Design and Implementation of Infinity Research Assistant

Nick Oliver and Yi Liu
Department of Electrical Engineering and Computer Science
South Dakota State University
Brookings, SD 57007
nicholas.oliver@jacks.sdstate.edu, yi.liu@sdstate.edu

Abstract

University research projects require large amounts of data storage, organization, and coordination. From tracking materials and equipment to managing personnel, documents, and budget, it is highly necessary to have good organization programs. There does not currently exist a customizable and flexible product for easily managing research projects. Infinity Research Assistant is designed to help a research group or a lab to efficiently manage research projects and disseminate the research results. The software allows the principle investigators to be able to use three modules, equipment management, materials management, and project management, to keep track of multiple projects and the associated personnel, budgets, and associated documents and manage inventory of labs, including materials and equipment, and also be able to define new modules to achieve the flexibility and expandability. This paper focuses on illustrating the architectural design of the Infinity Research Assistant. The paper highlights the use of both Client-Server pattern and Model View Controller pattern in the high level architectural design, and use of the design patterns Abstract Factory and Strategy Pattern in the medium level design to achieve the flexibility and expandability. This paper also addresses the successful implementation of the software and presents the user interface.

1 Introduction

University research projects require large amounts of data storage, organization, and coordination. From tracking materials and equipment to managing personnel, documents, and budget, it is highly necessary to have good organization programs. There does not currently exist a customizable and flexible product for easily managing research projects. This has led to the development of Infinity Research Assistant project management software.

Infinity Research Assistant (IRA) is designed to be fully customizable and flexible while still maintain easy usability. The IRA software allows the principle investigators to manage a number of different projects and labs and also serves as a centralized hub to display and track budgets, to store and download associated documents such as proposals, research reports, publications, and etc., to inventory and reserve lab materials and equipment, and to save and display important deadlines. In addition to these, IRA also allows the principle investigators to add customizable modules to fit any future needs. IRA's customizable interface allows the researcher to limit the information that the team members can access.

This paper focuses on illustrating the architectural design of the IRA and also briefly presents the implementation of the system. Section 2 gives a brief overview of the required features of IRA. Section 3 describes the architectural design of the system with the focuses on using two architectural patterns in the high level design and two design patterns in the medium design to support the flexibility and extensibility of the system. Section 4 highlights the tools used in the implementation, and demonstrates several user interfaces. Section 5 provides a conclusion to the paper.

2 Required Features

IRA is a software tool for helping a research group or a lab to manage and conduct research projects efficiently. IRA contains three main modules – equipment management, material management, and project management. It provides the ability of adding in user defined modules as well. These user defined modules is one of the main motivating factors in the need for a new software tool.

The equipment and materials management modules function as tools to track inventory and to also easily display what equipment is currently in use. These modules are associated with research projects, but with research labs. The materials management module allows the team members to see the amount remaining of materials and also allow them to update the amount used. The equipment management module functions in a similar manner. It displays the inventory of all equipment in each lab but also incorporates a system to reserve equipment and shows when pieces of equipment are in use.

The project management module serves as a centralized interface to all aspects of research projects. These include modules to keep a budget, upload important documents, save important dates and deadlines, and add custom modules.

Custom modules can be added by the lab or project managers as needed. The custom modules needs to be dynamically created with components that may be needed. An example of a custom module would be a Patent module. This is a module that may only be needed in one research project, the software will have to be able to create this module based on the user’s specifications at the time of addition.

Another aspect of flexibility is the ability to hide information from users. A research project contains information that is pertinent to one party only, or it may contain confidential information. The principle investigators or lab managers would like to be able to set permissions on each module, and sometimes sub-modules or categories, based on individual users and also on different user levels.

