

ITS Graphical Report Maker

Cheng-Train Chiou, John Myers, Adam Buehler, and Cesario Tam

Abstract—Systems Management in ITS at RIT needs a tool that can generate Reports that analyze data from a large database of information. The main function of this tool is the creation and execution of Reports and exporting results to an external format. This tool needs to be a distributed system that provides centralized persistency. The development strategy uses a plan that is based on the existing ITS process. Using the ITS process as guideline of what is expected, a list of deliverables was negotiated at the beginning of the process. A summary of the development plan discusses the initial project plan, milestones, and actual results of the plan. A description of the architecture and design covers the technologies used and our reflections about the project.

Index Terms—Software Engineering, Requirements/Specifications, Design Tools and Techniques, Design Techniques, Software Verification, Test Documentation, Maintenance, Management, Design, Software Architectures, Software Engineering Process, Graphs and networks, Relational databases, Client/server and multitier systems

1 INTRODUCTION

Rochester Institute of Technology (RIT) Information and Technology Services (ITS) provides RIT with various IT oriented services. The four primary categories of services ITS provides are: Administrative, Customer Support, Institutional Research & Policy Study, and Technical Support. Of these services, Technical Support is comprised of Enterprise Technologies and Architecture, Data Center Operations, Systems Management Tools and Reporting, Program Management Office, Telecommunications, Network and Systems Engineering and Support. This group is responsible for RIT's computing and networking infrastructure that supports RIT's information technology needs.

One of the responsibilities of ITS Systems Management is the generation of real-time data, historical data, and Reports with graphs on the capacity, availability and responsiveness of ITS supported services. This data is used to show ITS system performance to customers, help support staff manage the systems, and provide information for RIT leadership when making technical and business related decisions.

ITS has a database of raw data, collected from different RIT computing and networking systems. This data can be used to determine RIT's system network throughput, latency, etc. Currently, ITS uses many commercial off the shelf products to analyze the data contained within the database of network information. This is sufficient for technical and engineering staff use, but difficult to present to management or decisions makers.

The goal of this project is to develop a toolset to allow end-users to create a Report containing self-selected data elements and presenting them in the manner the user sees fit. The tool will extract data from the existing database. The data retrieved is then analyzed using provided statistical libraries. The analyzed data is then represented in a graph or table format. We named the tool Graphical Report Maker (GRM).

An acceptable solution will provide features for:

- Creating database queries.
- Processing the retrieved data.
- Displaying processed data in a graphical or tabular format.

- Exporting the graph or table to an external format for example JPEG
- Provide persistency.

The figure 1.1 is a representation of what the GRM system will do. RIT network data is collected and stored in the network performance database. An ITS system administrator creates a Report using a client interface, and then executes the Report. The GRM system then produces a graph, exports it into JPEG format and displays on an ITS webpage. An external program or script is also capable of executing the Reports contained within the GRM system.

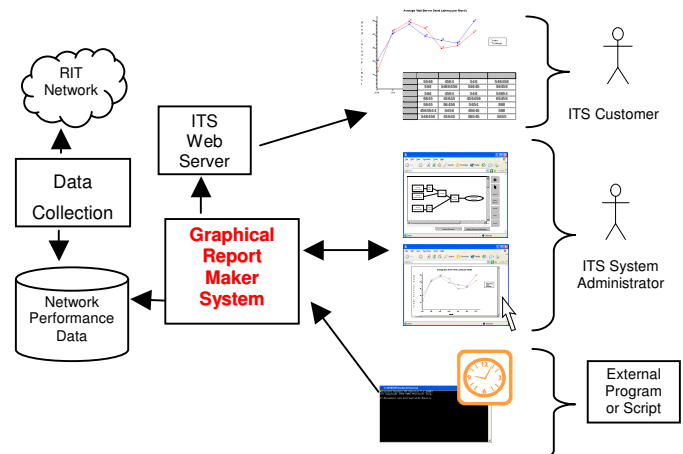


Figure 1.1 – System Interaction Diagram

2 PROJECT REQUIREMENTS

2.1 Users and Operational Environment

There is only one user class for the GRM system. This user class has access to all available functionality, as described fully in the Software Requirements and Specification (SRS). The user accesses the system via a web console which loads a Java thin-client as the graphical user interface (GUI).

