Aleen Co.

网络搭建模拟器
Software Architecture Document

Version 0.1

December 26, 2014

Revision History

Date	Version	Description	Author
December 31, 2014	0.1	这文档用来描述该系统的具体架构	Aleen

Table of Contents

Software Architecture Document

1. Introduction

This document provides a high level overview of the evolving technical architecture for the 网络搭建模拟器, and it also provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases.

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This Software Architecture Document will be applied to design team, and the job after modelling will be influenced by this document.

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Architecture Document**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

The 网络搭配模拟器 Technical Architecture (TA) is represented using a UML model with four elements, including Requirement Model, Analysis Model, Design Model and Implementation Model, which combines with 4+1 views.

1. Requirement Model: I use Use-Case Diagram to implement this model to grasp the requirement of clients. This model combines with Use-Case View in 4+1 views.
2. Analysis Model: I use Class Diagram to implement it with three types of class: <entity>, <control> and <boundary>. This model combines with Logical View in 4+1 views.
3. Design Model: In this model, we can see how the software system built by different classes. This model combines with Logical View and Process View in 4+1 views. Logical View consists of three packages: Model, View and Controller (MVC). In this view, you will see how the system exactly works in details.
4. Implement Model: Finally, the implement model will give you all the information about how to realize this system. This model combines with Implementation View and Deployment View.

2. Architectural Representation

The UML specification of the system has been divided into the following four packages:

1. Requirement Model: Describes the actors and use cases for the system, this package presents the needs of the user. In each use case, activity diagram will be shown to describe the main process of activities.
2. Analysis Model: Describes the key classes that make up the whole system. For example the boundary class 登录界面 is used to provide a login window for user to log in the system, while the entity class 资源数据库 is used to store all the information of this system.
3. Design Model: Describes the main design of this system, like how it works with different classes, and

how the message conveyed between processes.

4. Implement Model: Describe the implementation of the system. In this model, preliminary code will be provided by coder.

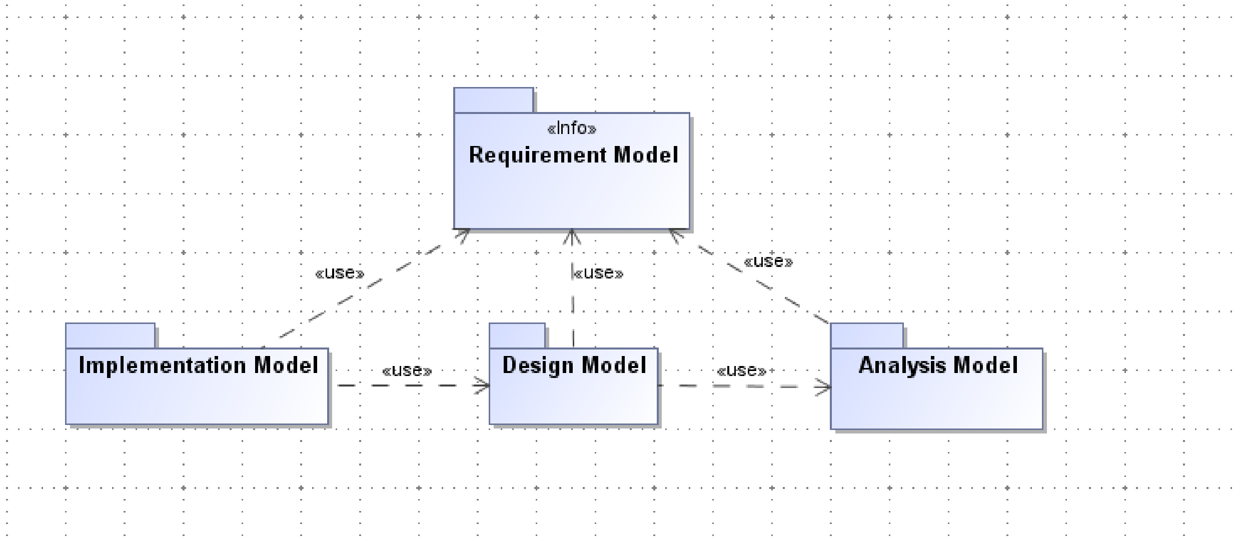

Diagram 1 The relationships of 4 models

Note, this document does not discuss the implementation model, the purpose of this modelling is just to learn how to use UML to build a system.

