2006

TestCASE Tool

John Louis Van Hemert
University of Northern Iowa

TESTCASE TOOL

A Thesis or Project

Submitted

in Partial Fulfillment

of the Requirements for the Designation

University Honors

John Louis Van Hemert

University of Northern Iowa

May 2006

This Study by John Van Hemert, entitled TestCASE Tool, has been approved as meeting the thesis/project requirement for the Designation University Honors.

5-1-06 _____
Date Dr. Janet Drake, Honors Thesis/Project Advisor

5/3/06 _____
Date Jessica Moon, Director, University Honors Program

John Van Hemert
Honors Thesis/Creative Project Reflection
May 2006

Topic Explanation

The title of my project is "TestCASE Tool." TestCASE Tool is the name, or label, I assigned to the software application that I created for this project. The application is a *Tool* for storing and reporting data in software testing projects. A *Test Case* consists of a planned action within a system, the expected result of that action and an assessment of whether the expected result occurred. The acronym, *CASE*, stands for Computer Aided Software Engineering, which is the software tool genre to which my application belongs.

Why I chose this topic

During a large testing project in a Software Engineering course, I found myself spending more time and energy storing, formatting and reporting testing information than I spent on the testing itself. I recognized the how well testing information would fit into a relational database, and decided that creating a software tool that automates much of the "grunt work" of testing would be worthwhile.

Steps Taken to Complete the Project

Research

I began this project by reading from three software engineering books:

- Humphery, Watts. The Personal Software Process.
- Guiney, Eamonn and Kulak, Daryl. Use Cases: Requirements in Context.
- Robertson, Suzanne and Robertson, James. Mastering The Software Requirement Process.
- Anderson,Virginia and Taylor, Allen G. Access 2003 power programming with VBA.

Use Cases

In order to begin to define the requirement of my final product, I created *Use Cases*. Use Cases are descriptions of the normal activities a user conducts in a system. Among other things, Use Cases include a summary and detailed list of events, triggers that cause the activities to begin and problems that may arise. I created four main use cases, along with a diagram that represents how the use cases relate to one another and the users.

Software Requirements Specification

A Software Requirements Specification (SRS or Spec) is a detailed document that defines what and how a new software application must do and behave. I wrote a complete SRS for TestCASE Tool, which represents the main writing document for my project, not only because it required a majority of the semester's work, but it represents the writing style used in my field.

Data Model

Since TestCASE Tool's main purpose is to store and organize testing information, a model for storing data is important. I created a model, called the Entity Relationship Diagram, that represents all the entities (or things) in TestCASE Tool, as well as the pieces of information that those entities contain.

Coding

Coding was, of course, an important step in creating TestCASE Tool, but did not represent a great deal of the effort. This shows that the creative design steps in Software Engineering are more involved that carrying them out in the coding steps. There was one vexing problem in coding TestCASE Tool that involved overcoming limitations in the database management system (Microsoft Access) that I used for this project.

Testing

Before using TestCASE Tool for real testing projects, it had to be tested, itself. This involved entering flagged data and verifying that it was stored and reported correctly.

Implementation

I actually used TestCASE Tool to test another project I was working on this semester. This was worthwhile because the other project required an additional report—the Test Plan, which I added to TestCASE Tool.

Final Work Description

My final work includes the Software Requirements Specification for TestCASE Tool, the actual application in Executable Microsoft Access Database format, stored testing data and reports from the project tested with TestCASE Tool, the PowerPoint slides presented on Saturday April 22, 2006, and four walkthrough videos in flash movie format. All of these items are stored on the CD-ROM that is attached to a hard copy of the SRS and PowerPoint presentation slide printouts. They are also available on the web at http://www.uni.edu/jvh/Projects/TestCASETool

The Importance of this Project

Software testing is a long and tedious process in software development. For normal software, 50% of the development budget is spent on testing. Further, for life-critical software, upwards of 70% of the development budget is spent on testing. This project is an attack at the tediousness of testing, so that testers can focus on the creative and analytical aspects of testing, such as writing and evaluating test cases. TestCASE Tool was designed to be used by one administrator and multiple users, which makes it a perfect tool for a testing project in a software engineering course.

Reflection on my Experience

This project provided me with four main learning experiences. First, I learned how powerful Use Cases can be in gaining an idea of the requirements of a system. Second, I learned the significance of personal time management in a software development. Third, I learned how to develop a Microsoft Access application using the programming language, Microsoft Visual Basic for Applications. Lastly, and perhaps most interestingly, I learned how to create a relation database Outer Join, using only Left Joins and Inner Joins. I developed the "home-made" Outer Join independently, using my own analytical skills and diligence. I plan to research this "trick" in the near future, and perhaps write a paper for it.

Software Requirements Specification

for

Test CASE Tool

Prepared by John Van Hemert

Undergraduate Project

University of Northern Iowa

Spring 2006

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

Test CASE Tool is a system for storage, organization, viewing and reporting of testing information in the classroom setting. Testing information includes projects, test sets, test cases and faults. All of these terms are defined below.

1.2 Document Conventions

Each requirement in this document has its own priority. Each section should be referred to with its name, including the outline number accompanying it. For example, the term Team Member should be referred to as "1.2.1.1 Team Member."

a. Terms

1.2.a.1 Team Member

A Team Member is a student working on a testing project.

1.2.a.2 Project

A Project is a section of a Software Requirements Specification and associated application code.

1.2.a.3 Test Set

A Test Set is a set of test cases with the same environment, settings and general purpose.

1.2.a.4 Test Case

A Test Case is an action in a software system such that the actual consequences of that action (actual result) can be compared to consequences expected according to a software requirements specification (expected result).

1.2.a.5 Fault

A Fault is a Test Case where the actual result is different from the expected result.

1.3 Intended Audience and Reading Suggestions

Developers should read this document thoroughly. Project managers should read Section 2: Overall Description. Users should first read Section 4: External Interface Requirements and then Section 3: System Features.

1.4 Project Scope

In a testing class, the objective is to learn the value-adding process, Software Testing. Documentation format is a trivial aspect of the curriculum, but it often consumes much of students' time working on a project. Test CASE Tool automates the storage, organization and reporting of testing data so that the students and professor can focus on the creation and analysis of testing data.

1.5 References

This document follows the Software Specification format presented by Liebowitz, Agresti and Djavanshir in Communicating as IT Professionals, Pearson Education Incorporated, 2006. An outline is available at http://www.processimpact.com/process_assets/srs_template.doc .
TBD

2. Overall Description

2.1 Product Perspective

Test CASE Tool is a stand-alone data-driven system. It is the end-product of an undergraduate project conducted by John Van Hemert in Spring 2006.

2.2 Product Features

Test CASE Tool shall allow students to store and report test data for their projects. It shall allow the professor to view all students' test data (See TestCase Use Case Diagram).

2.3 User Classes and Characteristics

a. Clients

2.3.a.1 Dr. Janet Drake, Project Advisor

This project was conducted under the advisement of Dr. Drake, who specializes in Software Engineering, Requirements Elicitation and System Modeling.

