. For the purposes of fulfilling this requirement, the evaluation of the execution time of the tests will include set up time by the employee using the Automated Testing Tool, the actual test execution time of the tool, and the time taken by employees to process the results of testing.

Error Reporting – When errors are found, the product must report these errors to CIMx's error reporting system, or must at least notify the user that errors were found and where those errors occurred.

Usability – The product must be usable, with appropriate training, by non-technical employees that are typical of the Quality Assurance department at CIMx.

Solution Description

Our product, The Automated Testing Tool, satisfies the requirements noted above. It is preconfigured with all core competency tests currently used at CIMx, and is capable of running these tests individually or in sets at the command of a CIMx employee. In addition, after running each test, the Automated Testing Tool records and notifies the user of any errors found and in which test case they were found. Our product also contains functionality to create and delete test cases in the system, and the ability to arrange these test cases into groups which can be created, edited, and deleted by the user.

Executing Tests

Tests are initiated by the test lead employee by clicking on Execute in the GUI section of the Automated Testing Tool application. Execute Test executes a selected macro or set of macros, created in Macro Express, which interacts with the CAPP System to obtain test results. The test will pass or fail, and the Execution logic will then call functions to record the test results. These results consist of a pass/fail value which is associated with the test’s execution iteration and the test’s ID (to distinguish it from other instances of that test being run). In addition to this record in the database, the user is notified at the end of automated testing which tests passed and failed in that iteration of test execution.

Generating Reports

Generating a report is an option of the QA manager. The Generate Report function recalls or prints the results of tests and test groups and entire test iterations in easily readable format.

The Test Case Manager

The Test Case Manager is a discrete portion of the user interface users can access to add, edit, delete, and group test cases, and to add, edit, and delete test case groups. Once a user enters this area, that user will be able to perform the following six functions.

Adding New Tests

Adding a new test is done by the test lead in the Test Case Manager through MacroExpress. MacroExpress will loosely tie into CS/CAPP and the Automated Testing Tool to record relevant database information for test execution data verification. All tests added will initially be unattached to any test group.

Once a user is in the Test Case Manager section, the basic flow of adding a test is this:
The user manually opens and performs the test in MacroExpress. As this test is performed, triggers in CAPP’s database record any database information in a baseline file. After finishing the test, the user finds the file this record is stored in, along with the macro file created by MacroExpress. Once these files are found, users can use the find file buttons on the Add Test panel to attach each of them to the new test. The user can then name the test, and attach it to any test group if they wish. After all these

steps are completed, the user can click the Add Test button, and the test will be recorded in the database. The user can then exit the Test Case Manager.

Edit Tests

Editing a test is accomplished through the Test Case Manager. Users select the option to edit test cases, and they select the test case to be edited. All values of a test case will be editable, excepting its unique database ID.

Deleting Tests

Deleting a test is also be done through the Test Case Manager. Users access the delete function by simply selecting the option to delete test cases. The user will then select the test to be deleted. And after final confirmation with the user, the test case is deleted from the system.

Adding Test Groups

Adding a test group is accessed through the Test Case Manager. Selecting the add test group option gives users the opportunity to enter the test case group’s name and confirm it. This places the test case group in the database and can now be edited in the Editing Test Groups area.

Editing Test Groups

Users seeking to edit test groups can access the function through the Test Case Manager. Users editing test groups are confronted with the option to change the group’s name, or to add or remove test cases to the test group. Once a test case is attached to a test group, it cannot be associated with any other test groups;   Test cases → Test groups are a one-to-one relationship in our product.

Deleting Test Groups

Deletion of test groups is also a part of the Test Case Manager. Users accessing the test group deletion function are presented with a list of current test case groups. Choosing a group from this list and deleting it will give a confirmation dialogue to the user. If the user accepts, the test group is deleted. All test cases currently attached to that test group will become unattached but remain in the system.

User Profiles

The Automated Testing Tool solution must recognize two user profiles and allow those user profiles to utilize functionality appropriate to them. These user profiles are:

> *Quality Assurance Manager* – The Q.A. Manager is only able to view, generate, and print reports containing the success or failure of test iterations or individual test cases.
>
> *Test Lead* – The Test Lead user is able to add, delete, and edit test cases. The Test Lead is also able to handle GUI mapping to create new tests or modify old ones, and the Lead Tester is able to execute tests individually or in group iterations. The Lead Tester can also view the results of tests or groups of tests.

Design Protocols

Figure 1, shown below, is a basic diagram of the functionality of the system. Users enter the application into a login screen, then receive a menu of options, and then proceed to the three basic units of functionality in the Automated Testing Tool.