3. Architectural Goals and Constraints

Goals:

1. 该网络搭配模拟器是想实现网络搭配的模拟，并提供客户连通性检查的功能。相对的，模拟器同样提供维护资源数据库的功能给系统管理员。
2. 在网络搭配的时候，系统将提供图形化界面操作给用户。在底层，图形将被系统解析成代码。连通性的检查将通过代码的词法分析、语法分析和语义规则分析来实现。

Constraints:

None

4. Use-Case View

This view presents the users perception of the functionality provided by portal implementations of the 网络搭建模拟器.

In this process of modelling, I have used requirement model to stand for this view.

4.1 Actor

4.2 Use Cases

Describe Use Cases of 3 Actors in Use-Case Diagram.

4.2.1 用户(User)

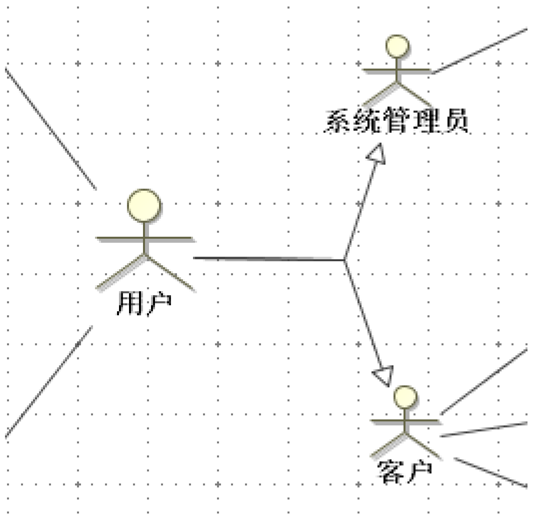

Diagram 2   Use Case of User

4.2.2   系统管理员(Administrator)

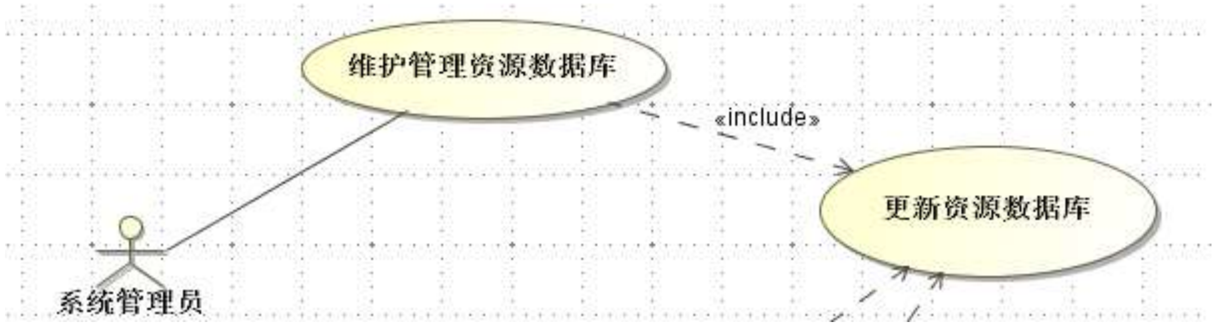

Diagram 3   Use Cases of Administrator

4.2.3   客户(Customer)

