CS 445 / ECE 451/ CS 645 / SE 463

Deliverable Five

Software Requirements Specification

Submitted by:	Group 11
Group Members:	Adam Nace, amnace, 20127555 Brett Lounsbury, blounsbu, 20124604 Colin Rhodes, cprhodes, 20130560 Jesse Bishop, jbishop, 20129787 Steven Robertson, s2robert, 20141286
Submission Date:	27 November 2006

Contents

List of Use Cases

List of Functions

List of Non Functional Requirement Models

List of State Machines

List of Graphical User Interfaces

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This software requirements specification describes a Voice over IP (VoIP) style telephone communication network. The intended readership of this document comprises the customers and the system developers (including designers, testers, and maintenance programmers).

This specification does not go into detail about general telephone network concepts; the reader is assumed to have a basic knowledge of telephone networks. Additionally, this document uses UML diagrams in order to effectively convey specification information, so the reader should be familiar with reading UML diagrams.

1.2 Scope

This specification document describes a software system that supports a VoIP telephone switching system called FooFone. The software enables the customer to assign a unique four-digit extension to any VoIP-enabled telephone connected to the system. The telephone can be remotely activated or deactivated according to the customer's needs.

The specification defines two types of users: administrators and users. Each administrator is uniquely identified by a name and password, and is able to control all aspects of the telephone system. Each user is identified by an account name and has zero or more associated telephone accounts, each of which has one and only one telephone. A telephone must either be associated with a phone account and user account or be indicated as an emergency number, which is distinguished by a reserved extension and a lack of charges for calls placed to that extension.

Once assigned and activated, a telephone may be used to place or receive calls to other connected telephones within the structure of usage policies that are defined by an administrator on a per-telephone basis. Additionally, users may edit a list of extensions from which to block incoming calls to a telephone associated with their phone account. Each connected call will have an associated charge based on the duration of the call according to some billing plan defined by an administrator. These billing plans may contain discount periods which are dependent on the date and time a call is connected. These accumulated charges are collected and included as part of a monthly bill which is generated for each telephone account.

The software will periodically test telephones to ensure they are connected and functioning, and will disable the telephone's connection and subsequently inform administrators if a problem is discovered.

The software must handle concurrent processing of multiple calls, user and telephone account administration, telephone testing, and bill generation. The software does not support connections to telephones outside the system, deliver generated bills to a user, or provide a mechanism for the user to directly pay bills; this functionality is outside the scope of the system.

1.3 Definitions, Acronyms, Abbreviations

- Administrator
  A user of the system who has an associated administrator account and is not associated with any phone accounts.
- Phone account
  A collection of data about one of a customer's VoIP telephones. A phone account includes the phone's extension, its state, and any billing information related to that phone.
- User account
  A collection of data about a customer in the system. A user account includes the customer's name, address, and an external phone number that they can be contacted at. One or more phone accounts will be associated with a user account.
- VoIP
  Voice over internet protocol - a standard for telephones that communicate using the internet protocol.

1.4 References

This document cites requirements specified by the *Overview of the Course project*, available from http://www.student.cs.uwaterloo.ca/~cs445/Fall2006/Project/se1-2-3-project-overview.pdf. These requirements are referred to in the document as *R##*.

Telephone hardware interface methods were derived from the *Hardware Interface Specification*, available from http://www.student.cs.uwaterloo.ca/~cs445/Fall2006/Project/se1-2-3-hw-interface.pdf.

Additional information and requirements were sourced from the seven customer meetings, as described below. The minutes from these meetings are referred to in the document as *M#-##*.

- M1 - held 09/22/06. Minutes are included as Appendix B-1.
- M2 - held 09/29/06. Minutes are included as Appendix B-2.
- M3 - held 10/06/06. Minutes are included as Appendix B-3.
- M4 - held 10/20/06. Minutes are included as Appendix B-4.
- M5 - held 11/03/06. Minutes are included as Appendix B-5.
- M6 - held 11/09/06. Minutes are included as Appendix B-6.
- M7 - held 11/23/06. Minutes are included as Appendix B-7.

1.5 Overview

The remainder of this document is organized as follows. Section 2 contains an overall description of the VoIP telephone switching system software requirements, including the system's product perspective, product features, user characteristics, constraints, and assumptions. Section 3 identifies specific requirements, including external interfaces, functional requirements, and quality attributes. The document concludes with a list of glossary terms included as Appendix A.