2.3.a.2 Jessica Moon, University of Northern Iowa Honors Program Director

This project was conducted as a Senior Honors Project and evaluated by Jessica Moon.

b. Users of Test CASE Tool

2.3.b.1 Students

Students will use the Test CASE Tool to organize and report test data for their testing projects.

2.3.b.2 Professors

Professors will use the tool to organize and manage class project materials.

c. Other Stakeholders

2.3.c.1 Users of software tested with Test CASE Tool

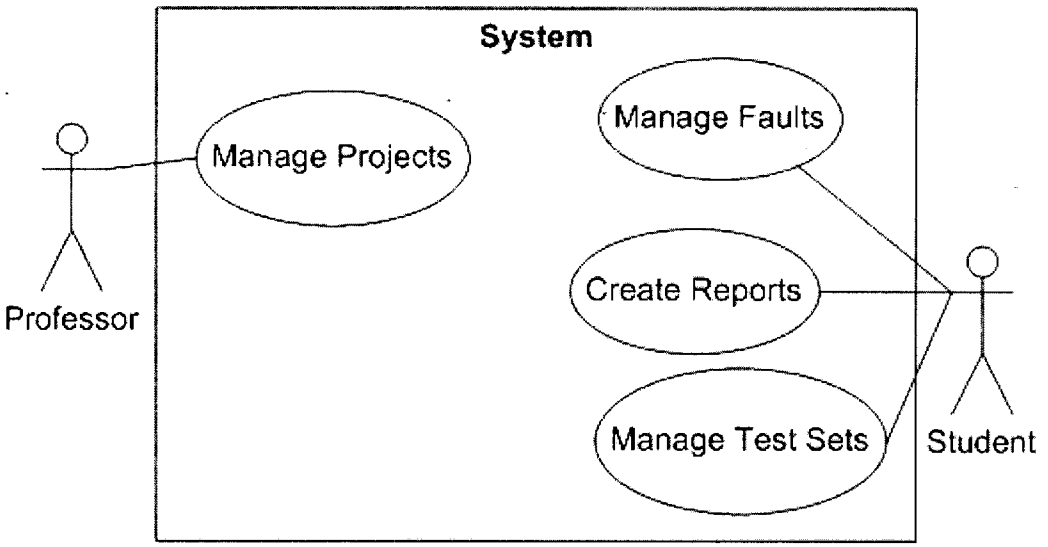
Users will experience software of higher quality due to the use of the tool.

2.3.c.2 Authors

Software authors will receive simpler and more organized feedback for their products within shorter timeframes.

2.4 Operating Environment

a. System Context Use Case Diagram



2.5 Design and Implementation Constraints

Test CASE Tool shall run as a Microsoft Access database application, and it is limited to machines with Microsoft Access installed.

2.6 User Documentation

Test CASE Tool shall include a user manual with instructions for the tasks conducted by all users (professor and students). One user can use one project at one time.

2.7 Assumptions and Dependencies

It is assumed that Microsoft Access will facilitate concurrency for multiple users.

3. System Features

3.1 Manage Projects

a. Description and Priority

The professor shall create, modify, view and remove student projects. This is of high priority because it is the starting point of the context process.

b. Stimulus/Response Sequences

When there is a software requirements specification to be tested, the professor will assign students to teams and create testing projects that refer to sections of the software requirements specification for the teams. The professor will then view the progress of all projects as needed, modify project information as needed and removed projects from the system when they are completed.

c. Functional Requirements

3.1.c.1
The professor shall add student information to build teams and the student information will be stored in the database. If a student has already been entered, an error message will be displayed and nothing changed in the system.

3.1.c.2
The professor will then create a project and assign one or more teams to the project. The project and its information shall be stored in the database. If there is already a project entered of the same name, an error message will be displayed and nothing changed in the system. A project is a unique section of a Software Requirements Specification that is to be tested.

3.1.c.3
The professor will be able to modify project information. When modifications are made, the new information will be set in the database.

3.1.c.4
The professor shall be able to view the progress of all student projects. The professor will select projects to view their information, including team assignment, test sets, test cases and faults.

3.1.c.5
The professor will be able to remove projects from the system. This will remove a project, its assigned students, its test sets, its test cases and its faults.

d. Use Case

Use Case Name	Manage Projects
Iteration	Finished
Summary	Professor creates, modifies, views and removes student projects.
Basic Course of Events	1. Professor adds student information in a student team 2. Professor sets up a project with students assigned to it
Alternative Paths	1. Professor modifies project information 2. Professor views progress of student projects 3. Professor removes a project (and the students assigned to it) from the system
Exception Paths	Event 1: There is already a student of the entered name → the professor is notified and nothing is changed Event 2: There is already a project of the entered name → the professor is notified and nothing is changed
Extension Paths	Anytime after Event 2: students conduct use cases: Manage Testing Information, Manage Faults, Make Reports
Triggers	A software project needs to be tested
Assumptions	1. "Project" means a unique section of an SRS that a particular student team is testing 2. The professor has assigned students to student teams
Preconditions	For Alternative Paths 1,2,3: the project exists
Postconditions	Event 1: the student information is in the system Event 2: the project information is in the system Alt Path 1: the modifications are set to the project Alt Path 2: project information is displayed Alt Path 3: the project and students assigned to it are not in the system
Related Business Rules	None
Author	John Van Hemert
Date	Façade completed: 1/22/06 Filled completed: 1/30/06 Focused completed: 2/6/06 Finished completed: 2/16/06

3.2 Manage Test Sets

a. Description and Priority

A student shall enter, modify, view and remove test sets and test cases belonging to the student's team and project. This is of high priority because the test sets are the principal entity in the system.

b. Stimulus/Response Sequences

When a student or student team conducts a test for a project, they will create a test set and test cases within the test set, storing the test set and test cases in the system. Once test sets are created, a student will modify test sets and test cases as needed, which will set the changes to the system. The students will view test sets as needed, displaying their data. Students will also remove test sets and test cases as needed, deleting them from the system.

c. Functional Requirements

3.2.c.1
A project assigned to a student will be the only project accessible to that student.

3.2.c.2
The student will create test sets and test cases within test sets.

3.2.c.3
The student shall be able to modify the test sets and test cases belonging to the student's team and project.

3.2.c.4
The student shall be able to view the test sets belonging to the student's team and project and the test cases they contain.