The user levels, administrator, power user, and public, are set to have different privileges of using the software. The administrator is the owner of the research project. They are able to see all modules of their own projects and also be able to assign permission to other users associated with their projects. The power users are able to see the modules that the administrator has given them access to and can update only certain parts of the modules, such as amount of material used. The power users usually are graduate research assistants. Public users are able to see only limited views of some of the modules, including documents and budgets. The purpose of the public level is to permit outside parties, such as donors or other university personnel, to see project progress or verify that the project is sticking to the budget and deadlines.

3 The Architectural Design

IRA is designed to use model-view-controller pattern and client-server pattern in the high level architecture and several software design patterns in the medium level, including abstract factory pattern and strategy pattern to support the flexibility and extendibility.

3.1 High Level Architectural Design

IRA uses two software architecture patterns, Model-View-Controller pattern [1] and Client-Server pattern[2], in the high level architectural design.

The decision to design this software using the Client-Server pattern came from the need to have a responsive and active user interface while still being able to perform tasks such as retrieving data from databases. Another important factor is the support of multiple users. The server will reside on a lab computer while researchers are able to access the software from many different user stations while still retrieving and viewing the same project information.

The Model-View-Controller pattern is used to separate the user interfaces from the data driven tasks of storing and manipulating the project information. This also provides a very dynamic user interface, which is important in the flexibility of both module addition and in user permissions.

Figure 1 shows the high-level architectural design of IRA with both the client-server architecture and also the model-view-controller architecture applied. Each client accesses the software through a view, the only client side interface. The view is used to get user input to send back to and display data that it retrieves from the model, which resides on the server. The controller communicates with the database to retrieve and manipulate the data before sending to the model, which in turn updates the view as the user requests.

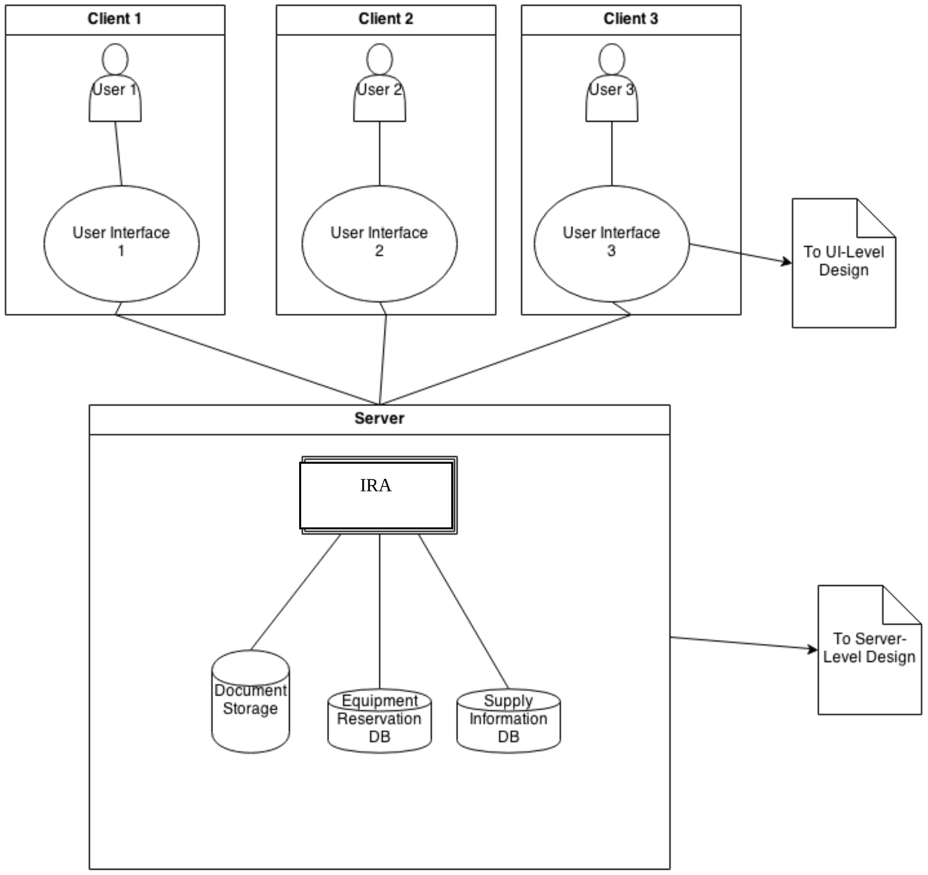


Figure 1: High Level Architectural Design

Each view is represented by a user interface while the models and controllers are represented by IRA on the server. The controllers interface with the database that contains the project information.