Appendix B consists of the approved minutes of customer meetings. These minutes do not constitute additional requirements; all requirements associated with these minutes have been incorporated into the requirements specified in Section 3.

2 Overall Description

2.1 Product Perspective

A context diagram for the VoIP system is presented below in Figure 1. The VoIP phone system consists of a number of phones that are connected to a server over a network. No connections to external systems are necessary. There may be other devices connected to the network; the phone system will not allow itself to be disrupted by these devices.

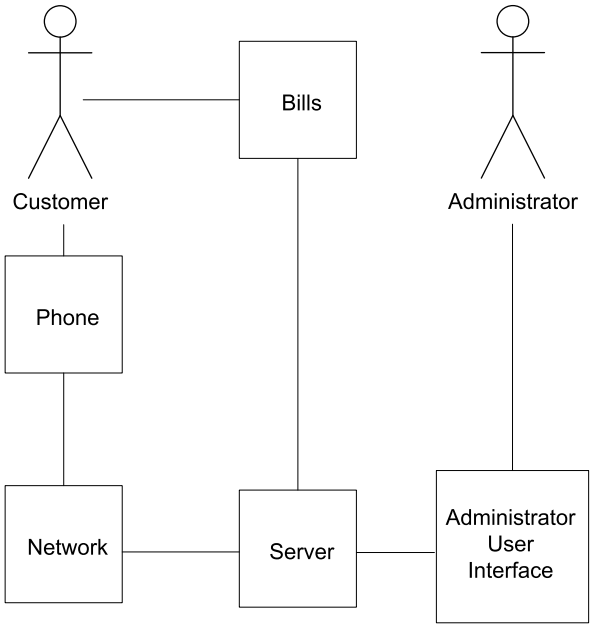


Figure 1: System Context Diagram

Communication Interface

The server communicates with the phones over a communication network. The protocol used is proprietary. The administrator interface will communicate with the server over either the same network or another network.

Software Interface

The messages sent via the communication network are specific to the phones being used. An XML interface to the messages can be found in the se1-2-3-hw-interface.pdf file on the SE1 course website.

Hardware Interface

The phone services will monitor and control Nortel i2004 IPPhones.

User Interface

There will be a graphical user interface to the services provided by the server for use by administrators. The interface should be intuitive enough that administrators are able to learn the system

in under a week.

2.2 Product Features

The system's features are described in the list of usecases in Table 1. As per the SE1 project requirements, all exceptions and non-essential alternate flows have been removed. The system also includes features described by the use cases in Table 2, but they have not been included because they do not have associated functions (they either did not change the domain model, they were very similar to other functions, or were deemed to be of low importance).

UC#	Description
01	A customer must be able to use his/her phone(s) to be able to establish calls with other customers.
03	A customer must be able to block incoming calls from other customers if he/she has paid for this ability.
04	Administrators must have identities in the system, and must be authenticated.
06	Administrators must have password protected accounts, and must be able to change those passwords.
07	Administrators must be able to create new accounts for other administrators.
08	Administrator accounts must be able to be deleted.
12	Customers must have accounts in the system to keep track of their contact information.
13	Customers must have separate accounts for their phones to keep track of their billing information.
14	Customer accounts must be able to be deleted when a customer chooses to terminate their services.
16	Administrators must be able to create and update billing plans to specify how to bill calls.
20	Administrators must be able to credit phone accounts to record receipt of payment for bills.
21	Administrators must be able to delete billing plans that are no longer offered to customers.
23	Administrators must be able to take the system offline in order to perform maintenance.
26	Administrators must be able to add and edit filters on a phone account.
30	Administrators must be able to reset the software running on phones if a phone is still recognized by the system, but cannot achieve some of its required functionality.
31	The system must automatically generate bills at 5pm, and must automatically suspend the accounts of customers who are more than 3 months behind on their payments.

Table 1: List of Included Use Cases

2.3 User Characteristics

There are two main types of users of the phone system.

- Customers are members of the general public, and they only interact with the phones. They are assumed to have no special training other than being familiar with existing phone systems.
- Administrators are employees of the company offering the phone system. They have access to a special interface to create and alter accounts, and monitor calls. They are assumed to have a working knowledge administering a phone system similar to this system and they are

assumed to have a general knowledge of computers. They are not assumed to have any special training beyond these two requirements.

2.4 General Constraints

The only general constraint on the system is that the system must work with the Nortel i2004 IPPhones. It does not need to support any other model of phone, and there are any laws or regulations by which the system needs to abide.