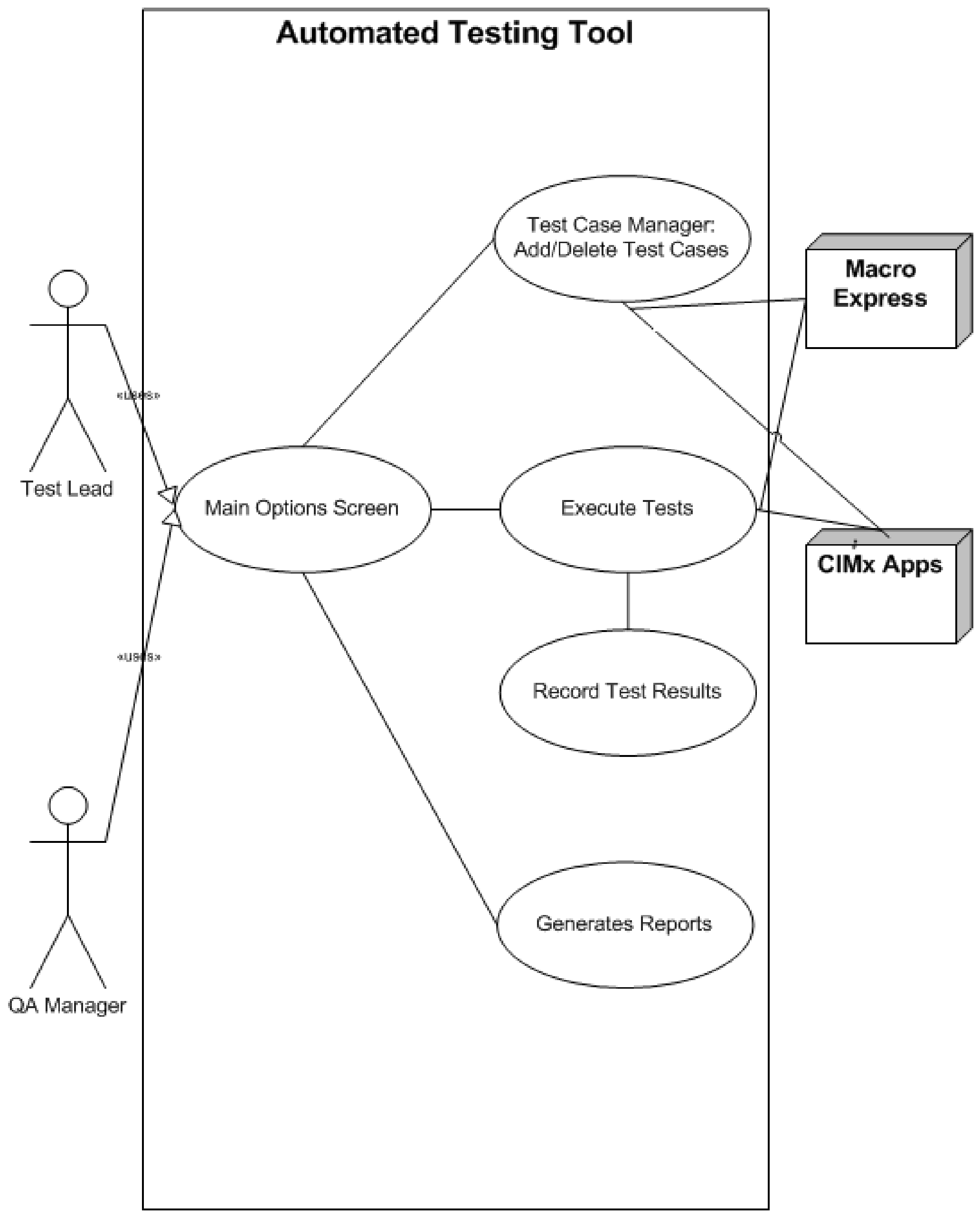


Figure 1

Database Diagram

The following diagram shows the database tables that the Automated Testing Tool will directly use. The TEST_CASE table holds data related to test cases, the TEST_ROUND table holds the groups. The TEST_RESULT table holds the results of the tests performed by the Testing Tool. The TEST_ITERATION table holds information, such as iteration description, related to each testing iteration. And finally the TEST_LOGIN table holds users' passwords, login names, and user type.

