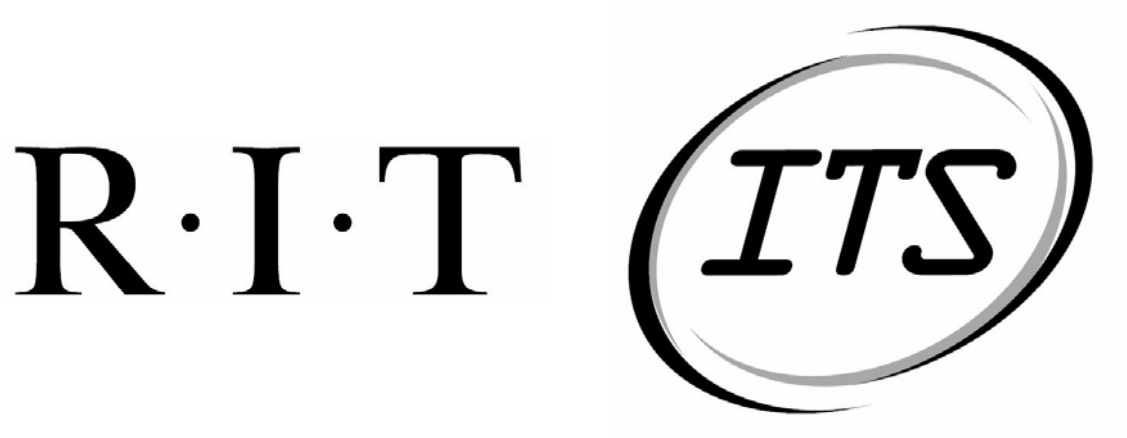


ITS Graphical Report Maker

High-Level Design

30 March 2004

Team JACT Software
RIT Software Engineering Department

Version 1.3.1

Revision History

Revision	Date	Author	Section	Comments/Changes
1.0.0	27 March 04	All	All	Initial Revision
1.1.0	28 March 04	Train	All	Additional Content
1.2.0	29 March 04	All	All	Additional Content
1.3.0	30 March 04	Myers	Requirements	Updated section with Persistency information
1.3.1	30 March 04	Train	All	Edit for consistency and mistakes

1. Project Overview

1.1 Purpose

The purpose of this document is to specify the high-level design for the ITS Graphical Report Maker (GRM). This document will act as an outline for implementation and discuss the design considerations.

1.2 Audience

This high-level design is intended to be used by members of the development team that will implement the functionality of the GRM. This document will also be used to communicate the high-level design and design considerations to the ITS staff members.

- Emilio DiLorenzo, ITS, Director of Technical Support Services
- Mark J. Kimble, ITS, System Management and Tools Technical Support Services
- Patrick Saeva, ITS, Program Manager
- Dr. James Vallino, Faculty Advisor
- Dr. Stephanie Ludi, Assistant Faculty Advisor
- Adam Buehler, Development Team
- Cesario Tam, Development Team
- John Myers, Development Team
- Cheng-Train Chiou, Development Team

1.3 Design Process

The high-level design was selected by deciding what aspects of the system were most important and then building an architecture around them. Several architectures were proposed: standalone system accessing a database, distributed system utilizing web

services technology, and distributed system utilizing the Java Remote Method Invocation (RMI) technology. The pros and cons of each architecture and technology were discussed in meetings. For each technology proposed we explored and researched on feasibility and capability. Web services technology would allow the system to be less coupled and more cohesive. Communication between client and server would utilize XML over SOAP using port 80. This technology would also provide ability for any interface to utilize the web services of the web server on the network. Java RMI technology would allow the system to be more coupled meaning the client is more dependent upon the server side. Communication between the client and server would be invocations through a proxy passing Java serialized objects as parameters. After research and proof of concepts, the team reached consensus to implement a distributed system that uses Java RMI technology.