2.2 Functional Requirements

The main goal of an end-user of the GRM system is to execute a Report. A Report Element consists of Rover Elements

that retrieve raw data from a database, Operation Elements that process the raw data into meaningful information, and a Generator Element that generates a graphical or tabular Report of the processed data. After executing a Report, the end-user can export the generated graphical or tabular result into an external format such as a JPEG image or CSV file stored in a specified location.

The GRM system provides the end-user with the ability to build each type of Element. To build a Rover Element the end-user needs to provide attributes that can be built into a SQL query for obtaining raw data from the database. A list of Operation Elements is already provided within the GRM system for the end-user to select from. Custom Operations can be added using dynamic class loading. The end-user implements the Operation class to perform a custom operation and places the compiled class into the operation directory. To build a Generator Element the user can select either a Graphical Generator or Tabular Generator and provide the attributes for the graph or table to be generated. A Graphical Generator produces a line graph from its input while Tabular Generator produces a table.

The end-user can also build composite Elements which contain more than one Element. A Report is a composite Element that contains at least one Rover and one Generator. Composite Elements with no inputs and at least one output are categorized as Rovers. Composite Elements with at least one input and at least one output are categorized as Operations. Composite Elements with at least one input and no outputs are categorized as Generators. Composite Elements with no inputs and no outputs are categorized as a Report and are executable.

Preview is a function the end-user can use to check the results of executing an individual Element. Previewing a Rover will generate a generic tabular result of the Rover's output(s). Previewing an Operation will generate a generic tabular result of the Operation's output(s) using generic input(s). Previewing a Generator will generate the graph or table minus any data.

The GRM system also provides persistency by allowing the user to save Elements he has built. The ability to delete Elements compliments the save function for basic management of persistent Elements.

2.3 Non-Functional Requirements

The GRM system interacts with two external systems. One external system is a database running MySQL maintained by ITS with a pre-existing schema. The GRM system also provides a console interface that allows external scripts, possibly scheduled on a periodic basis, to execute Reports and export results.

Sufficient documentation accompanies the GRM system to provide ease of use and maintenance. An operations manual describes the use of the GRM system. Design documents describe both the high-level architectural view and detailed-level class view of the GRM system design. A deployment plan describes how to install and set up the GRM system. Post development system support has also been discussed with ITS management.

Design considerations contribute to the maintainability of the system. The server system is divided into various

components that are well documented in the High-Level Design.

Performance, availability, extensibility and usability attributes of the GRM system are also specified in the SRS document.

3 DEVELOPMENT STRATEGY

3.1 Process Model

ITS uses a process similar to the waterfall software process model. There are six phases in the ITS process: initiate, analyze, design, develop, test, and implement. For each phase there is a corresponding phase gate. The phase gate is a meeting in which deliverables and project status are presented to ITS management. At each phase gate meeting the management reviews the project status and signs off on the deliverables signifying the end of the phase. It is possible to be in more than one phase at a time; however it is recommended that the project team is never in more than two phases at a time.

ITS does not dictate specific deliverables for each phase, instead they provide a purpose for each phase and we determined the specific deliverables that will satisfy the purpose of each phase. For our project the list of deliverables was negotiated during the initiate phase gate.

The initiate phase serves as a starting point for a project and determines feasibility and/or need of a project. In addition to the proposed list of deliverables, we also discussed our plan and strategy for the project.

The analyze phase entails further examination of the requirements of the project through an elicitation process. For this phase gate we presented detailed level requirements and test plans for verifying that our product matches the requirements specified.

Once requirements and specifications are determined, the next step is designing a system that can satisfy those requirements and specifications. A high-level design and detailed-level design were presented at the design phase gate to describe the implementation of the GRM system.

The develop phase consists of coding the system and preparing user documentation for use of the system. Source code, API, and an operations manual are products from this phase.

When development of the system is completed, the test phase verifies and validates that the system satisfies requirements determined in the analyze phase. Results of the testing are presented during this phase gate.

Finally the implement phase involves deploying the GRM system. During the phase gate a deployment plan is presented.

3.2 Roles and Responsibilities

As part of our development strategy, we assigned roles to each member of the development team. Each role came with certain responsibilities to ensure that these tasks are distributed among each member in the development team.

The team leader is responsible for team moral and task accountability. He also acts as primary contact for any official persons.

The quality assurance engineer is responsible for quality

of work and reviews each document for consistency and completeness.