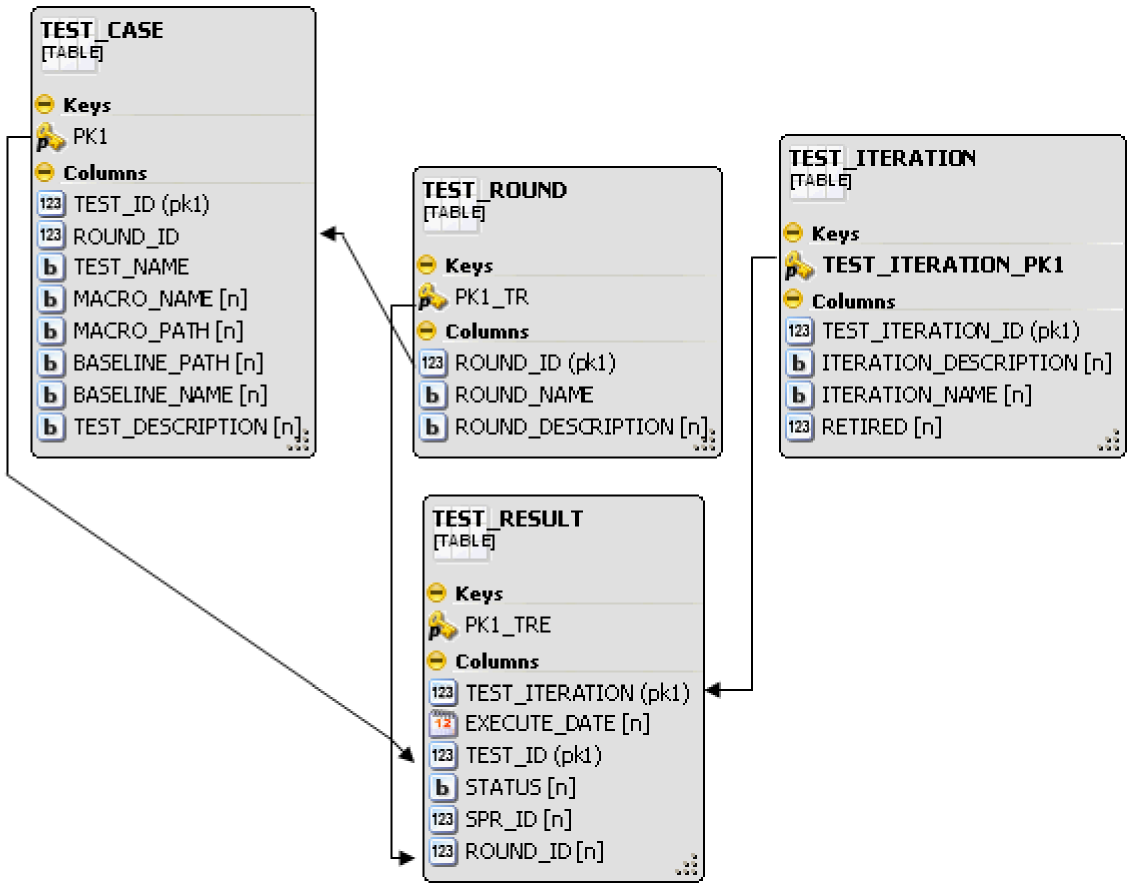


Figure 2

Use Cases

The following tables are some of the use cases implemented in the final product.

ID	ATT-001	
Name	**Adding Test Case**	
Priority	High	
Description	Test Lead must be able to add a new test case to the system.	
Business Value	Automated Testing Tool is expandable for new functionality	
User(s)	Primary: Test Lead	
Pre-Condition(s)	Test Lead has valid userid and password for the system	
Post-Condition(s)	Test Case is added to system and retrievable to execute.	
Typical Path	**Action**	**Response**
	1. User presses Manage Test Cases button.	2. Test Case Manager form is displayed.
	3. User selects Add button.	4. Right side of form displays add information, no data autofilled.
	5. User enters Test Case Name in text box.	6. Text box accepts user entered text.
	7. User clicks Find New Macro button.	8. Windows dialog is displayed.
	9. User chooses Macro file and clicks ok.	10. Macro path and file name is displayed in form.
	11. User clicks Find New Baseline button.	12. Windows dialog is displayed.
	13. User chooses Baseline text file and clicks ok.	14. Baseline path and file name is displayed in form.
	15. User clicks Add Test Case Button.	16. Form displays message, 'Test Case [name] has been added.
Alternates and Exceptions	9. Invalid file format for macro. Form displays message, path and file name not shown. User must choose valid file type.	
	13. Invalid file format for baseline. Form displays message, path and file name not shown. User must choose valid file type.	
Business Rules	1. System allows unlimited addition of test cases. 2. System forces correct file type for macro and baseline file. 3. Test Case Name must be unique.	
Issues	1. No format of test case name is forced. Special characters that may cause problems are not prohibited.	

Editing and Deleting Test Case User Stories in Appendix A

ID	ATT-002	
Name	**Adding Test Group**	
Priority	High	
Description	Test Lead must be able to add a new test group to the system.	
Business Value	Automated Testing Tool is expandable for new functionality	
User(s)	Primary: Test Lead	
Pre-Condition(s)	Test Lead has valid userid and password for the system.	
Post-Condition(s)	Test Group is added to system and retrievable to execute.	
Typical Path	**Action**	**Response**
	1. User presses Manage Test Groups button.	2. Test Group Manager form is displayed.

	3. User selects Add button.	4. Right side of form displays add information, no data autofilled.
	15. User clicks Add Test Group Button.	16. Form displays message, 'Test Group [name] has been added.
Business Rules	1. System allows unlimited addition of test groups. 2. Test Group Name must be unique.	
Issues	1. No format of test group name is forced. Special characters that may cause problems are not prohibited.	

Editing and Deleting Test Group User Stories in Appendix A

ID	ATT-003	
Name	**Adding Test Cases to Group**	
Priority	High	
Description	Test Lead must be able to add a test case to a test group in the system.	
Business Value	Automated Testing Tool is expandable for new functionality	
User(s)	Primary: Test Lead	
Pre-Condition(s)	Test Lead has valid userid and password for the system	
Post-Condition(s)	Test Case is added to Test Group and retrievable to execute from within the group.	
Typical Path	**Action**	**Response**
	1. User presses Manage Test Groups button.	2. Test Group Manager form is displayed.
	3. User selects Edit Group button	4. Right side of form displays drop down list of existing test groups.
	5. User selects test group from drop down to edit.	6. Test group is displayed in drop down list. Test Cases currently in group are listed in right side text box. Available test cases to add to group are displayed in left side text box.
	7. User clicks a test case in left side text box. User clicks add text case arrow.	8. Test case is added to right side text box. Test case is removed from available test cases text box. Test case is now a member of group.
Business Rules	1. System allows unlimited addition of test cases to group.	
Issues	1. When test case is added to test group, test case cannot be executed individually.	

Deleting Test Case from Test Group in Appendix A

ID	ATT-004	
Name	**Executing Test Group**	
Priority	High	
Description	Test Lead must be able to execute a test group in the system.	
Business Value	Automated Testing Tool can execute test groups	
User(s)	Primary: Test Lead	
Pre-Condition(s)	Test Lead has valid userid and password for the system. Test group exists in system with more than 1 test case in the group.	
Post-Condition(s)	Results of test cases within test group are shown after test group runs.	
Typical Path	**Action**	**Response**
	1. User clicks Execute Tests button	2. Execute test form is displayed.