2.5 Assumptions and Dependencies

Four assumptions have been made about the system: administrators are not malicious, the server hardware will not fail, a loss in power that affects the server will also cause the network to fail (terminating all calls), and that the system will only recieve one input at a time from each source. The first assumption means that administrators can be trusted to not sabotage the system by deleting the accounts of all other administrators, and the system therefore does not need to include a way to add an admin account without being logged in. The second and third assumptions mean that the server cannot fail without all ongoing calls being terminated, and the system therefore does not need to include some method of determining when a call ended when it was not being monitored by the server. The fourth assumtion means that for each phone and administrator interface console, the system will not recieve multiple inputs at the same time – these inputs will instead be recieved in series.

UC#	Description
05	Administrators must be able to log out, and must be logged out automatically after 10 minutes of inactivity.
09	There must be a maximum number of ongoing calls allowed, and the administrator must be able to modify it.
10	Administrators must be able to view a list of ongoing calls.
11	Administrators must be able to configure the extensions that are used as emergency numbers.
15	Administrators must be able to update phone accounts once they have been created.
19	Bills must be printed so that they can be mailed to customers.
22	The system must be able to run hardware tests on phones to detect interruptions in service automatically.
24	Administrators must be able to put the system back online after taking it offline to perform maintenance.
26	Administrators must be able to configure filters on what extensions each phone account can call and be called by.
29	Administrators must be able to update user accounts once they have been created.

Table 2: List of Non-Included Use Cases

3 Specific Requirements

3.1 External Interfaces

3.1.1 User Interface

An overview map of the administrator interface screens is presented below in Figure 2. This overview shows the paths that the user can take to transition between the screens in the interface. The screen name corresponding to the number in the overview map can be found in Table 3.