The planning coordinator is responsible for scheduling dates and deadlines for deliverables. He also tracks any changes in the schedule and manages the risks on slippage.

The support manager maintains the development server and tools used. He also keeps the team website up to date.

The release management engineer ensures documents and code are properly prepared, organized and dispersed.

The customer liaison acts as the point-of-contact with the customer.

The development lead manages the design and code. He is in charge of making final technical decisions and trade-offs in the system.

The secretary takes meeting minutes and disperses them to the team.

3.3 Requirements Elicitation Methodology

The methodology we used for requirements elicitation involved a mix of informal interviews with the customer and email correspondence.

During the initiate and analyze phase we had two-hour weekly meetings with the customer. For each meeting we prepared a list of questions for the customer. This allowed the customer to come prepared to answer the questions and keep the meeting time short. The questions addressed at the interviews provided in depth details of the customer requirements.

Email correspondence involved simpler questions that could be effectively answered through text. This method was effective when the customer was not readily available.

The combination of face-to-face interviews and e-mail provided an effective arrangement for both the customer and the team to gather requirements.

3.4 Requirement Specification Tools

The techniques and tools we used for requirements elicitation included use cases, finite state diagram and the definition of a formal grammar.

Use cases showed our customer exactly how an end-user will use the GRM system. This technique allowed the customer to verify that the system will be capable of doing what is needed and also led to new requirements. Analysis of the use cases lead to normalizing our use cases. Any repeating use scenario was factored into a common use case.

A finite state diagram provided the customer a visual representation of the states of the GRM system. Using the finite state diagram we could show the customer the sequence the GRM end-user would follow to accomplish each use case scenario.

A formal grammar provided a definition of the Elements within the system. The grammar specified how each Element is represented within the GRM system and ensured consistency within the definitions.

3.5 Design Methodology

Our design methodology started with researching the technologies that could be used. We performed a technology comparison that is documented in the Technology Comparison Paper. After investigating the trade-offs of each technology and architecture, we made a decision to use

Java Remote Method Invocation (RMI) technology.

Following the architectural research we developed the detailed-level design based on our high-level design which consists of the UML class diagrams and component descriptions.

4 DEVELOPMENT PLAN

4.1 Initial Plan and Estimates

The initial plan and estimates are in the Planning and Strategy document. In summary, the steps in our plan were to assign roles, identify stakeholders, identify deliverables, produce a schedule with estimated dates and identify risks.

4.2 Planned Deliverables and Phase Gates

For each phase we determined a list of deliverables that satisfies the needs of ITS management and Software Engineering department requirements. For each phase gate, we estimated a date of completion.

Phase Gate Deliverables (Planned Date/Actual Date):

* Denotes deliverables not originally planned.

Initiate phase (19 December 2003)

- Planning and Strategy
- High Level Requirements
- Commercial Graphing Package Technical Proposal*

Analyze phase (30 January 2004 / 12 February 2004)

- Software Requirements and Specification
- Operational Testing Scenario
- Acceptance Test Plan

Design phase (25 February 2004 / 30 March 2004)

- High-Level Design
- Detailed-Level Design
- Technology Comparison Paper*

Develop phase (2 April 2004 / 3 May 2004)

- Source Code
- Operations Manual

Test phase (23 April 2004 / 14 May 2004)

- OTS Results Report
- ATP Results Report
- Issues Review

Implement phase (5 May 2004 / 20 May 2004)

- Deployment Plan

4.3 Major Plan Changes

There were two types of plan changes that we encountered during the project: change in schedule and change in deliverables. Changes in schedules typically occurred in preparation for phase gates and were presented to ITS management during the phase gate accompanied with reasons for the changes. Schedule changes in earlier phase gates would also affect the following phase gates. There were no significant changes to deliverables although there were additions for certain meetings.

In preparation for our first phase gate, we realized that the scope of our project would be too large if we had to write our own graphing package. Therefore we researched graphing packages and included a Commercial Graphing Package Technical Proposal at the Analyze phase gate. This document led to the purchase of RChart, a graphing pack-

age written in Java.

The Analyze phase gate was originally predicted to be on January 30th but was rescheduled to February 12th due to an extended requirements elicitation period. The schedule change was justified in order to produce a better requirement specification. During this phase gate, ITS management informed us of the need for an issues log. This became an additional deliverable at each phase gate and is a contiguous log of issues.