3.2.c.5
The student shall be able to delete the test sets belonging to the student's team and project and the test cases they contain.

d. Use Case

Use Case Name	Manage Test Sets
Iteration	Finished
Summary	Student enters, modifies, views and removes testing information.
Basic Course of Events	1. Student creates a Test Set 2. Within the Test Set, the student creates a Test Case.
Alternative Paths	1. Student modifies a Test Set or the Test Cases it contains 2. Student views Test Sets or the Test Cases they contain 3. Student removes Test Sets or the Test Cases they contain
Exception Paths	None
Extension Paths	1. Student conducts use cases Manage Faults or Make Reports 2. Professor conducts use case Manage Projects
Triggers	*Student team conducts a test*
Assumptions	None
Preconditions	Project has been set up by professor For Event 1, Alt Paths 1,2,3: Test Set exists
Postconditions	Event 1: new Test Set is in system Event 2: new Test Case is in system, under the Test Set Alt Path 1: Test Set is changed Alt Path 2: Test Sets are displayed Alt Path 3: Test Set is removed
Related Business Rules	Refer to testing documentation description There can be many different Test Sets for a project. There can be many different Test Cases for a Test Set
Author	John Van Hemert
Date	Façade completed: 1/22/06 Filled completed: 1/30/06 Focused completed: 2/6/06 Finished completed: 2/16/06

3.3 Manage Faults

a. Description and Priority

A student shall enter, modify, view and remove fault information belonging to the student's team and project. This is of high priority because fault discovery and logging are the objectives of testing.

b. Stimulus/Response Sequences

When a student stores a test case that does not meet its expected results, the student will create a fault report for the test case, storing the fault in the system. Once fault reports are created, a student will modify faults as needed, which will set the changes to the system. The students will view fault reports as needed, displaying their data. Students will also remove fault reports as needed, deleting them from the system.

c. Functional Requirements

3.3.c.1
The student will create fault reports indicated failed test cases.

3.3.c.2
The student shall be able to modify the fault reports belonging to the student's team and project.

3.3.c.3
The student shall be able to view the fault reports belonging to the student's team and project.

3.3.c.4
The student shall be able to delete the fault reports belonging to the student's team.

d. Use Case

Use Case Name	Manage Faults
Iteration	Finished
Summary	Students creates, modifies, views and removes faults
Basic Course of Events	1. Student creates new Fault, corresponding to a Test Case 2. Student enters new fault information
Alternative Paths	1. Student modifies Fault information 2. Student views Fault information 3. Student removes Fault information
Exception Paths	None
Extension Paths	Student removes a Test Case (this cascade deleted the corresponding Fault if it exists)
Triggers	A Test Case result does not match the expected result
Assumptions	None
Preconditions	There is a Test Case that corresponds to the Fault
Postconditions	Events 1,2: new Fault is stored in system and corresponds to a Test Case Alt Path 1: Fault modifications are set in system Alt Path 2: Fault information is displayed Alt Path 3: Fault is not in system
Related Business Rules	See Fault documentation
Author	John Van Hemert
Date	Façade: 2/16/06 Filled: 2/16/06 Focused: 2/16/06 Finished: 2/16/06

3.4 Create Reports

a. Description and Priority

Students shall generate and print a Test Summary, Fault Report and Test Set. This is of medium priority because it is not essential to the organization of testing data.

b. Stimulus/Response Sequences

When the student wishes to print reports on the project belonging to the student's team, the student views the Test Summary, Fault Report and Test Sets separately and prints each.

c. Functional Requirements

3.4.c.1
If there are no test sets in a project, the user is notified and reports are not generated.

3.4.c.2
See the documentation on Test Summary, Fault Report and Test Set format for report output.

3.4.c.3
The student can view each report and then print the screen display.

d. Use Case

Use Case Name	Create Reports
Iteration	Finished
Summary	Students select and print a Test Summary, Fault Report or Test Set
Basic Course of Events	1. Student views the Test Summary, Fault Report or Test Sets for a project 2. Student prints each report
Alternative Paths	There are no Faults → Test Summary and Fault Report state that there are no Faults
Exception Paths	There are no Test Sets → user is notified that there are not Test Sets and no reports are generated
Extension Paths	None
Triggers	Students wish to generate reports on their project
Assumptions	None
Preconditions	The student is assigned to a project
Postconditions	The Test Summary, Fault Report and Test Set are printed
Related Business Rules	See Test Summary, Fault Report and Test Set documentation
Author	John Van Hemert
Date	Façade: 2/16/06 Filled: 2/16/06 Focused: 2/16/06 Finished: 2/16/06

4. External Interface Requirements

4.1 User Interfaces

The application will use MS Access forms.

4.2 Hardware Interfaces

Test CASE Tool will run on the workstations in University of Northern Iowa's College of Natural Sciences student workstations. It will also print on CNS lab printers.

4.3 Software Interfaces

The application is created with Visual Basic for Applications and MS Access 2003. The end product is a MS Access Database Executable, which is a MS Access Application, with all VBA code compiled, and access to form interfaces only.

4.4 Communications Interfaces

The application will run on a network drive, accessed by the users from connected workstations with MS Access 2002 or 2003.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

All operations and queries shall complete or present errors within one minute of their invocation.

5.2 Safety Requirements

NA

5.3 Security Requirements

Administrator (professor, etc), alone can manage projects and users. The other users can only manage the project assigned to them by the administrator.

5.4 Software Quality Attributes

Maintenance

Corrective and Perfective: SRS, code exist to assist in maintenance.
Adaptive: The source application can be adapted in MS Access.

6. Other Requirements

6.1 Data Requirements

a. See Appendix A for Entity Relationship Diagram.

6.1.a.1 TeamMember
Any user of the application

6.1.a.2 Project
A section of an SRS, code or application to be tested.

6.1.a.3 TestSet
A part of a Project that involved the same specific purpose and environment.

6.1.a.4 TestCase
A step in a TestSet.

6.1.a.5 Fault
A report of a TestCase in which the Actual Result differs from its Expected Result.

6.1.a.6 SeverityLevel

High – User cannot complete his task while this fault exists
Medium – Fault stops work on this path but a work-around is available
Low – Fault does not affect the user's task

6.1.a.7 PriorityLevel

High – Fault must be fixed immediately
Medium – Fault will be fixed in this cycle
Low – Fault need not be fixed in this cycle

6.1.a.8 Status

Pending – Fault has been entered into the system
Open – The fault has been accepted
Assigned – The fault has been assigned to an engineer
Retest Ready – Fault has been corrected and is ready for test
Closed – Fault has been successfully fixed
Duplicate – Fault report is a duplicate
Rejected – Fault is not a fault

A. Appendix A: Glossary

a. Test Set and Test Case Format

Test Procedure ID:
Test Procedure Purpose:
Test Environment and Setup:
Test Case Steps and Pass Fail Criteria

Step No.	Action and/or Pass Fail Criteria	Result	Expected Results with Tolerances

b. Fault Report Format

Date:
Fault Title: (Short title)
Material being tested:
Test case/step:
Fault Description: (Detail description)

Severity: (High, Medium, Low)
Priority: (High, Medium, Low)
Repeatable: (Yes, No)

Example

Date: 03/30/04
Fault Title: Area Code Accepts Characters
Material being tested: VCMS, Customer Order
Test Case/Step: Test Case 100 step 30
Fault Description: For the phone number area code "aaa" was entered and the program accepted the invalid value.
Severity: Low
Priority: Low
Repeatable: Yes

c. Test Summary Format

Number of test cases:
Number of test cases passed:
Number of test cases failed:

Severity
Number of **High**:
Number of **Medium**:
Number of **Low:**

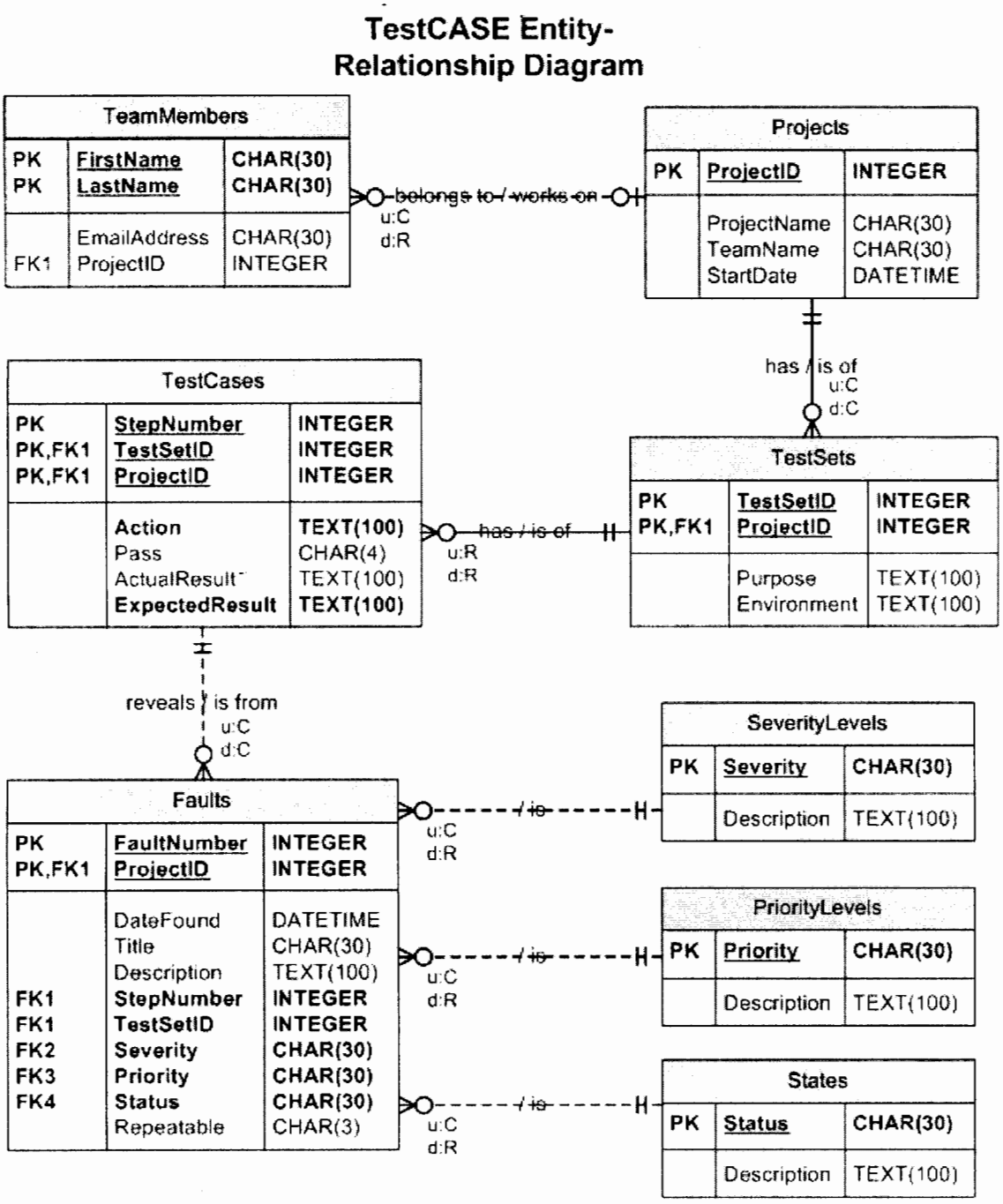
Fault Report

Fault number: (number sequentially)
Title:
Description:
Test Case & test step number:
Severity:
...

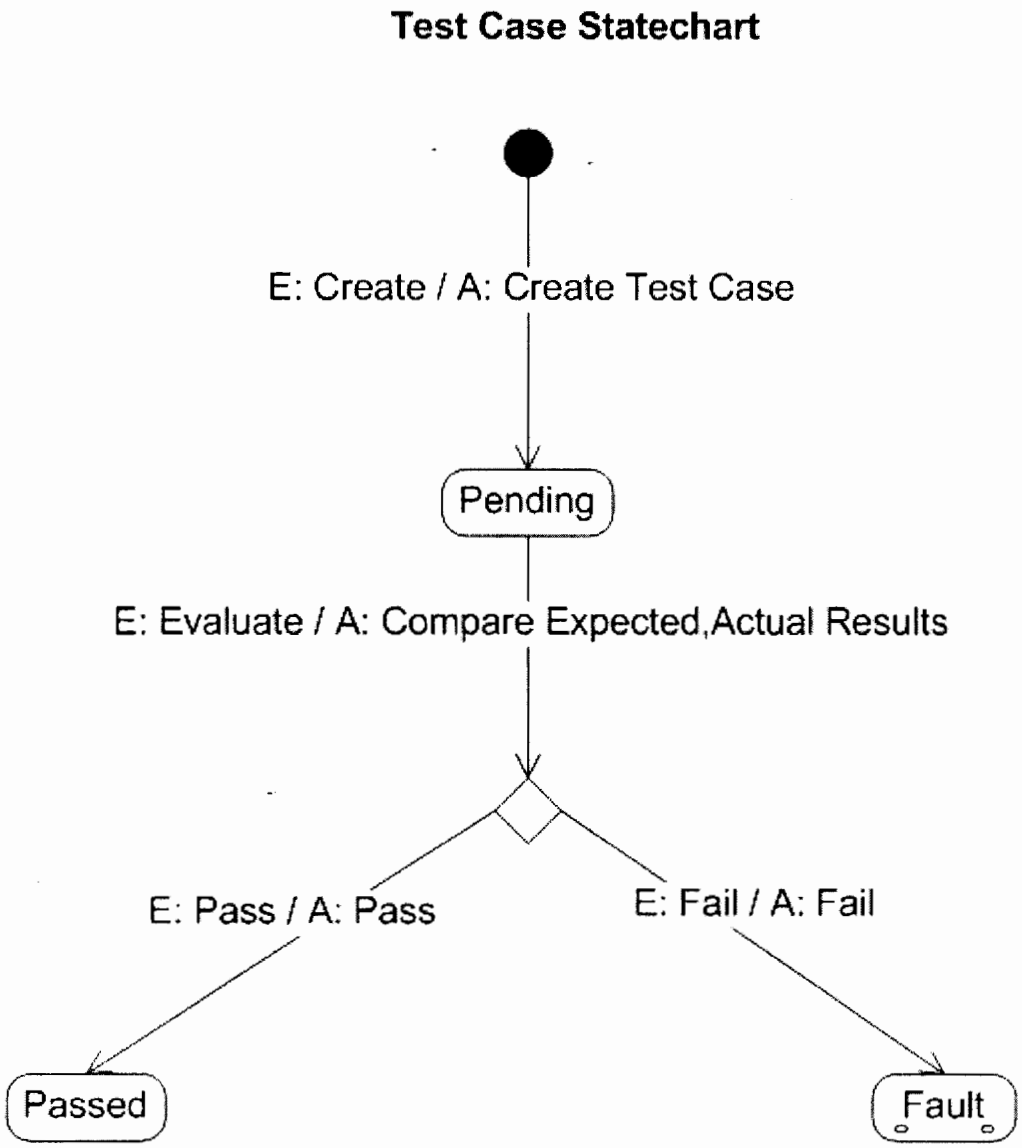
High: Crashes system or data lost or corrupted.
Medium: Can still use the system successfully by working around the problem.
Low: Incorrect but does not interfere with system functionality.