3.2 Medium Level Design

The medium level design architecture consists of several design patterns. The initial design included the abstract factory pattern and strategy patterns. The strategy pattern [3] is used for the flexibility of displaying the views depending on user permissions. The abstract factory pattern [3] and composite pattern [3] are used to design the creation of custom modules.

3.2.1 View Dispatcher Strategy

In order to provide a different view depending on the user's permission levels and access rights, the views are designed using the strategy pattern. The strategy pattern allows the software to decide which algorithm to use on run time. Figure 2 shows the class diagram for the designed view dispatch strategy. Using this class, the model would call the Dispatch method, which would receive the user's permissions from the controller and update the view accordingly.

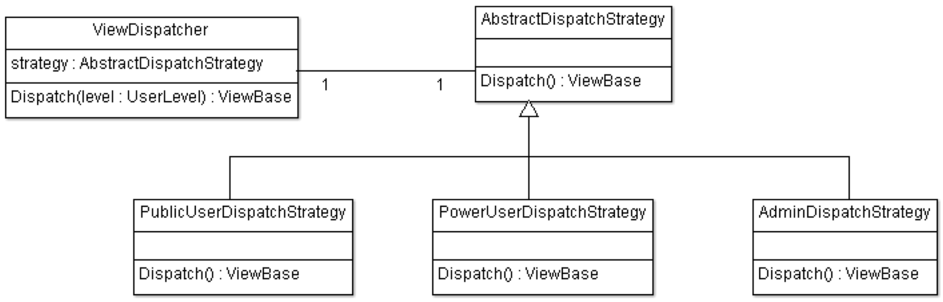


Figure 2: Abstract view dispatch strategy class diagram

The concrete strategy methods will be able to add or subtract from the view based on that user's access level. The Admin dispatch strategy would display the entire view, while the public user dispatch strategy would strip off components that they would not have access to.

3.2.2 Custom Module Factory

The abstract factory pattern provides a dynamic object creation that is well suited for IRA's custom module creation. Because the custom module is not already defined, there will be no existing view or model for it until the researcher wishes to create it. IRA will let the user pick which components they desire in the module and then the CustomModuleFactory will create the view associated with those components. Figure 3 shows the class diagram for the CustomModuleFactory. At the time of writing, only two module component bases where identified – grid based and calendar based. When the user selects to create a new custom module, the CustomModuleFactory's createModule method will be called. Depending on which type of module the user wishes to create, the corresponding concrete factory method will be called. These will then instantiate a new custom module object. These custom modules objects will contain views and models associated with the newly created model.