The design phase gate was scheduled for March 12th and was rescheduled to March 30th due to an extended design phase and scheduling conflicts between ITS management and project team. The extended design was due to spending an extra week researching technologies that could be used in the design. This resulted in a Technology Comparison Paper as an additional deliverable for the design phase gate. The develop phase was also extended to April 30th, an additional week to accommodate a more complex design at the cost of less time for the test phase.

The actual develop phase gate took place on May 3rd. A major change occurred as we approached the end of the develop phase when we determined the need to perform sufficient testing before delivering code for the develop phase gate. The develop phase gate became a progress check meeting while the deliverables in the develop phase got moved to the test phase gate. The purpose of this change was to ensure that the delivered code is properly tested and of sufficient quality.

Test phase gate was officially scheduled for May 14th and Implement phase gate was officially scheduled for May 20th, both deadlines that are dictated by the end of the quarter.

5 ARCHITECTURE AND DESIGN

5.1 Technology Comparison

Our first step in developing the architecture was to research technologies that could be used to implement the GRM system. The primary technologies we did research on were Web Services, Java RMI, Smart-Client, Java Server Pages (JSP), and Java Applets. We also included standalone Java program to ensure that we considered as many solutions as possible. We performed a compare and contrast of each of the technologies we felt could provide a good architecture for the GRM system, considering both advantages and disadvantages of each. The details of this research and comparison are further documented in the Technology Comparison Paper. We determined that Java RMI would be the underlying communication technology for the GRM system.

The advantages of using RMI are a smaller client program to be downloaded, less logic and processing in the client, and minimal parsing of data transferred between client and server.

5.2 Architecture of GRM System

Web Server - The web server for the GRM system provides an access point for all authorized users to the GRM client application. To access the client application, users have to login to the GRM and the system will spawn the

main client application on to the client machine. Apache has been chosen as the web server that will handle authentication and client application delivery.

GRM Server - The GRM server is the server that communicates with the client application. The communication between the client and the server will be implemented using Java's RMI technology. An RMI registry will be running on the server in order to provide naming services to the remote objects. The GRM server will also handle all the system's persistency and logic.

ITS DB - The ITS database provides all the necessary data that the GRM server requires in order to compute the correct Report data. The schema and entries of the database are provided by the ITS department. New data can be added to the database at any time.

GRM Web Applet - The web client of the GRM handles authentication for the GRM system, and initiation of the main thin-client application. The web client will communicate with the GRM web server through HTTP and is a combination of HTML and a simple Java applet. It is accessible by any machine that has a Java class library and common web browser installed.

GRM Thin Client - The thin-client of the GRM handles the presentation and user input layer for the GRM main system. It provides an interface for the user to communicate with the system. The application will be implemented in Java that utilizes RMI technology to communicate with the GRM server.

GRM Console - The GRM console is a text based interface that takes commands to execute Reports residing on the GRM server.

External Program/Script - Any external scripts or programs can feed text-based commands into the GRM console for automated execution.

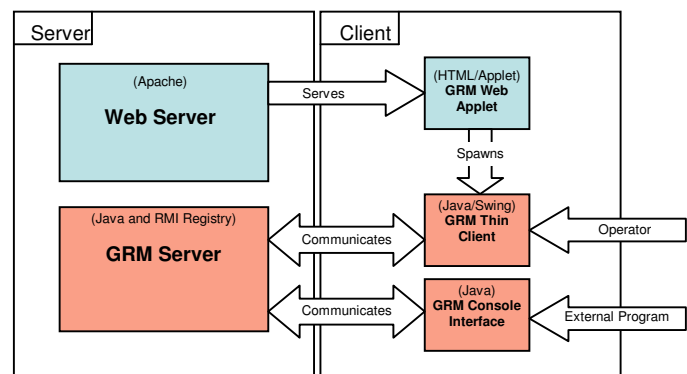


Figure 5.2.1 - Architecture

5.3 External Software Packages

During the analyze phase we decided that the scope of the project would become too large if we had to develop our own graphing package to generate the graphs mentioned in the High-Level Requirements. That led to research and the purchase of RChart, by Java4Less, which is a software package that can generate graphs. Details of this package can be found in the Commercial Graphing Package Technical Proposal.