	3. User selects group to be run from drop down menu.	4. Test group is selected in drop down. Test cases assigned to group are displayed in below text box.
	5. User clicks run selected group button.	6. Form goes to the background. Test cases are run in sequence as added in group. Systems database triggers fire data validation procedures as needed for test case. Also fires text entered to text file as needed. After completion of each test case, text file is compared against baseline text file and results of procedures and text file comparison are stored. At the end of the test group, message is displayed indicating results of each individual test case, pass or fail.
	7. User clicks ok on results message.	8. User is returned to Execute test form.
Business Rules	1. System allows all test cases to be run in sequence in the group. 2. Message is displayed to indicate to the user the tests are completed and the results of each.	
Issues	1. On fail, no message is displayed to indicate what failed. 2. If CS/CAPP system fails in a way that is not handled, macro will continue to run and unexpected results will occur.	

Project Deliverables

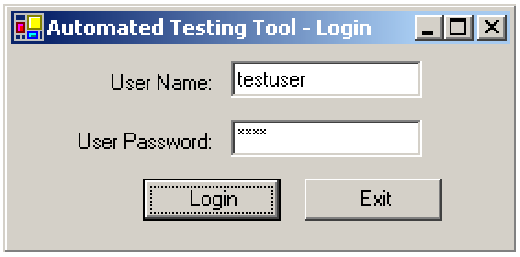
To facilitate the creation of a schedule and unit case layout for the Automated Testing Tool, the following deliverables have been set by the project team.

1. An interface developed from C#, using the .NET framework, to allow users to access the various features of the product, with accompanying help files.
2. This interface must include a user login that will authenticate users against database information and allow the users to view options in the user interface appropriate to their user type.
3. The interface must include an option to execute tests either alone or in groups.
4. The interface must allow a user to view and set up groups of automated tests.
5. The interface must allow a user to add, edit, or delete tests' recorded information.

6. A set of tests created using Macro Express encompassing the Core Competency test set for the CIMx product. These tests must be able to interact in some way with the user interface portion of the program to allow users to execute, view, and organize them freely.
7. On execution of tests, the Automated Testing Tool must record failure or success accurately in the database.
8. A backend database that uses Oracle 9i to store user information and testing results.
9. The Automated Testing Tool must interact with the CIMx product's associated database (which also uses Oracle 9i) to determine whether test steps passed or failed.
10. The interface must be capable of allowing users to generate, view, and print reports on the failure/success information of tests or groups of tests.

User Interface

The following are actual screenshots of the interface of the working product. This section explains the function of each form and how they relate to each other.



User Name: testuser

User Password: ****

Figure 3

User Login – This form is the initial form any user using the Automated Testing Tool will see. The form will take a user name and password for the logging user; compare them to information on the database; and then call the main menu form and pass it a value corresponding to the user's profile.

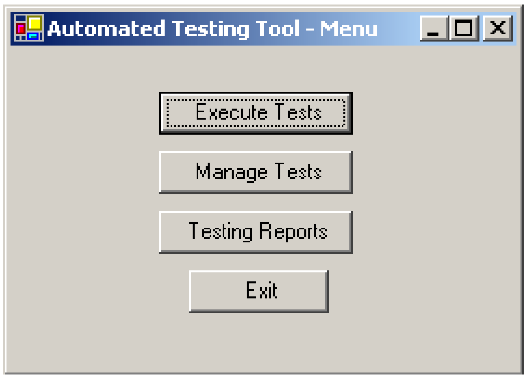


Figure 4

Main Menu – This form displays all the available functions for the user. The value passed by the login form tells the main menu form what options to display. Test Lead users have access to all the functionality, including execute and manage tests, and testing report generation. When a user clicks any of these buttons, a form appropriate to each option will be called (each are discussed below).