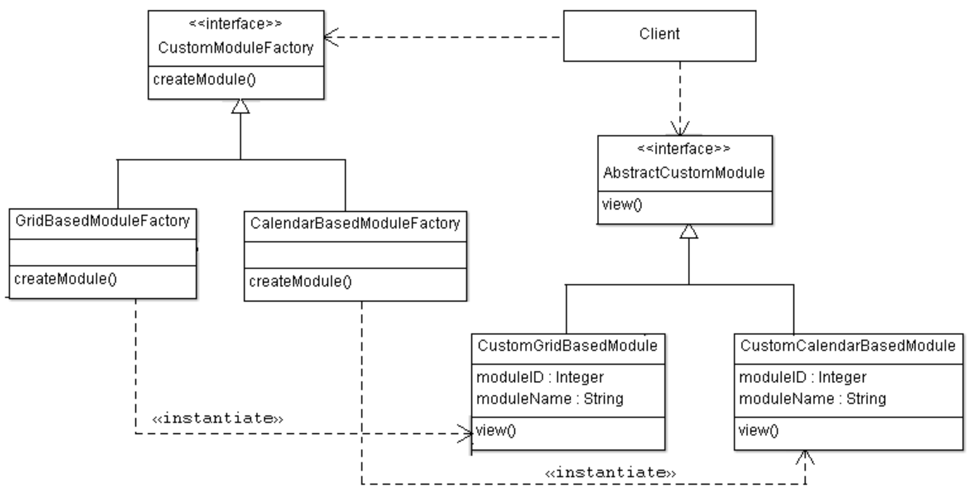


Figure 3: Custom module factory

4 Implementation

4.1 Tools used in implementation

IRA is developed using Microsoft Visual Studio and written in C#, Html, JavaScript, and CSS using Microsoft's ASP.NET framework for model-view-controller. ASP.NET provided a convenient solution for developing IRA using the model-view-controller pattern by providing the necessary functions for interaction between the models, views, and controllers. Entity framework with SQLServer is also used to interact with the database.

4.2 Implementation using design patterns

Because of ASP.NET's convenient methods of interfacing the views with the models, the AbstractViewDispatcher as mentioned in section 3.2.1 is unnecessary. Instead, each module's model implemented the IUserPermission interface as seen in Figure 4 which simply held the current user's permissions for that module or submodule.

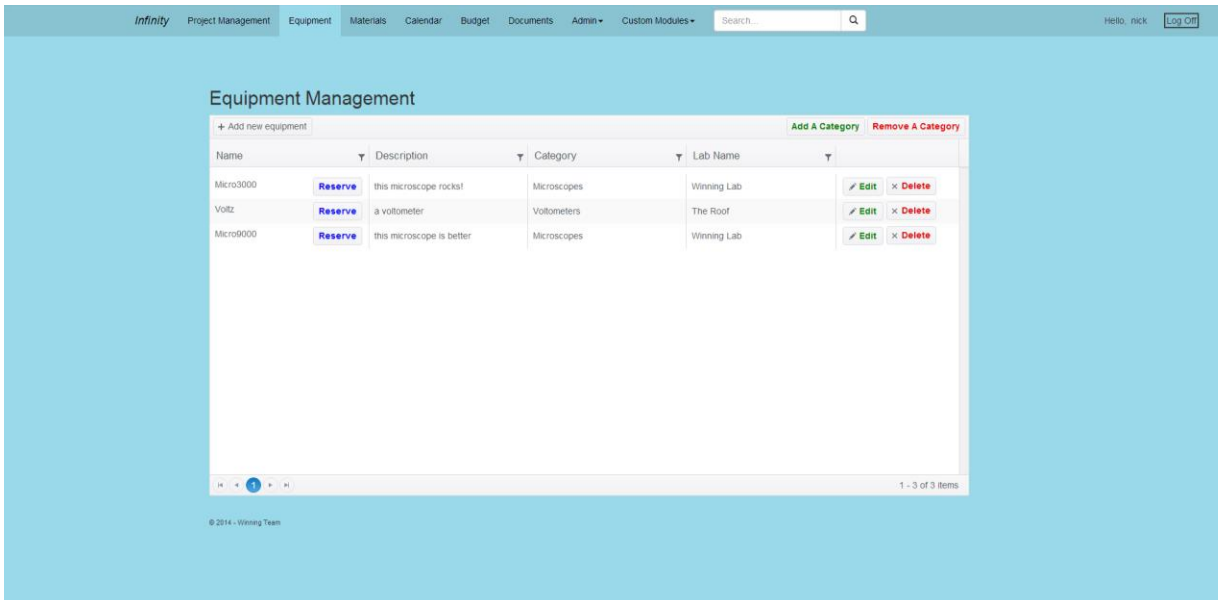
```
public interface IUserPermission
{
    PermissionEnum UserPermission { get; set; }
}
```