Appendix B: Analysis Models

a. Entity-Relationship Diagram

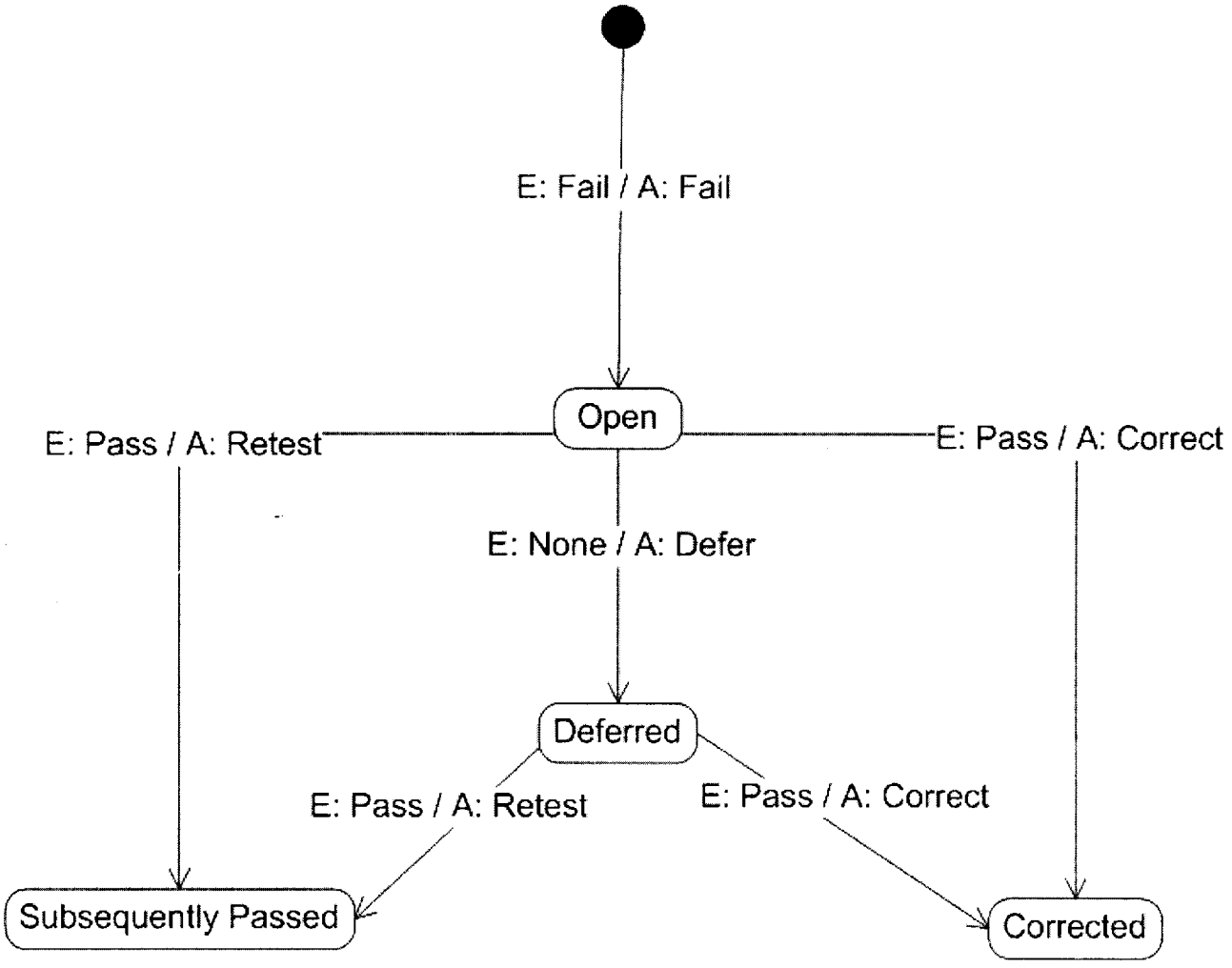


b. Statechart for Test Cases



c. Statechart for Faults

Fault Statechart
(subchart of Fault state in Test Case Statechart)



Appendix C: Issues List

NA

An undergraduate software development project

John Van Hemert

Spring 2006

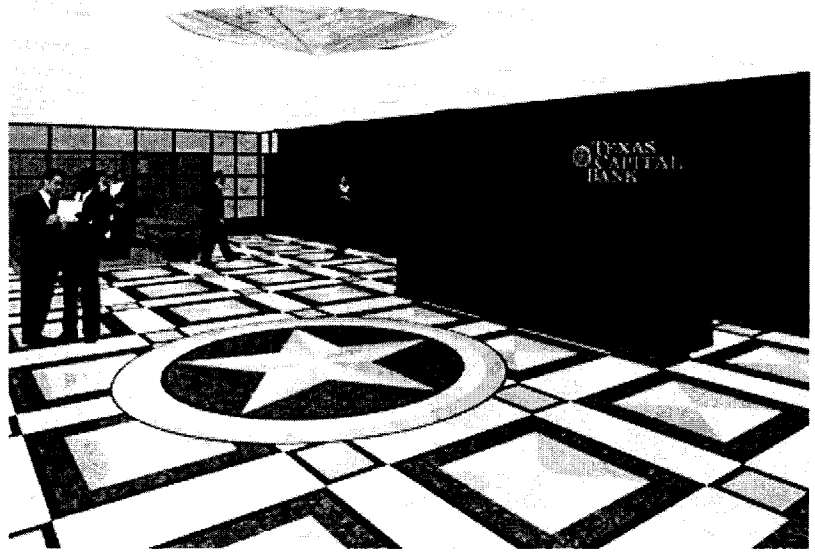
TestCASE Tool

Test Case: ***An action for which there is an expected result to be compared to an actual result.***