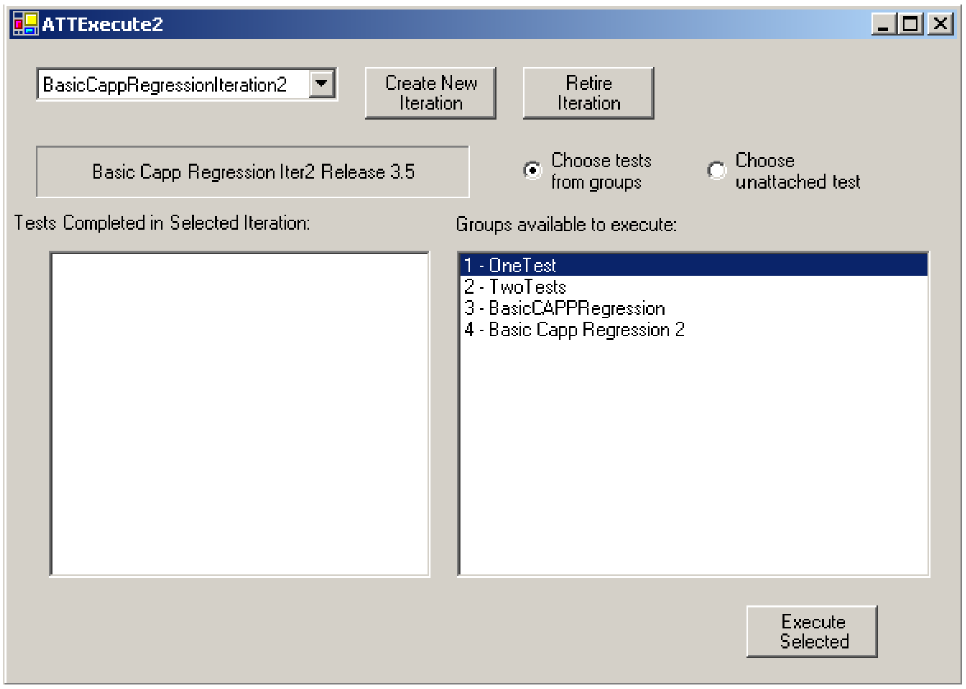

ATTExecute2
BasicCappRegressionIteration2
Create New Iteration
Retire Iteration
Basic Capp Regression Iter2 Release 3.5
Choose tests from groups
Choose unattached test
Tests Completed in Selected Iteration:
Groups available to execute:
1 - OneTest
2 - TwoTests
3 - BasicCAPPRegression
4 - Basic Capp Regression 2
Execute Selected

Figure 5

Execute Tests – The execute test form allows users to select various test cases and test groups to be performed. Once the user has this list, they can choose to execute selected tests from the list. Only a single group or unattached test can be selected for execution at a time. Due to limitations of the Automated Testing Tool, CAPP, and MacroExpress, the execute form is designed to allow users to only execute a test case, or a test group, but not both at a time. This is to avoid confusing the user generated macros by limiting the information and objects inserted into CAPP to certain grouped scenarios. E.g., tests Make Plan, Make Operation, Make Step may work fine when grouped together and run in sequence, but testing the approvals system thoroughly will insert so many objects into CAPP that if they are run out of order or with additional tests in between, the macros may no longer work.

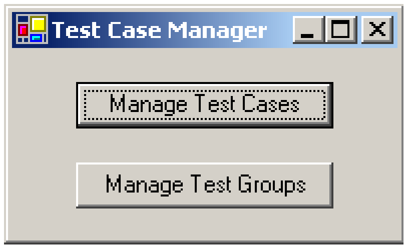
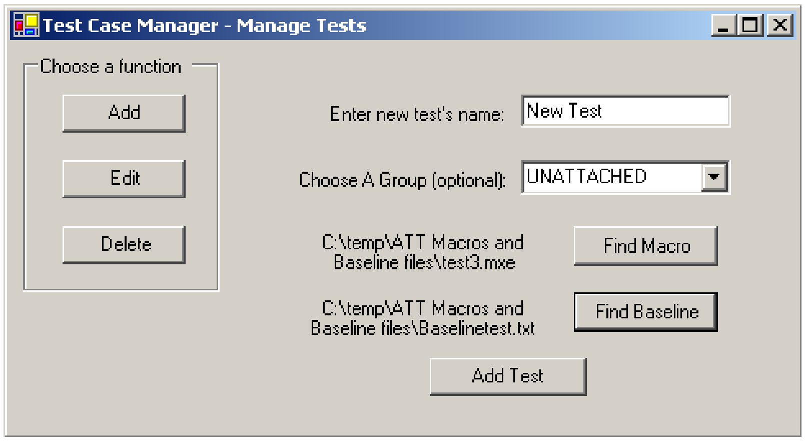


Figure 6

Test Case Manager – If a user selects to manage test cases in the Menu form, the user will receive this Test Case Manager form. The form is simple, and gives the user the option to manage either test cases or test groups.



Enter new test's name: New Test

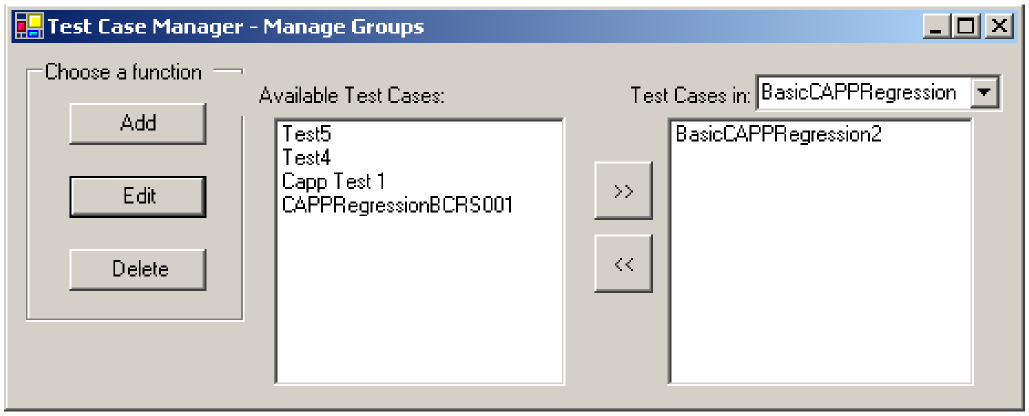
Choose A Group (optional): UNATTACHED

C:\temp\ATT Macros and Baseline files\test3.mxe

C:\temp\ATT Macros and Baseline files\Baselinetest.txt

Figure 7

Manage Test Cases – On selection of Manage Test Cases, the above form appears. Initially the right side of the form is blank and populated with fields, buttons, and options only when the user clicks the Add, Edit, and Delete buttons on the left. Once the user selects one of these options, the right side of the form displays the appropriate objects. Users can then use the newly displayed buttons, fields, and options on the right side to accomplish the function selected.