Figure 4: IUserPermission interface

Although the AbstractViewDispatcher is unnecessary, each module still needs to be able to determine the view based on the user's permissions. Within the controller, before updating a model, the software can retrieve all of the user's permissions for that module from the database. This is done using another class that functioned similarly to a strategy pattern, while not exactly implementing this pattern. This class is called based on the module, instead of the user permission, as originally designed. Passing these permissions to the module's model which implemented the IUserPermission interface, the software is able to dynamically modify the view based on what the user had permission to access.

4.3 User Interface

Figures 5 and 6 show the required modules Equipment and Materials, respectively. These are shown with full user permissions. The administrator can change the access permissions on the categories of these materials so that a user may not see any materials in the Top Secret Experiments category. In addition to changing the permissions on categories, Figure 7 shows the view when a user does not have permission to the Budget module. The IUserPermissions updates the model and tells the view that the user does not have permission to the Budget module and dynamically hides the Budget module.

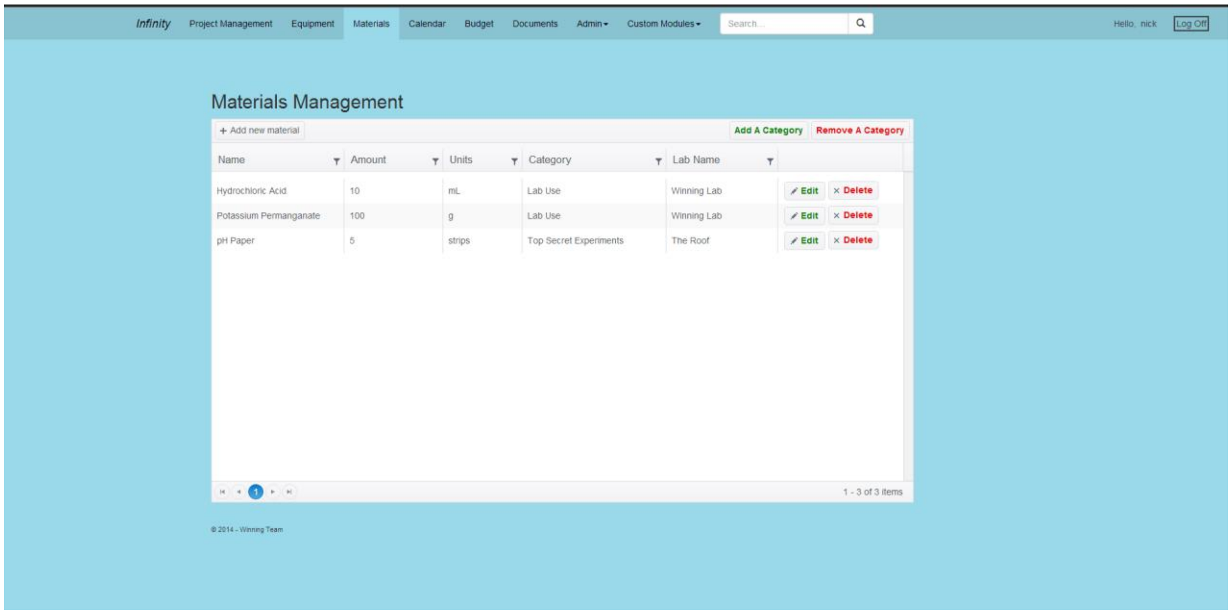


Name		Description	Category	Lab Name	
Micro3000	Reserve	this microscope rocks!	Microscopes	Winning Lab	Edit Delete
Voltz	Reserve	a voltometer	Voltometers	The Roof	Edit Delete
Micro9000	Reserve	this microscope is better	Microscopes	Winning Lab	Edit Delete