CASE: ***Computer Aided Software Engineering***

Overview

- **Need**
  - **Why this project is important**
- **Software Requirements Specification**
  - **What/Why**
- **TestCASE Tool Demonstration**



----------Software----------

--

Watts Humphery: The Personal Software Process

A production system and paradigm that focuses on:

1) **Time management**
   for providing solid statistical timeframe data on future project bids

2) **Defect minimization**
   for Quality Management

Watts Humphery: The Personal Software Process

- Software Engineering: Deliver high-quality software at a set cost and schedule
  - Plan work
  - Work according to plan
  - Strive for quality
- Engineering Notebook
  - Categorize activities (Metric)
    - Size (LOC)
    - Complexity (# of modules, interfaces)
    - Development Stage
  - Record time spent on each major activity

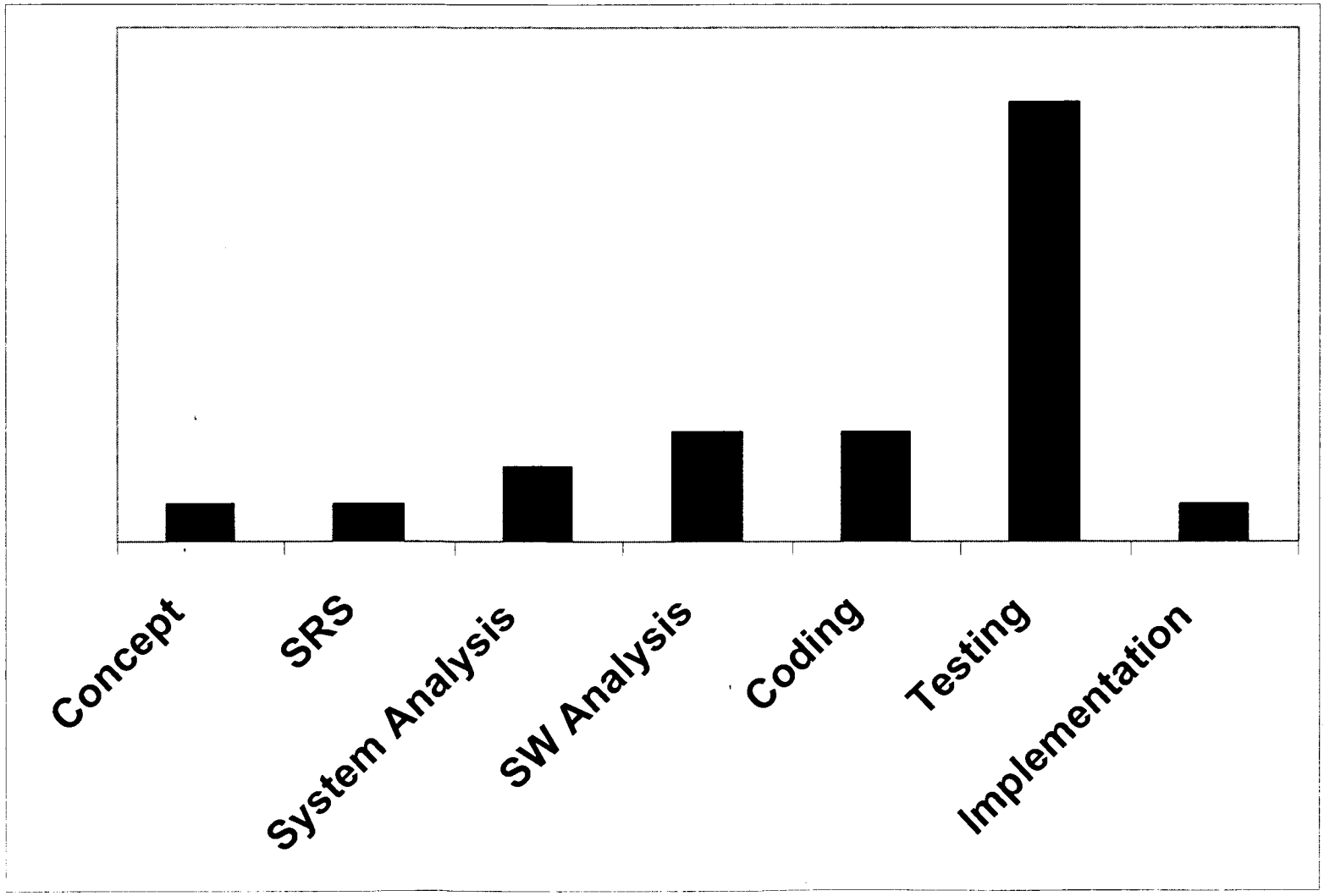
Watts Humphery: The Personal Software Process

- Historical data →
  - Accurate bids
    - "How long will it take me to complete a task of this complexity and size?"
  - Increase quality (fewer faults)
    - Focus on quality (accurate) planning
    - Less deadline pressure
- Communication
  - Vertical
  - Horizontal

Software Engineering

1. **System Concept**
2. **System Requirements Specification**
3. **System Analysis**
4. **Software Analysis**
    1. **Architectural Design**
    2. **Detailed Design**
5. **Coding**
6. ***Testing***
7. **Implementation**
8. **(Maintenance)**

SE Costs


Concept
SRS
System Analysis
SW Analysis
Coding
Testing
Implementation

Testing Effort

- **Extremely important**
- **Also extremely expensive**
- **Normal Software ~50%**
- **Life-Critical Software ~70%+**

Software Requirements Specification

- Product Constraints
  - *The Purpose of the Product*
  - *The Client, Customer, and other Stakeholders*
  - *Users of the Product*
  - *Requirements Constraints*
- Functional Requirements
  - *The Scope of the Product*
  - *Functional and Data Requirements*
- Non-functional Requirements
  - *Usability*
  - *Performance*
  - *Security*
  - *Legal*
- Project Issues
  - *Open Issues*
  - *Cutover*
  - *Risks*
  - *Costs*

TestCASE Tool SRS

TestCASE Tool Process

- **User navigates and works through forms**
  - **Forms: windows with buttons, lists, text boxes, etc.**
- **Begin with a login form (window)**
- **Username determines:**
  - **Which forms to open**
  - **Access to data (Project(s), data management forms, etc.)**