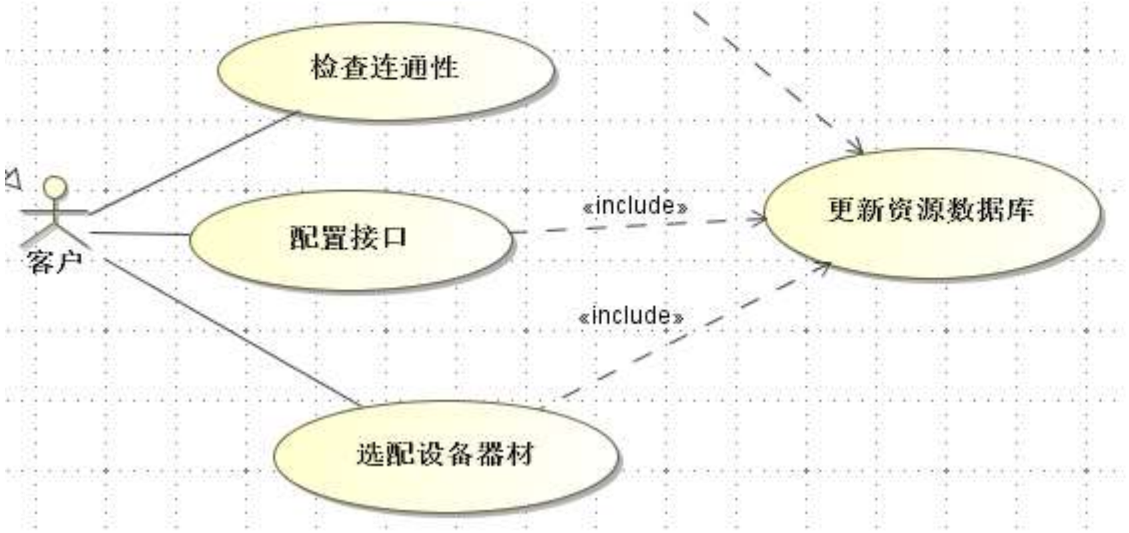

Diagram 4   Use Cases of Customer

5. Logical View

This view has described the architecturally significant parts of the design model. So it has been set into parts of design model.

5.1 Overview

Logical View is one part of the Design Model, and the other one is Process View, which has mainly described the communication between different processes.
Under Logical View, we can see there are two packages: Dynamic Structure and Static structure. They are two domains of the architecture of the system.

5.2 Architecturally Significant Design Packages

5.2.1 Dynamic Structure

This package is mainly used to describe how the system behavior dynamically towards users. Because there are two types of users, 系统管理员(Administrator) and 客户(Customer), so I have designed two sequence diagrams and two state machine diagrams to describe two types of users respectively.