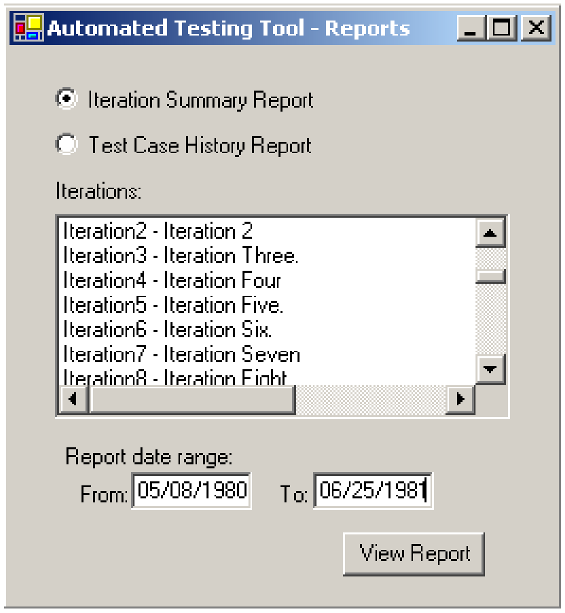
Test5
Test4
Capp Test 1
CAPPRegressionBCRS001

Test Cases in: BasicCAPPRegression

BasicCAPPRegression2

Figure 8

Manage Test Groups – This form is much like the Manage Test Cases form; the right side is initially blank, and the left side gives the user an option to Add, Edit, or Delete test groups. Objects and information appear according to function selected.



(●) Iteration Summary Report
( ) Test Case History Report

Iteration2 - Iteration 2
Iteration3 - Iteration Three.
Iteration4 - Iteration Four
Iteration5 - Iteration Five.
Iteration6 - Iteration Six.
Iteration7 - Iteration Seven
Iteration8 - Iteration Eight

From: 05/08/1980 To: 06/25/1981

Figure 9

Testing Reports – The testing reports form allows users to view and print summaries of entire iterations and reports on the history of individual test cases. The form presents a list of the iterations or test cases, the user then selects the iteration or test case they are interested in, enters a date range to restrict the results set (optional), and clicks the View Report button. This will bring up a report containing information about the selected iteration or the history of the selected test case. Users have the option from here to simply view and close the report, or print it out on paper.

Help Options – Mouse over tool tips will be shown whenever a user hovers the mouse over an option for a certain amount of time.

In addition, a directional read-me file is included with the product which details how to create, maintain, and import their own macros into the Automated Testing Tool so that new users can become acclimated to the product with minimal human intervention. This read-me file also explains the basic functionality and use of the product, and will include a version history in any future product revisions.

Proof of Design

The screenshots above were taken from a working prototype which was displayed in our Senior Design 2 oral presentation. Every function related to test case and group management is entirely functional. Also, the prototype is capable of executing tests either individually or in groups and reporting back accurate information.

The significant design challenges represented by creating a macro, having that macro run specific tasks in the CIMx CAPP product, and having a way to verify the data from that test run have been overcome. The prototype application can do all these things, and report to the user the pass and fail state of the test.

With all the above, the only remaining deliverables left to implement are as follows

- Out of product help resources, including a help menu option for general information, mouse over for help with specific functionality, and an external read me for self training, basic information, and version history
- The reports generation functionality

- A stable of macros covering the Core Competency test cases

While each of these points is a good chunk of work, all the *technical* challenges of this product have been overcome. Therefore, this working prototype represents a valid proof of concept for the Automated Testing Tool, that such a product can be developed.

Testing

We completed unit testing for the final product. We tested each piece individually upon completion and each user story at the end of completion of a user story. Due to the large scope of the database triggers and procedures, a large portion of the testing was executed to find any database errors. Below is an example of one of our test cases. We have a test case for each user story in the prototype. We record each run of a test case based on a user story with pass or fail. All other test cases follow the below format, but using a different use case.

<Execute Add Process Plan Test Case>

<ATT-CS/CAPP Core Comp 001>
Add function: Add process plan

EXECUTION STATUS

Executed By	**RMM**	**Run – Pass / Fail**	**P**
Build Date	**02/21/06**	**Execution Date**	**02/21/06**
Database Name	**RM1**	**Database Version**	**9.2.0.5.0**

TEST CASE STEPS:

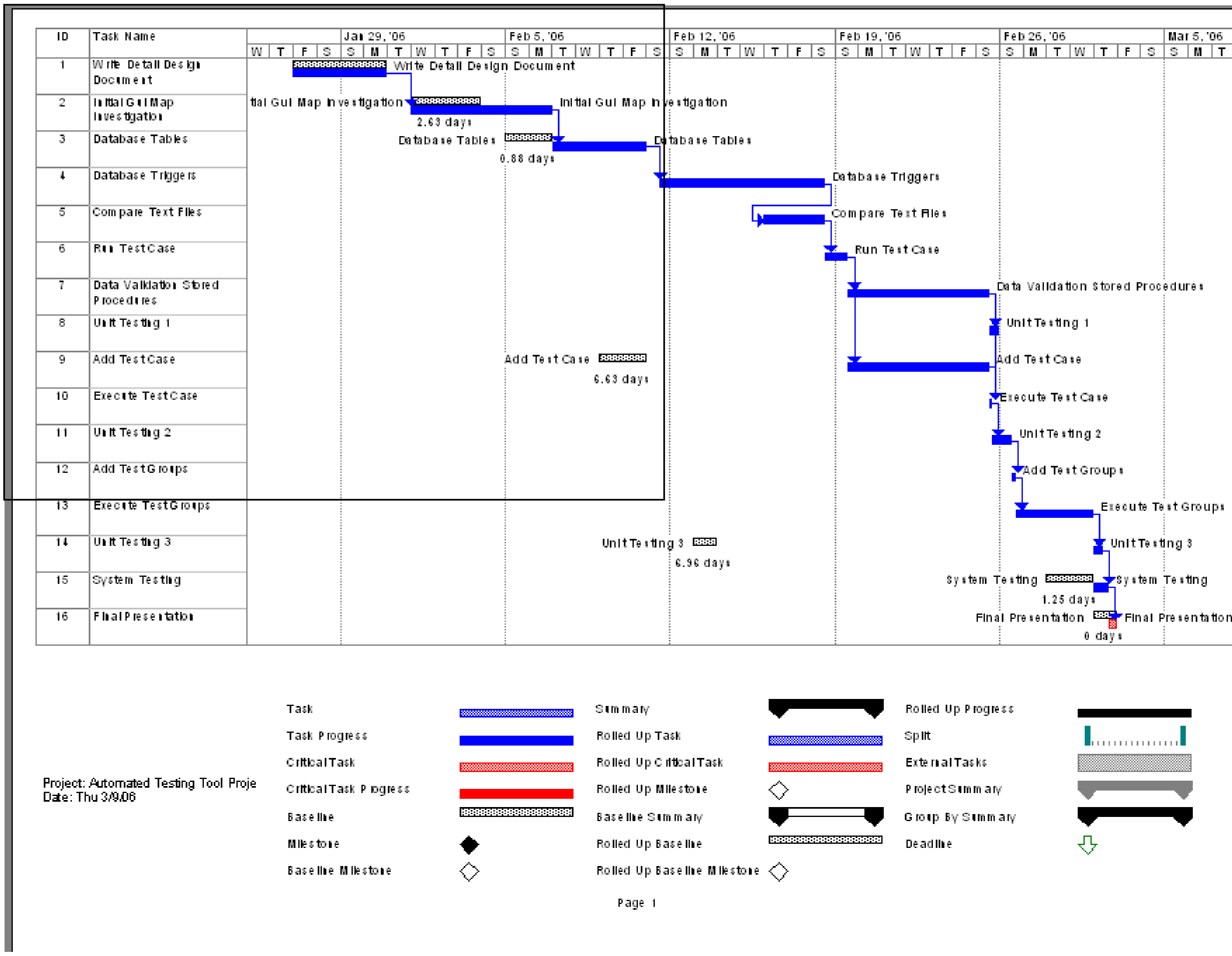
STEP	ACTION	EXPECTED RESULTS	ACTUAL RESULT (P/F)	PROBLEM REPORT NUMBER
1.	Log onto Automated Test Tool with correct userid and password	Log on is successful		
2.	Path: Execute Tests – choose group Add Process Plan	Group Add Process Plan is displayed with test case Add Process Plan contained within.		
3.	Click Execute Group	The form is moved to the background as the capp system is opened by the macro. The test case executes as expected resulting in a new process plan in the capp system. The database triggers fired on insert in process_plan table to execute Insert_Plan stored procedure. This procedure checks that all required fields are entered, and an entry in Part table is made with the same plannbr as process_plan record. All data entered into capp forms is put into text file. Program exits capp. Program compares baseline file of test case to actual text file just created on execute. Text files are identical. On exit, pop box is displayed indicating success of test case Add Process Plan.		
4.	Click ok on results pop up box.	Pop up box is closed. User is returned to execute test group form of automated test tool.		

DOCUMENT HISTORY: Document version begins with 1.0. Complete all fields. Update each time a change is made.

DOCUMENT VERSION	DATE OF CHANGE	REASON FOR CHANGE	INITIAL
1.0	02/21/06	Initial version	*RMM*

Schedule

The development team keeps track of progress using Microsoft Project. We saved a baseline at the beginning of the project. As time progressed, due to problems in implementation, we changed this schedule. Most of the items in the baseline are not repeated in the actual timeline so they don't show up at all. The few that do show up are earlier for completion than the actual dates. Everything on the final schedule was fully completed.



ID	Task Name
1	Write Detail Design Document
2	Initial GUI Map Investigation
3	Database Tables
4	Database Triggers
5	Compare Text Files
6	Run Test Case
7	Data Validation Stored Procedures
8	Unit Testing 1
9	Add Test Case
10	Execute Test Case
11	Unit Testing 2
12	Add Test Groups
13	Execute Test Groups
14	Unit Testing 3
15	System Testing
16	Final Presentation