TestCASE Tool Process Diagram

Demonstration

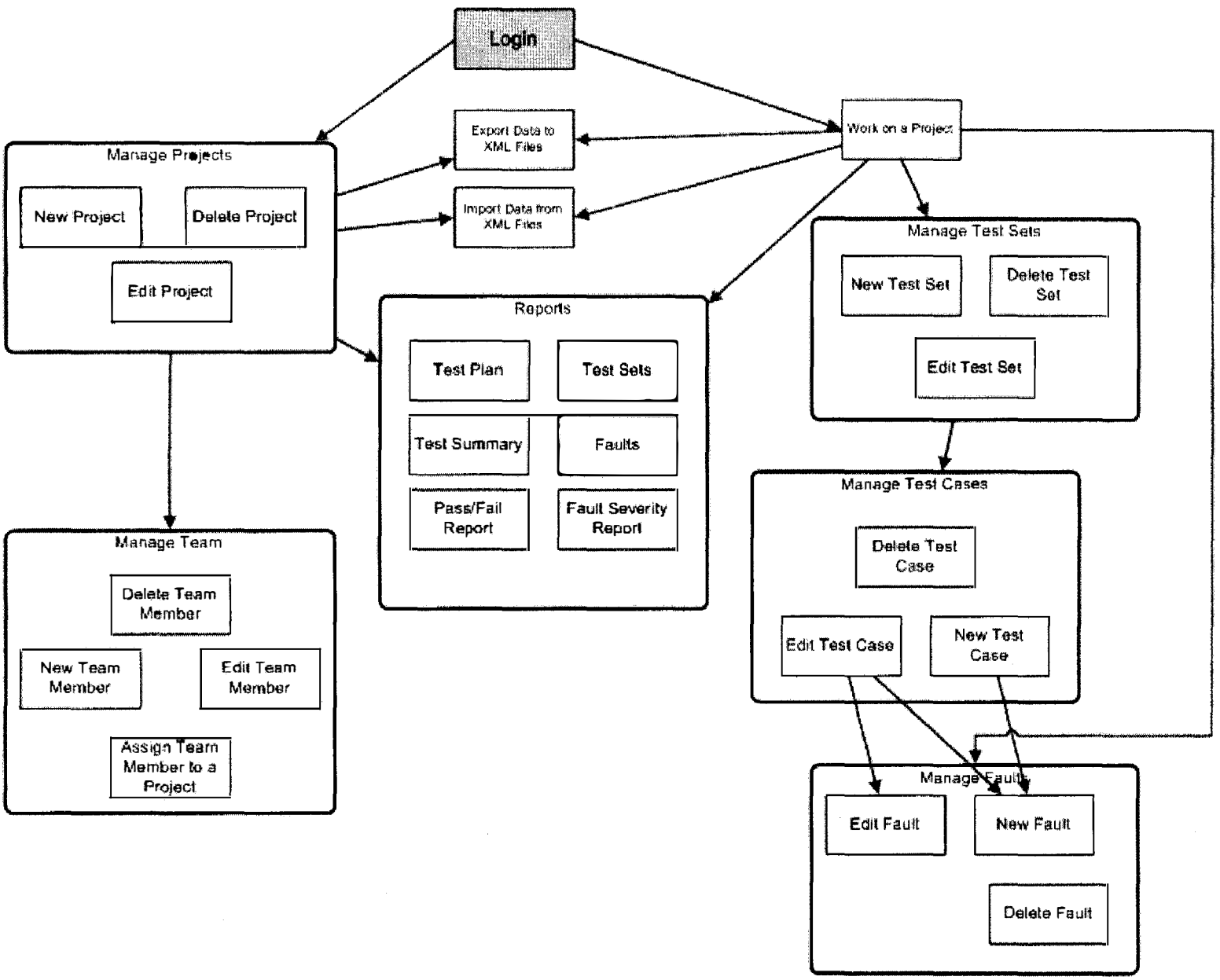
- **Use Cases: A system's processes from the user's point of view.**
  - **Manage Projects**
    - **Demo**
  - **Manage Test Sets**
    - **Demo**
  - **Manage Faults**
    - **Demo**
  - **Create Reports**
    - **Demo**

Reports

- Test Plan
- Test Sets
- Faults
- Test Summary
- Pass/Fail Report
- Severity Report
- Administrator only:
  - Aggregate Pass/Fail Report
  - Aggregate Severity Report

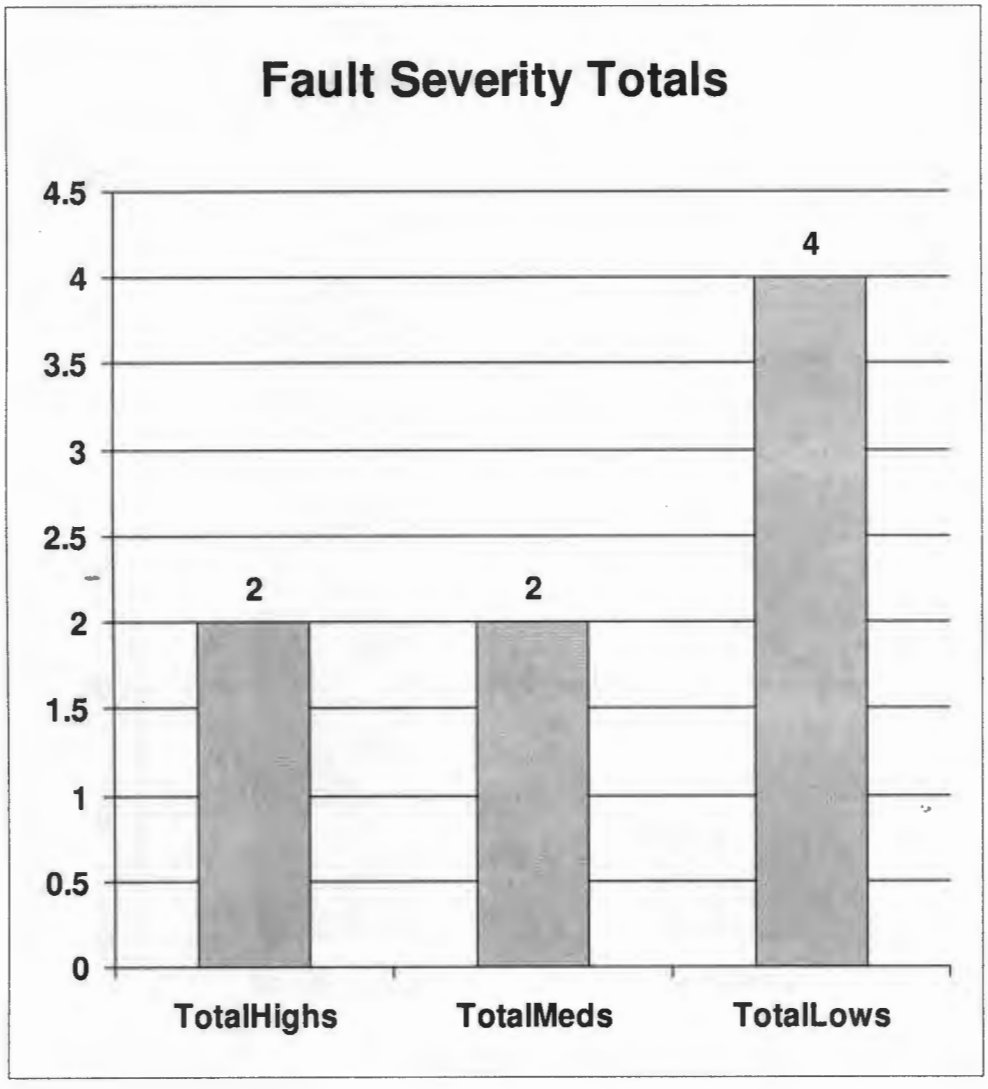
Further Research, Projects

- TestCASE Tool on the Web
- PSP CASE Tool
- SQL Algebra


Login
Export Data to XML Files
Import Data from XML Files
Work on a Project
Manage Projects
New Project
Delete Project
Edit Project
Reports
Test Plan
Test Sets
Test Summary
Faults
Pass/Fail Report
Fault Severity Report
Manage Test Sets
New Test Set
Delete Test Set
Edit Test Set
Manage Test Cases
Delete Test Case
Edit Test Case
New Test Case
Manage Team
Delete Team Member
New Team Member
Edit Team Member
Assign Team Member to a Project
Manage Faults
Edit Fault
New Fault
Delete Fault