5.2.1.1 Administrator

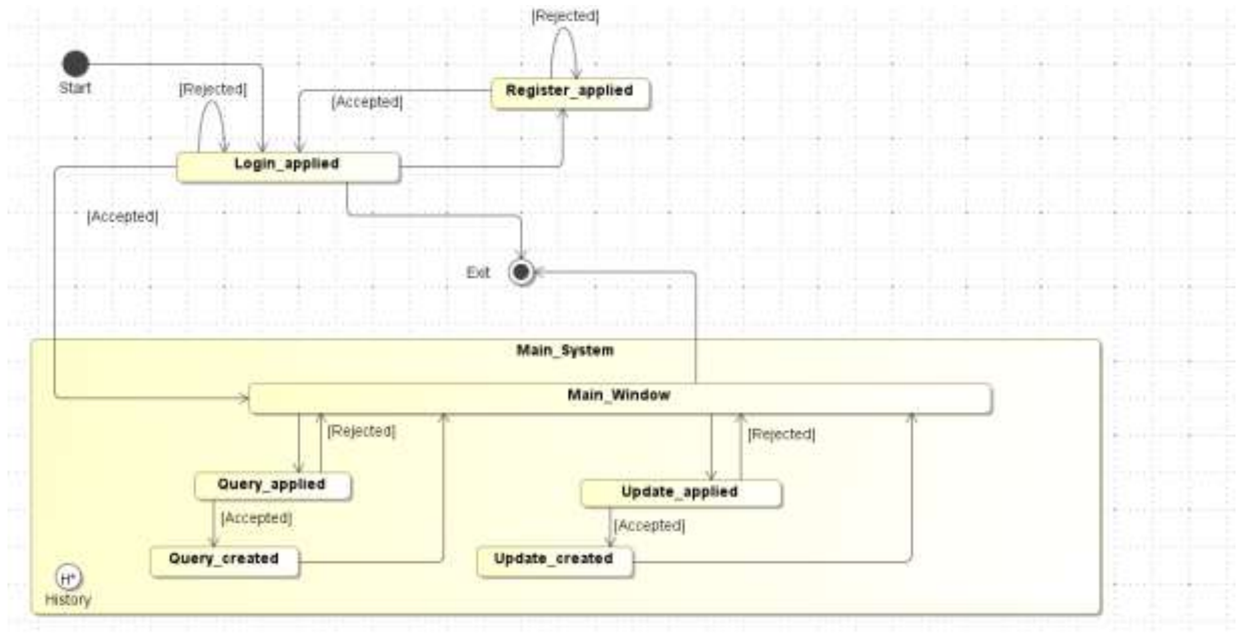

Diagram 5 State Machine Diagram of Administrator

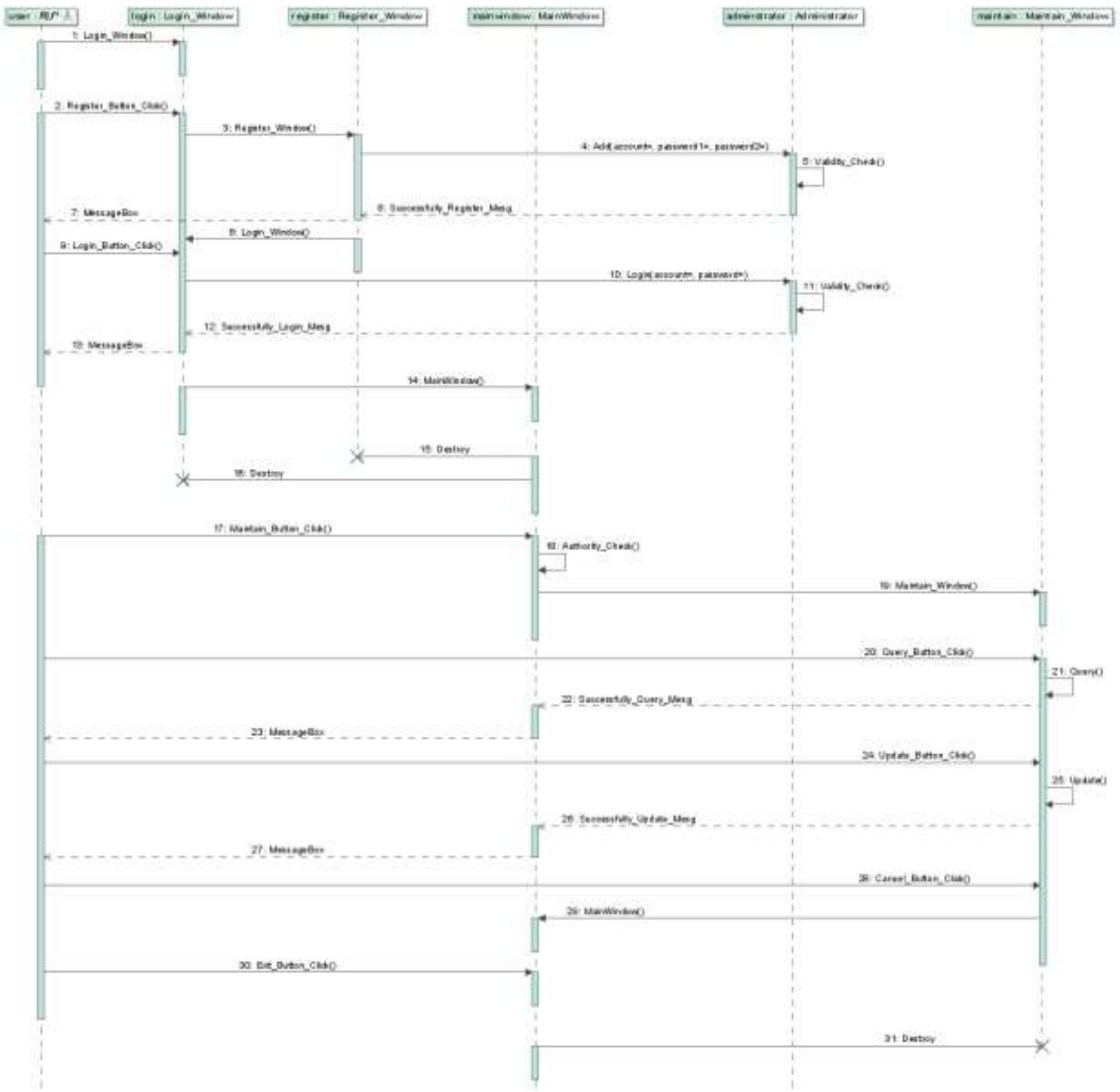

Diagram 6 Sequence Diagram of Administrator

5.2.1.2 Customer

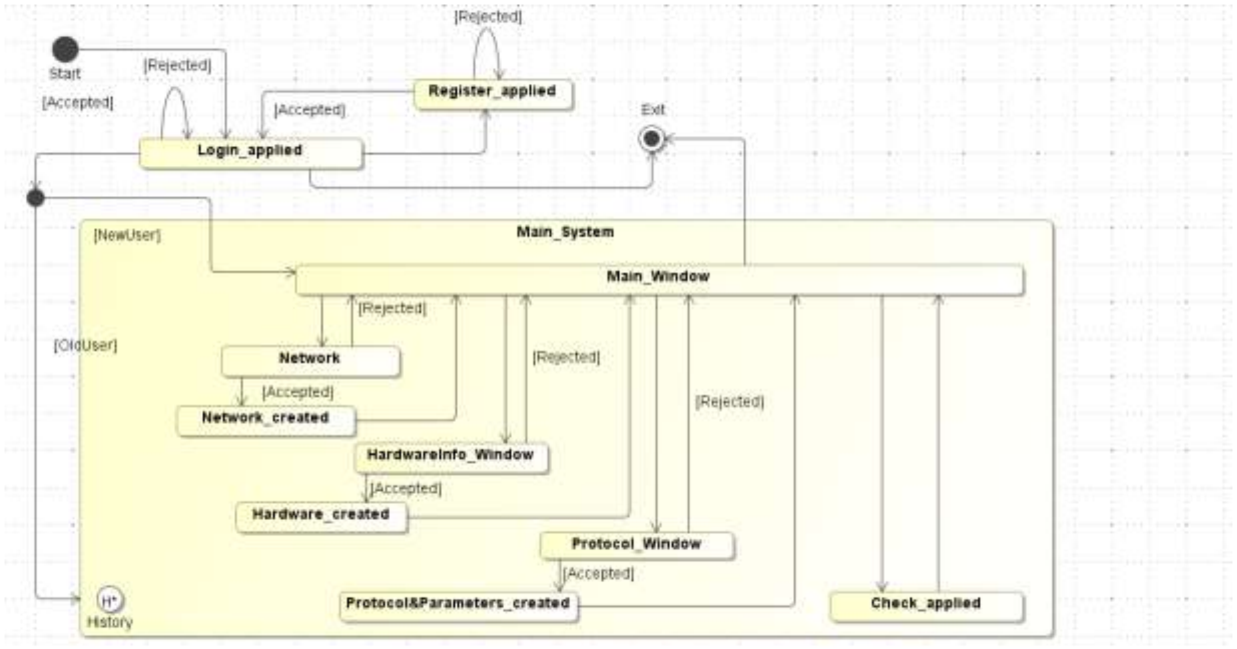

Diagram 7 State Machine Diagram of Customer

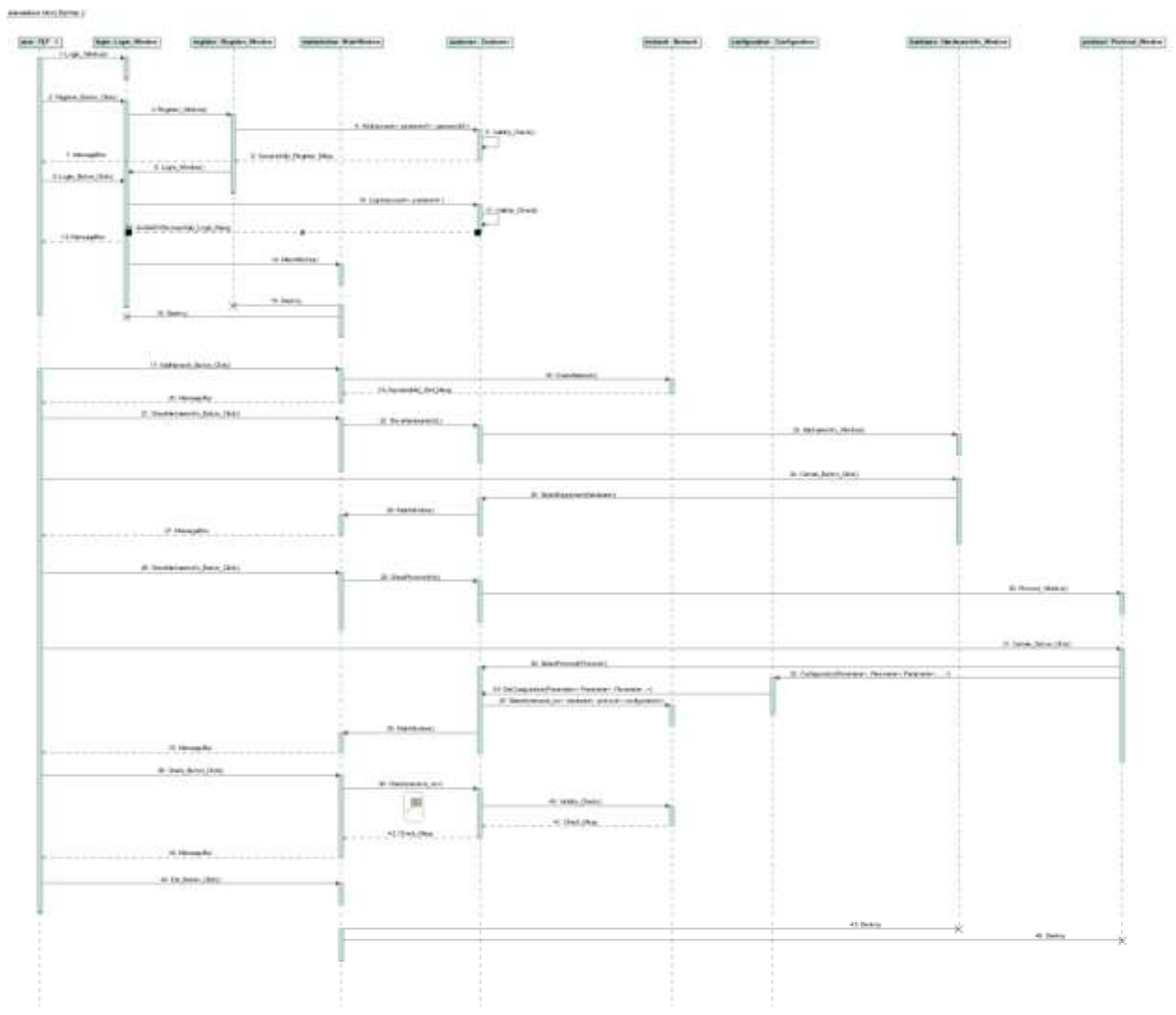

Diagram 8   Sequence Diagram of Customer

5.2.2 Static Structure

This package is mainly used to describe that the system consists of Model-View-Controller