1 - 3 of 3 items

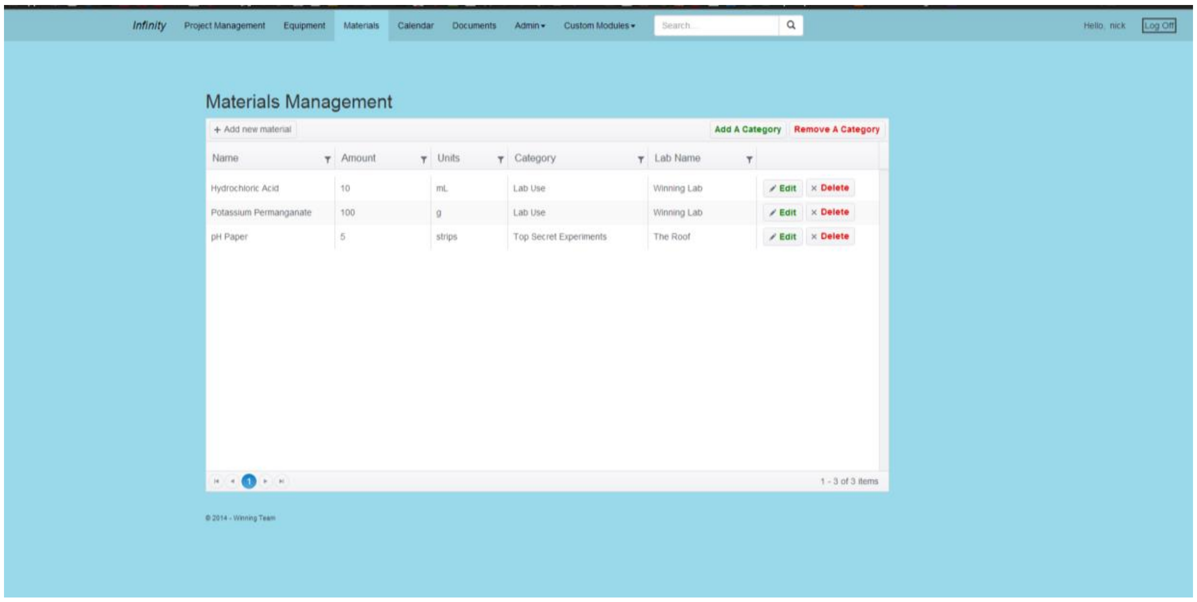
© 2014 - Winning Team

Figure 5: Equipment Management Module



Name	Amount	Units	Category	Lab Name
Hydrochloric Acid	10	mL	Lab Use	Winning Lab
Potassium Permanganate	100	g	Lab Use	Winning Lab
pH Paper	5	strips	Top Secret Experiments	The Roof

Figure 6: Materials Management Module



Name	Amount	Units	Category	Lab Name
Hydrochloric Acid	10	mL	Lab Use	Winning Lab
Potassium Permanganate	100	g	Lab Use	Winning Lab
pH Paper	5	strips	Top Secret Experiments	The Roof

Figure 7: User Interface With Budget Module Hidden

5 Conclusion

Infinity Research Assistant provides flexible and comprehensive control for project management. IRA has the potential to be very useful within the department, the university, and potentially further. Using the architectural and design patterns, the software is designed to react dynamically to however the user wished, giving them the ability to modify what other users can access and also by being able to create new custom modules in addition to the required modules.

References

[1] Buschman, F., Meunier, R., Rohnert H., Sommerlad, P., and Stal, M., Pattern-Oriented Software Architecture Vol -: A System of Patterns. Wiley, 1996.

[2] Qian, K., Fu, X., Tao, L., Xu, C., and Diaz-Herrera, J. Software Architecture and Design Illuminated. Jones and Bartlett Publishers, 2008.

[3] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995. ISBN: 0-201-63361-2.