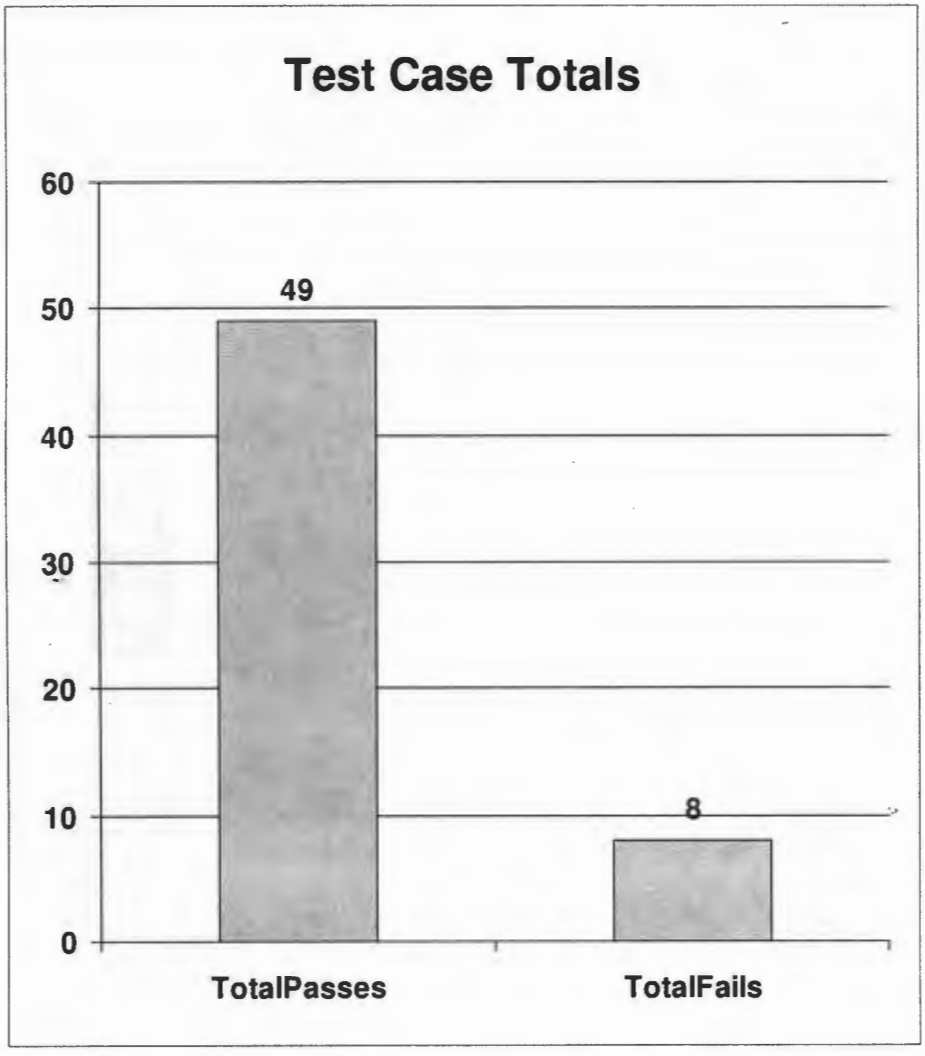
Fault Severity Report For All Projects

Total Highs:	2	(25 %)
Total Mediums:	2	(25 %)
Total Lows	4	(50 %)
Total Faults	8	



Pass/Fail Report For All Projects

Total Passes:	49	(	85 %)
Total Fails:	8	(	14 %)
Total Cases:	57		



Test Plan

Project ID: 1

Project Name: Submit New Housing Information

Team Name: 2-3

Start Date: 4/12/2006

Test Set ID: 1

Purpose: Test New Contact option in Step 1 of the listing submittal process and Step 2: Enter Housing Information

Environment: Not logged into admn tool.

Step Number	Action	Expected Result
1	Visit http://nisg.freewebsitehost.net/housing.php in web browser.	Display Housing Site main page.
2	Click Submit A Listing on left nav-bar.	Display Step 1: Enter Contact Information, and two link choices: Returning Contact and New Contact.
3	Select New Contact link.	Display New Contact form: First Name, Last Name, Email Address, Phone Number.
4	Enter John for First Name, Test for Last Name, jvh@uni.edu for Email Address, 319-222-4810 for Phone Number (submit)	Display Step 2: Enter Housing Information form
5	Submit: Type(House), Address(123 Test Street),Blocks From Campus(3),Date Available(2006-04-30),Total Monthly Rent(900),Max Occupants(3),Number of Bedrooms(2), Utilities Included(Yes),Air Cond(Yes),CableTV(Yes),Pets(No),Smoking(No),Notes(Test Notes)	Display 'Successful listing submitted' message

Test Sets

ProjectID: 1

Project Name: Submit New Housing Information

Team Name: 2-3

Start Date: 4/12/2006

Test Set ID: 1

Purpose: Test New Contact option in Step 1 of the listing submittal process and Step 2: Enter Housing Information

Environment: Not logged into admn tool.

Step Number	Action	Expected Result	Actual Result	Pass
1	Visit http://nisg.freewebsitehost.net/housing.php in web browser.	Display Housing Site main page.	Display Housing Site main page.	Pass
2	Click Submit A Listing on left nav-bar.	Display Step 1: Enter Contact Information, and two link choices: Returning Contact and New Contact.	Display Step 1: Enter Contact Information, and two link choices: Returning Contact and New Contact.	Pass
3	Select New Contact link.	Display New Contact form: First Name, Last Name, Email Address, Phone Number.	Display New Contact form: First Name, Last Name, Email Address, Phone Number.	Pass
4	Enter John for First Name, Test for Last Name, jvh@uni.edu for Email Address, 319-222-4810 for Phone Number (submit)	Display Step 2: Enter Housing Information form	Display Step 2: Enter Housing Information form	Pass
5	Submit: Type(House), Address(123 Test Street),Blocks From Campus(3),Date Available(2006-	Display 'Successful listing submitted' message	Display 'Successful listing submitted' message	Pass