Another external software package that was used in the GRM system is JHotDraw. This package is used as part of

the Graphical User Interface to allow the end-user to build Reports graphically by drag-and-dropping Elements and connecting the outputs and inputs between them.

5.4 Design Rationale

The thin-client interacting with a server allows the system to be distributed and multi-user since the thin-client is available online. The thin-client is used to build Reports and is also capable of calling execute and export on the server. The server provides central persistency, allowing multiple clients to share the same elements when building Reports. This also allows the use of a console client to execute and export the results of Reports that have already been created by the end-user using the thin-client. The console client can also be used by external systems, such as scripts, to execute Reports on a timely basis. An executed Report can then be exported to an external format, like JPEG, for displaying on ITS websites.

The purchase of RChart contained the scope of the project so we could focus on Report creation, server, graphical user interface and console interface. The use of JHotDraw provided an integral component for the Graphical User Interface.

5.5 Server Side Components

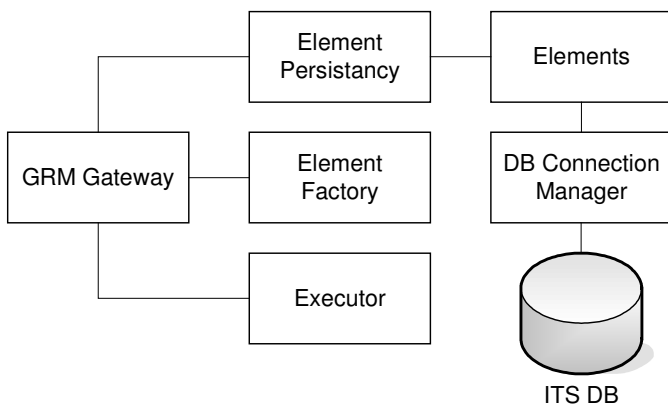


Figure 5.5.1 - Server Components

The GRMGateway component acts as the interface between the client and the server. The primary services that the server provides to the client are: create a base element, create a composite element, execute an element and modify an element. The client uses RMI to call methods on the server.

The Element Persistency component provides an Element database which is a container for all created Elements. This component provides methods to obtain created elements, modify the elements, and save. The save method serializes Elements in the Element database and stores it into a save file. When the server is first started, the Element database loads the save file and recreates each Element.

The Element Factory component provides a common interface for creating different types of Elements. When creating an Element, the type of Element is determined and the proper Element Factory generates the new Element from the parameters passed in from the client-side. Elements are designed to be used together and have the same interface. They differ in their attribute definition and execution

method.

The Executor and Exporter components are used to execute Reports and then export the generated results into a specified external format for example a JPEG of a graph.

The Database Connection Manager component provides database connectivity with the ITS network performance database. A singleton DB Connection Manager ensures a single point of entry to the ITS database as well as save resources by sharing the same connection between any processes that need a database connection.

5.6 Element Categorization and Types

There are two categories for Elements: base and composite. A base Element is a single Element of any type. A composite Element is one that contains more than one Element.

There are three types of Elements: Rover, Operation and Generator. Rovers have no input references and generate data for its output. Operations take input(s), perform actions on the data, and produce output. Generators take input(s) and generate graphical or tabular forms of the data. Composite Elements are also sorted as one of the three types, defined by the number of inputs and number of outputs. An example of a Rover is Database Rover that retrieves data from a database. An example of an Operation is Add Operation that takes two inputs and produces an output where input one's dataset is added with input two's dataset. An example of a Generator is RChart Generator that takes inputs and plots the data in a line graph.

5.7 Report Execution

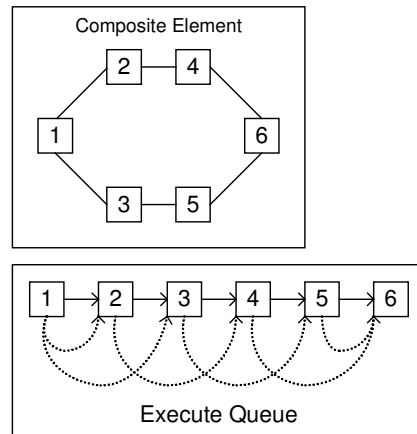


Figure 5.7.1 - Executing a Report

A Report is defined as a composite Element with no inputs and no outputs. Each element contains its own execution method. Within the Executor is a queue of Elements executed in sequence. The queue is built by the Executor when it parses the Composite Element's composite graph attribute. Each Element in the Composite is a node in the graph and the connections between the Elements are edges. The execution queue is built using a breadth first traversal. As each Element in the execute queue is executed, it retrieves input data from Element(s) in its input reference.