2. Requirements

The overall requirements of this system remain unchanged from the “Analyze” phase gate deliverable of the Software Requirements Specification. In summary from that document, the main requirements are:

- To provide the ITS staff with a medium to generate graphical reports for upper management review and technical analysis.
- To provide the ITS staff the ability to generate graphical reports using the data from the provided database.
- To allow the ITS staff around-the-clock, online access to reports that have been prepared in advance.

2.1 Proposed Solution

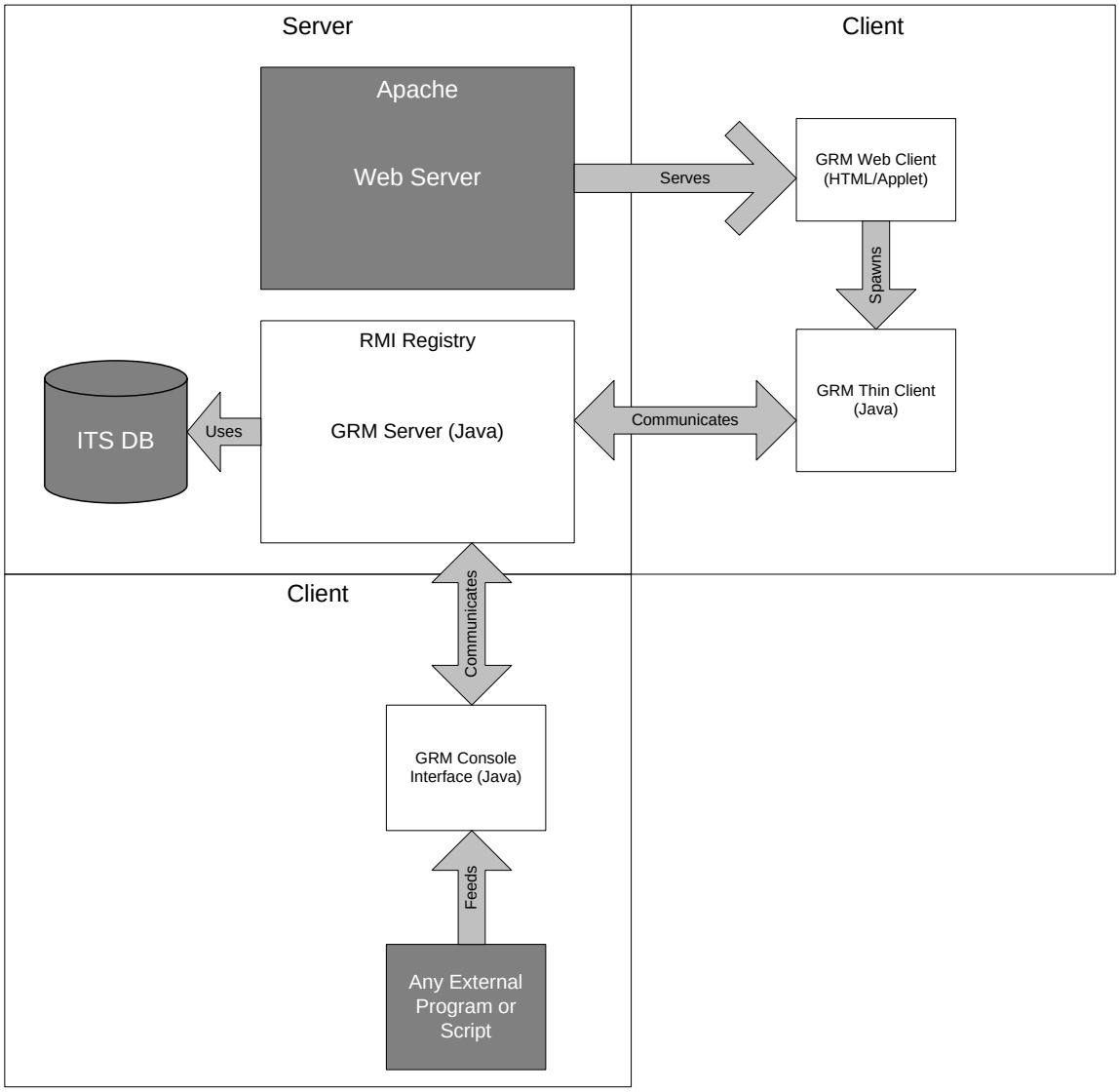
The proposed solution for the overall system is obvious when taken into the context of the requirements elicited during the analyze phase. To allow for clean interoperability between different platforms, Java is being utilized as the implementation medium. In addition, the client-server architecture is being achieved independent of a Tomcat or any other servlet-serving engine at the request of the customer. Therefore, through Java RMI, a thin-client is being downloaded to the client's computer and communication will occur through that. For information on this interaction, please refer to the detailed design document, which describes this interface.