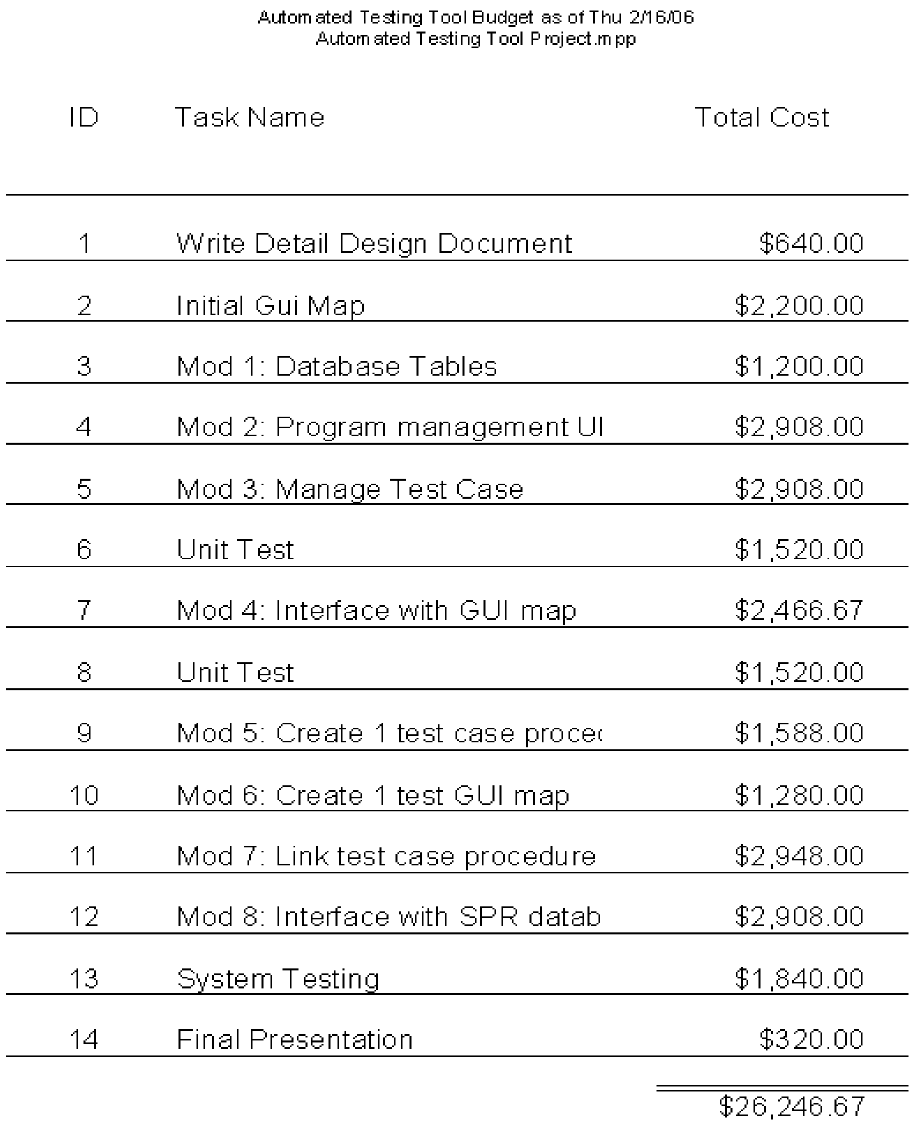
Project: Automated Testing Tool Proje
Date: Thu 3/9/06

Task	▨	Summary	▬	Rolled Up Progress	▬
Task Progress	▬	Rolled Up Task	▨	Split	┆
Critical Task	🟥	Rolled Up Critical Task	🟥	External Tasks	▨
Critical Task Progress	🟥	Rolled Up Milestone	◇	Project Summary	▬
Baseline	▨	Baseline Summary	▬	Group By Summary	▬
Milestone	◆	Rolled Up Baseline	▨	Deadline	⇩
Baseline Milestone	◇	Rolled Up Baseline Milestone	◇		

Page 1

Figure 10

Budget



Automated Testing Tool Budget as of Thu 2/16/06
Automated Testing Tool Project.mpp

ID	Task Name	Total Cost
1	Write Detail Design Document	$640.00
2	Initial Gui Map	$2,200.00
3	Mod 1: Database Tables	$1,200.00
4	Mod 2: Program management UI	$2,908.00
5	Mod 3: Manage Test Case	$2,908.00
6	Unit Test	$1,520.00
7	Mod 4: Interface with GUI map	$2,466.67
8	Unit Test	$1,520.00
9	Mod 5: Create 1 test case proce	$1,588.00
10	Mod 6: Create 1 test GUI map	$1,280.00
11	Mod 7: Link test case procedure	$2,948.00
12	Mod 8: Interface with SPR datab	$2,908.00
13	System Testing	$1,840.00
14	Final Presentation	$320.00
		$26,246.67

Table 1

If this were a real world application, the budget above indicates how much it would cost to produce this Automated Testing Tool. In our case, no money will be needed to produce the application except for $40.00 for the MacroExpress product.

Bibliography

1. CIMx Professional Services. "Technical Package for CIMx Apps v3.4". July 1, 2005.
2. "CocoBase Software Web site". http://www.thoughtinc.com.
3. Dustin, Elfrieded and Rashka, Jeff and Paul, John. *Automated Software Testing: Introduction, Management, and Performance.* New York: Addison-Wesley Professional, 1999.
4. Furguson, Jeff. *C# Bible.* New York: Wiley Publishing, July 2002.
5. Larusso, Robin. "FastGuide: PL/SQL" http://searchoracle.com. June 7, 2004.
6. McClane, Kristin. Operations Manager, CIMx. Personal interview. August 15, 2005.
7. Overland, Brian and Morrison, Michael. *Java 2: In Plain English*. Foster City, CA: M&T Books, 2001.
8. "QA Wizard Software Web site". http://www.seapine.com/qawizard.html.
9. Sceppa, David. *Microsoft ADO.net.* Redmond, Washington: Microsoft Press, 2002.
10. Schwartz, Karen. "Choosing an Automated Testing Tool". http://www.ftponline.com. September 29, 2003.
11. Urman, Scott. *Oracle 9i PL/SQL Programming*. Emeryville, CA: McGraw-Hill Osborne Media, 2001.