The total effect of executing a Report is retrieving data from the ITS database, processing it and then generating a graph that can be exported.

6 QUALITY ASSURANCE AND TESTING

6.1 Issues Log Tracking

An additional deliverable at each phase gate was the Issues Log. The purpose of this log is to keep track of concerns that occur during the project. Issues can range from small tasks that require a few minutes to accomplish or large quality defects that require significant effort to fix. These issues are categorized by significance, status and the team member responsible for following up on the issue.

Having an issues log allows our team to visualize our progress in each phase and plan our work schedules accordingly. The issues log provides documented concerns so that we do not let problems go unresolved.

6.2 Acceptance Test Plan

The Acceptance Test Plan was developed in unison with the SRS to ensure sufficient coverage of the different permutations of problems that may arise within the GRM system. The ATP contains test conditions that may occur for each use case. During the testing phase we perform each test case in the ATP to ensure that the GRM system satisfies the test conditions we determined during the analyze phase.

6.3 Operational Testing Scenarios

The OTS contains actual testing scripts that are based on the use cases. Each test script includes input data and description of possible outputs that denote if the test script passes or fails. These tests can be out-sourced to QA testers who will perform each test and provide a Pass/Fail value for GRM system testing.

7 DELIVERY AND PROVISIONS FOR SUPPORT

For the final phase, the primary deliverable is a Deployment Plan. This document outlines the procedure to compile and install the GRM system on ITS servers and the additional packages or software needed to run the GRM system.

Post project support for the system was a topic discussed during the Analyze and Design phase gates. It was determined that post project support for the system will be done by ITS.

8 REFLECTION

Interaction with our client was a smooth process for the duration of our project. Our point of contact at ITS replied to emails promptly and was also available to us on AOL Instant Messenger. Phase gates were all scheduled at least two weeks in advance. For the first two phases our point of contact met with us weekly for requirements elicitation. During design and implement phases, the ITS support team was responsive to our emails and requests.

Phase Gate meetings were very effective in keeping us on track as well as communicating project progress to our sponsors. Each phase gate meeting was an opportunity for us to interact with ITS management and gain experience in making presentations and having discussions with higher

management. The combination of phase gate presentations and issues log were an effective method for us to keep track of our schedule and progress.

Keeping an Issues Log, that documents a contiguous log of problems we encountered or tasks that need to be addressed, ensured that we did not forget to attend to those issues. At first we did not fully comprehend the use of the Issues Log. Therefore we did not use it effectively and it was always a topic at phase gate meetings with ITS management.

The Initiate, Analyze and Design phase gates were successful and the deliverables produced were satisfactory. During the Develop phase we started to realize that the scope of the project is larger than we originally predicted. During the earlier phases we tried to generalize workings of the GRM system into Elements. The specification of the Element led to a complex design and an even more complex implementation as we realized the limitations of our design. This added complexity to our system was not realized until we were deep into the Develop phase and the team worked over-time to refine the design.

Another unexpected issue was that we were discouraged from using a Java application server by our customer. This preference was not discovered until a week into design and caused the loss of a week in the schedule. Fortunately the Technology Comparison Paper provided alternatives and we were able to switch technologies on the fly.

9 ACKNOWLEDGEMENTS

RIT Information and Technology Services:

- Mark Kimble, ITS Systems Management
- Patrick Saeva, ITS Program Management
- Emilio DiLorenzo, ITS Director
- ITS Customer Support Staff

RIT Software Engineering Faculty Advisors:

- Dr. Jim Vallino
- Dr. Stephanie Ludi

10 REFERENCES

- [1] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Planning and Strategy," unpublished.
- [2] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Software Requirements and Specifications," unpublished.
- [3] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Operational Testing Scenario," unpublished.
- [4] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Acceptance Test Plan," unpublished.
- [5] A. Buehler, C. Chiou, J. Myers, and C. Tam, "High-Level Design," unpublished.
- [6] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Detailed-Level Design," unpublished.
- [7] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Technology Comparison Paper," unpublished.
- [8] A. Buehler, C. Chiou, J. Myers, and C. Tam, "Commercial Graphing Package Technical Proposal," unpublished.