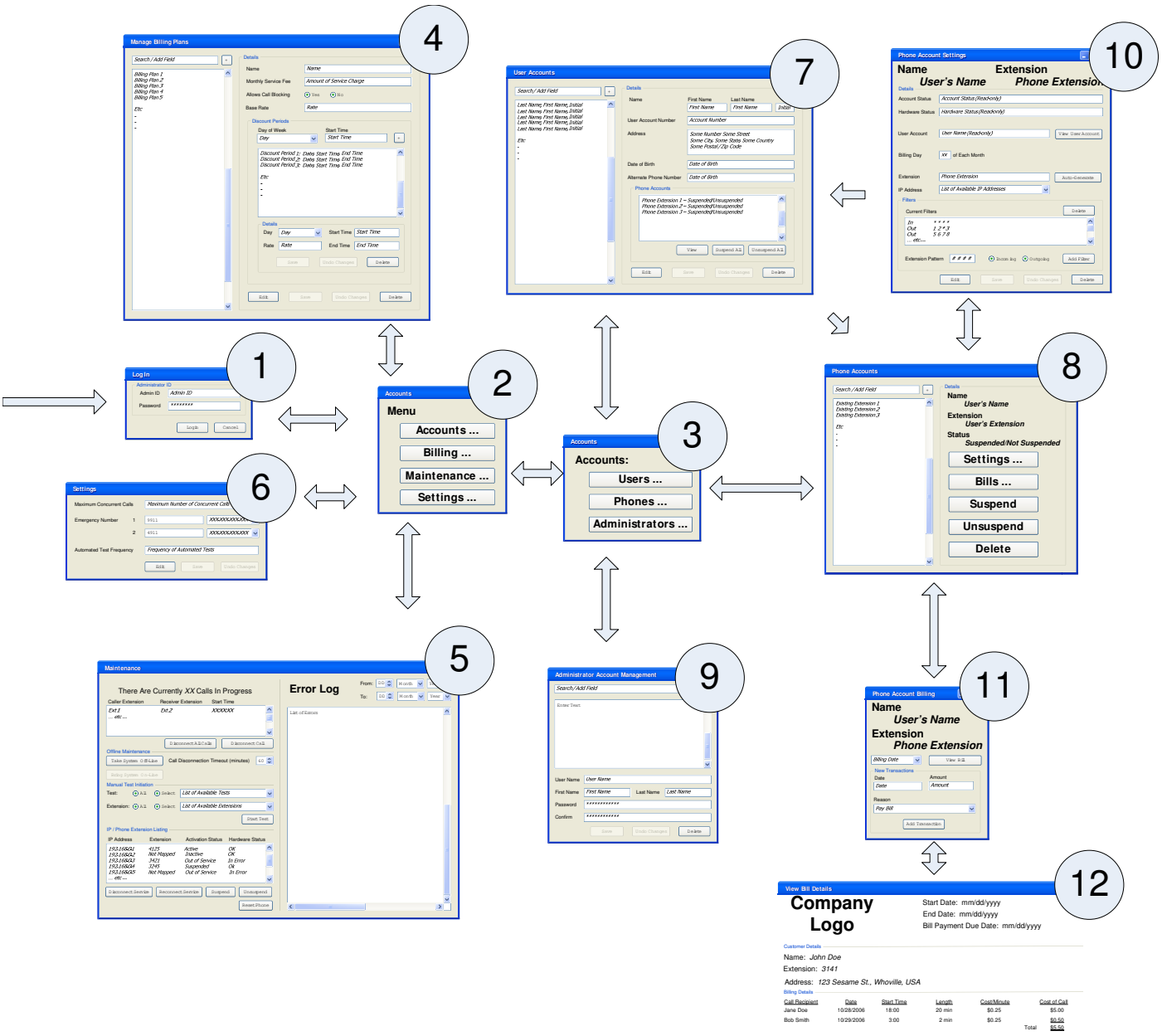


Figure 2: Overview Map of UI Forms

Form 1	Administrator Log-In
Form 2	High Level Menu
Form 3	Accounts Sub Menu
Form 4	Billing Plans Form
Form 5	Maintenance Form
Form 6	Settings Form
Form 7	User Accounts Form
Form 8	Phone Accounts Sub Menu
Form 9	Administrator Accounts Form
Form 10	Phone Account Settings Form
Form 11	Phone Account Billing Form
Form 12	Phone Bill Viewing Form

Table 3: List of GUI Forms

Administrator Session Life Cycle

The administrator login form (Form 1) is used to authenticate administrators in order to protect the system from unauthorized access. The form has input boxes for admin id and password, and buttons to log in and cancel. All events that are raised are sent to the Displaying Admin Session state machine.

- Clicking the "Login" button attempts to log into the system with the credential provided in the two input boxes. This raises a Login event.
- If validation fails, an error dialog box communicates this with the user.
- Successful validation causes a transition to Form 2 (High Level Menu)
- Clicking the "Cancel" button clears the contents of the input boxes. This raises a Cancel event.

The top level menu form (Form 2) allows the administrator access to various different forms. It has four buttons labelled "Accounts...," "Billing...," "Maintenance...," and "Settings...." All events raised by this form are sent to the Displaying Administrator Session state machine.

- Clicking the "Accounts..." button causes a transition to Form 3 (Managing Accounts). This raises a ManageAccounts event.
- Clicking the "Billing..." button causes a transition to Form 4 (Managing Billing Plans). This raises a ManageBillingPlans event.
- Clicking the "Maintenance..." button causes a transition to Form 5 (Maintenance). This raises a PerformMaintenance event.
- Clicking the "Settings..." button causes a transition to Form 6 (Settings). This raises a ChangeSettings event.
- Clicking the "X" at the top right causes a transition to Form 1 (Login Form). This raises a CloseScreen event.