Model (MVC). In a macroscopic view, we can see the system consists of three parts, Model, View and Controller. The relationship between them has been shown in the following diagram.

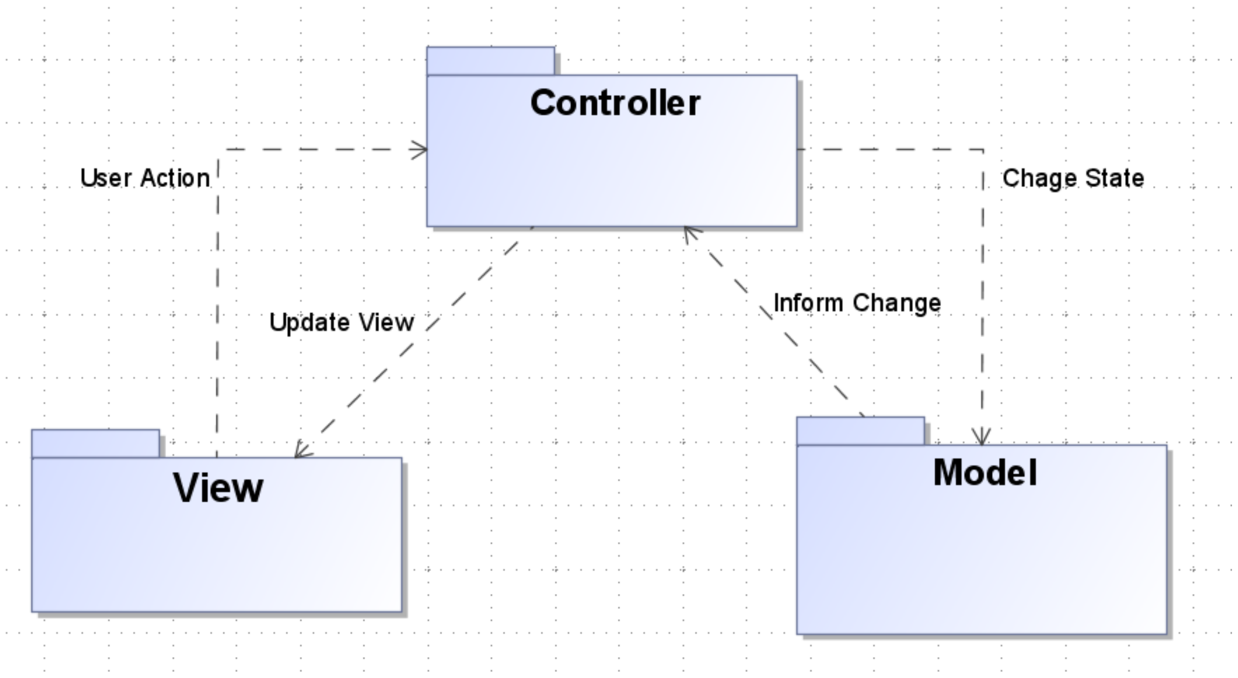

Diagram 9 Relationships between Model, View & Controller

From the **Diagram 9**, relationships between model, view and controller has been described clearly. View stands for the display of the system, and in this package, there are 8 classes. They are respectively MainWindow, Login_Window, Maintain_Window, Register_Window, HardwareInfo_Window, Protocol_Window, Window and System.Window.
Because I would like to design this system with Windows Presentation Foundation (WPF), so those windows should be generalized towards a three-party library, Window. According to the document of Microsoft Co., Window can be generalized towards System.Window. It's because I used a three party library that I have applied a stereotype <<Utility>> to Window and System.Window.