2.2 Capacity Planning

Persistency within the system is being accomplished through Java Serialization of the Element objects. This type of serialization is very minimal, and containing the most data, a typical Element should not pass 1.5 kb in size. Therefore, if the persistent storage grew to a size of 100,000 Elements (a large number of equations), the GRM system would still only require about 150 MB of storage. Therefore, in terms of large volume capacity planning, allocating less than 1 GB of storage to the GRM system would be more than sufficient. It should be noted that the amount of space utilized by the database of the GRM system (outside of its scope) is significantly larger than what will be used by the equations and images.

3. Architecture

3.1 Design



Server:

- **Web Server**
  - The web server for the GRM system provides an access 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.

Client:

- **GRM Web Client**
  - 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 program can feed text-based commands into the GRM console for automated execution.

3.1.1 Version

Version 1.0 – Initial version of the system that has the basic J2EE architecture.

Version 1.1 – After meeting with technical staff from the ITS, the development team found out that there would be no application server residing on the machine. The current version utilizes Java's RMI technology which can avoid the need for an application server.

3.1.2 Server Roles

Apache – A web server that is needed to host a thin client of the GRM system. It will be installed on a Solaris 8 machine. The functionality of this server will not only handle the hosting of the thin client, it will also provide the only means of authentication.

RMI Registry – The server will have to run an RMI registry in order to utilize the technology. The registry provides the capability of locating the Java remote objects that are residing on it.

3.2 Access

The system will be accessible through a Java thin client. The client system will download and execute the Java thin client from the server side. The Java thin client will run on the client system and access the server side for functionality. The server also accesses another system for the mySQL database.

3.2 Hardware and Platform Requirements

Hardware Requirements:

- Sun Sparc (Test bed on Sparc 1 for backward compatibility)

Operating System:

- Solaris 8.0

Packages Installed:

- gcc-2.95.3
- binutils-2.11.2
- perl-5.6.1
- glib-1.2.6
- screen-3.9.5
- sudo-1.6.6
- tar-1.13.19
- top-3.5beta12
- lsof-4.49
- wget-1.8.1
- ncftp-3.0.1
- lynx-2.8.3
- ipf-3.4.31 --- http://cheops.anu.edu.au/~avalon/ip-filter.html
- ssh-3.2.5 -- ftp://ftp.ssh.com/pub/ssh
- sendmail-8.11.6

3.4 System Connectivity

The client will connect with the system using Java RMI technology so both client and server needs to reside on the RIT network and the necessary ports are available to both client and server.

4 Standards

4.1 Security Standards

4.3.1 Authorization and Logon

The system shall verify the username and password using ITS LDAP Authentication.

4.2 Disaster Recovery

Restart server program as directed in the operational manual and the system will return to last safe state.

5. Support

The following support documentation will be provided: Code, Design Document, Operations Manual, and Deployment Plan. The system code shall be documented according to the "Code Conventions for the Java Programming Language" available at http://java.sun.com/docs/codeconv/. The system shall be described by a "Design Document." The system shall be accompanied by an "Operations Manual" describing proper use of the system. The system shall be deployed using operations described in the "Deployment Plan."