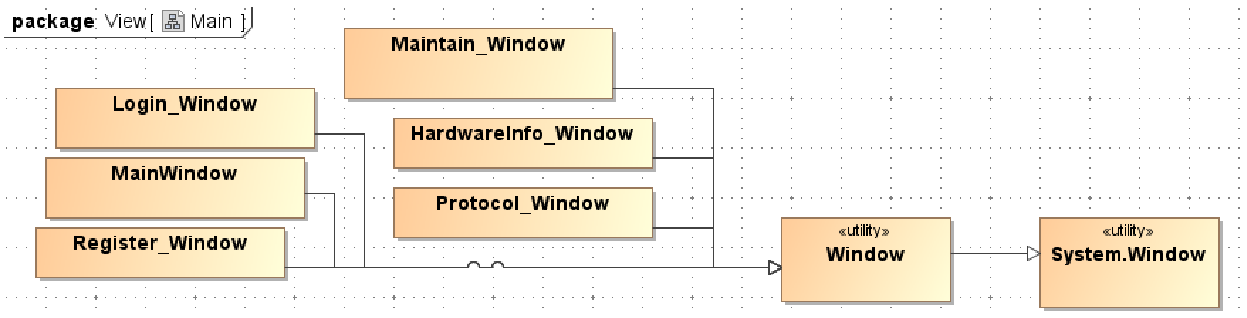

Diagram 10 Class Diagram of the View Package

MainWindow is the main view of the system. In this window, different buttons has been designed according to different functions. I would like to give you an introduction of how MVC works with an example. If the customer wants to create a network, he or she can click the AddNetwork_Button to trigger an event. For the system, it will handle the event, and give customer's action to the Controller.

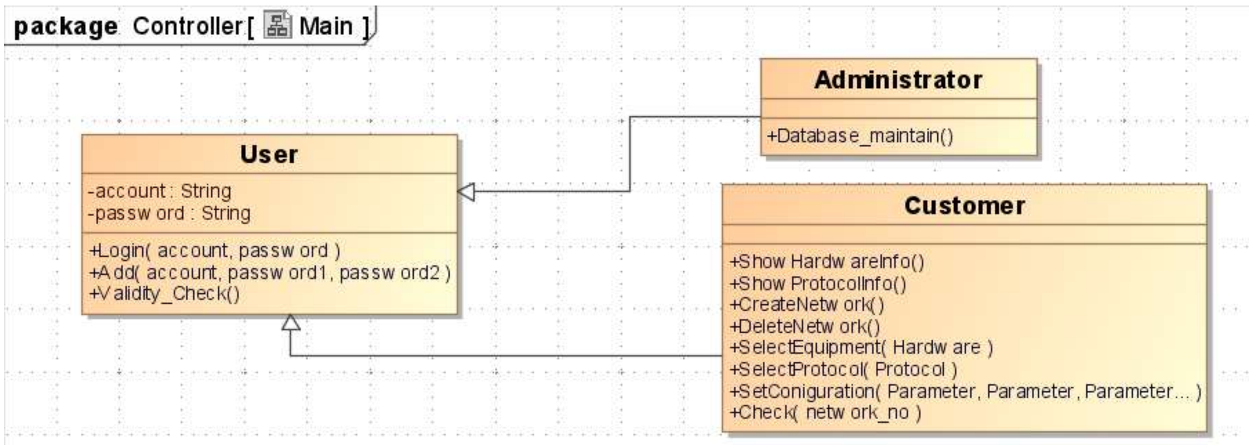
package Controller[Main]
Administrator
+Database_maintain()
User
-account : String
-passw ord : String
+Login(account, passw ord)
+Add(account, passw ord1, passw ord2)
+Validity_Check()
Customer
+Show Hardw areInfo()
+Show ProtocolInfo()
+CreateNetw ork()
+DeleteNetw ork()
+SelectEquipment(Hardw are)
+SelectProtocol(Protocol)
+SetConiguration(Parameter, Parameter, Parameter...)
+Check(netw ork_no)

Diagram 11 Class Diagram of the Controller Package

When the controller get the action from the View, it will call a function CreateNetwork() of Customer Class. When the function CreateNetwork() has been called, the Controller Package will give a signal to the Model Package, to change a state.

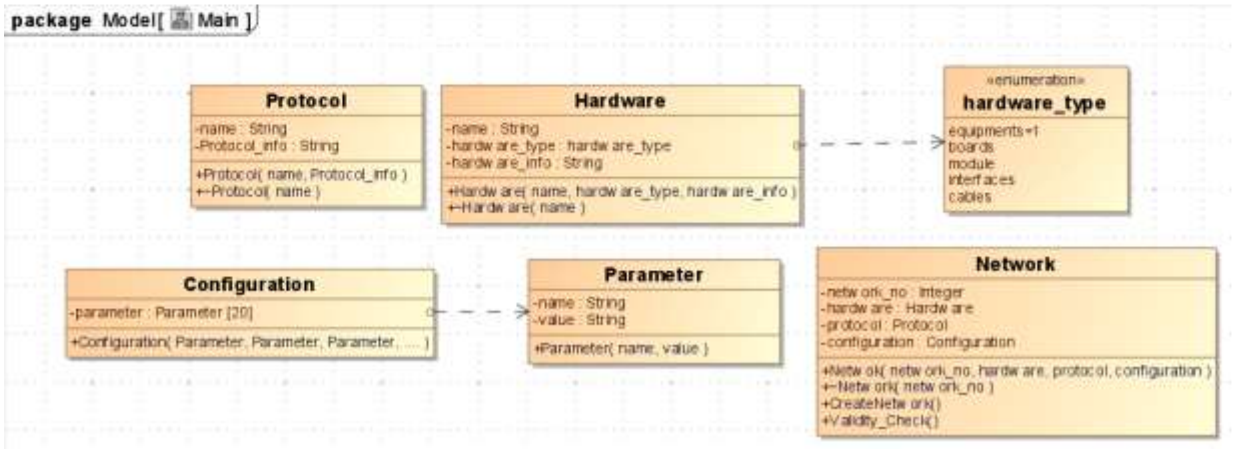

Diagram 12 Class Diagram of the Model Package

When the Model Package receives the signal from the Controller Package, it will call a function CreateNetwork() from the class Network. After successfully creating, the Model Package will inform the chart and send another signal to the Controller Package. The Controller Package will update view at the moment that it received the signal, sending a MessageBox, writing "Successfully create a network!", to the monitor. From the example, we can see that the function of this system has been designed according to the MVC model.

6. Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

7. Deployment View

[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the ***Process View*** *onto the physical nodes.]*

8. Implementation View

[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

8.1 Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers.]

8.2 Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

9. Data View (optional)

This view has a description of the persistent data storage of the system. In this system, data will be stored in the database with several tables.

User (Account, Password, Authority)

Account (varchar(50)): store the username of Users.
Password (varchar(50)): store the password of the user’s account.
Authority (binary(1)): 0 for administrator and 1 for costumer.

Hardware (Name, Hardware_no, Hardware_type, Hardware_info)

Name (varchar(50)): mark the name of the hardware.

Hardware_no (nchar(50)): because different customers will choose the same hardware, in order to distinguish the same hardware, there is a Hardware_no.

Hardware_type (binary(3)): mark the type of the hardware. There are totally 5 types of hardware, 1 for equipment, 2 for boards, 3 for modules, 4 for interfaces, 5 for cables.

Hardware_info (varchar(100)): have a description of the hardware in detail.

Protocol (Name, Protocol_no, Protocol_info)

Name (varchar(50)): mark the name of the protocol.

Protocol_no (nchar(50)): because different customers will choose the same protocol, so we have a Protocol_no to distinguish it.

Protocol_info (varchar(100)): have a description of the protocol in detail.

Parameter (Parameter_no, Name, Value)

Parameter_no (nchar(50)): mark the identification of a parameter.
Name (varchar(50)): the name of the parameter.
Value (numeric(10,2)): the value of the parameter.

Configuration (Configuration_no, Parameter_no)

Configuration_no (nchar(50)): mark the identification of a configuration from the customer.
Parameter_no (nchar(50)): a reference key of Parameter table, which is the identification of a parameter.

Network (Network_no, Hardware_no, Protocol_no, Configuration_no)

Network_no (nchar(50)): mark the identification of a network created by the customer.
Hardware_no (nchar(50)): a reference key of Hardware table.
Protocol_no (nchar(50)): a reference key of Protocol table.
Configuration_no (nchar(50)): a reference key of Configuration table.

10. Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

11. Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]