Account Management

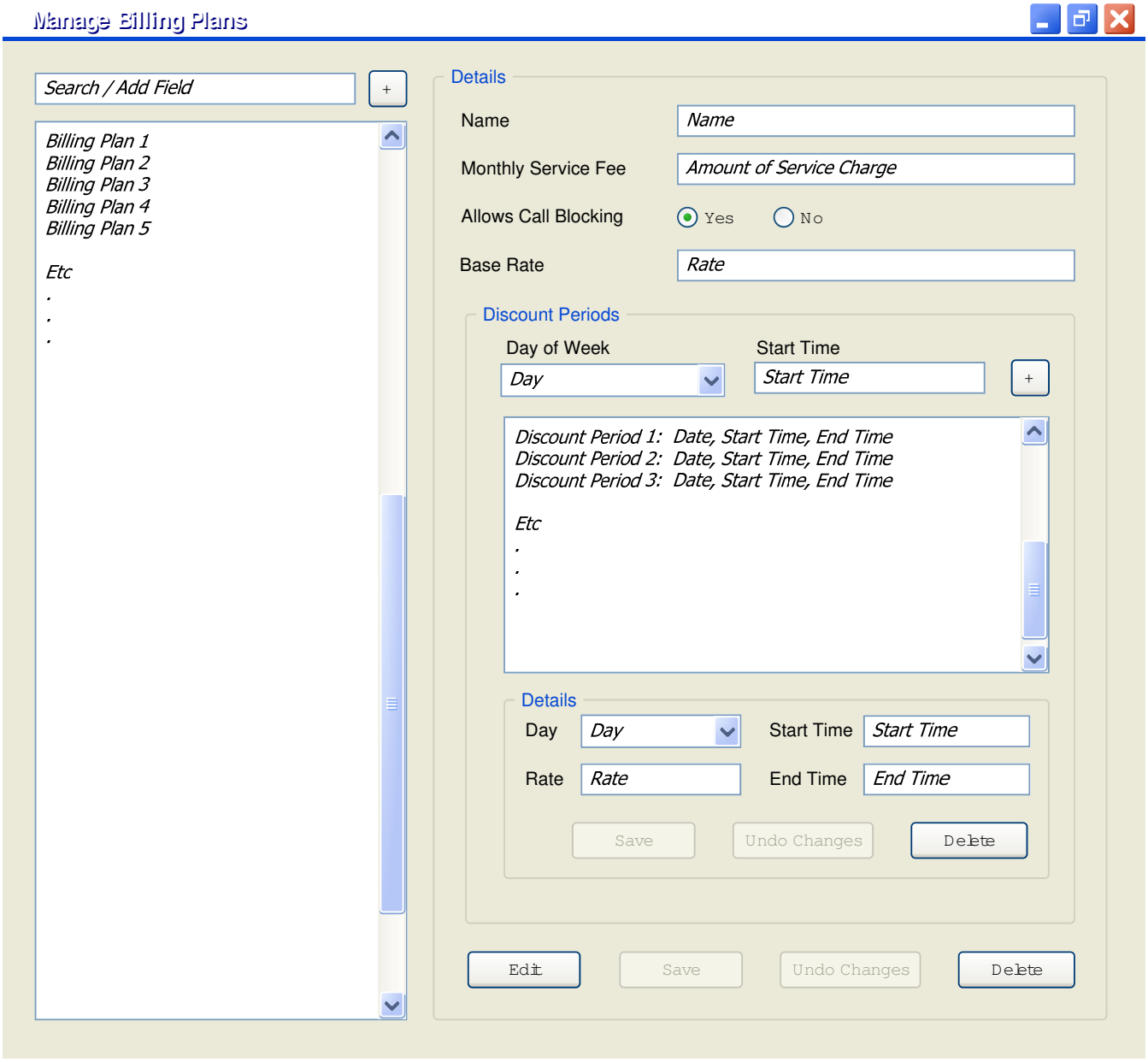
The Accounts form is a multiplexer to three other forms that have more specific responsibilities. It can be used to access each of the different types of accounts by means of three buttons: "Users...," Phones," and "Administrators." All events that this screen raises are directed to the Managing Accounts state machine.

- Clicking the "Users..." button causes a transition to Form 7, and raises a ManageUserAccounts event.
- Clicking the "Phones..." button causes a transition to Form 8, and raises a ManagePhoneAccounts event.
- Clicking the "Administrators..." button causes a transition to Form 9, and raises a ManageAdminAccounts event.
- Clicking the "X" at the top right causes a transition to Form 2, and raises a GoBack event.

Billing Plans Management

The Manage Billing Plans form (Graphical UI 1) allows an administrator to edit all aspects of existing billing plans, as well as create and delete them. There is also a subsection of the form that allows an administrator to edit all discount periods associated with a billing plan. All events raised by this form are sent to the Managing Billing Plans state machine.

- As a name is typed into the Search/Add field at the top left of the screen, the list below it will update to show only billing plans with names starting with the contents of the field.
- When a billing plan is selected in the left pane, the details appear in the right pane. This raises a SelectPlan event.
- Clicking on the "+" button at the top left of the screen (above the list of billing plans) will create a new billing plan. This raises a SelectPlan event, and then immediately raises an EditPlan event.
- Clicking on the "Edit" button makes billing plan details writable (they are read-only by default). This will also prevent any other administrators from reading it. This raises an EditPlan event.
- Clicking on the plan level "Save" button commits any changes (for the plan and any periods), makes the plan details read-only again, and allows other admins to read it again. This raises a Save event.
- Clicking on the plan level "Undo Changes" button rolls back any changes to the billing plan's data. This raises an UndoAll event.
- Clicking on the plan level "Delete" button prompts the admin for confirmation, then removes the billing plan from the system. This raises a DeletePlan event.
- Entering data in the "Day of Week" and "Start Time" input boxes changes what is displayed in the list of discount periods.

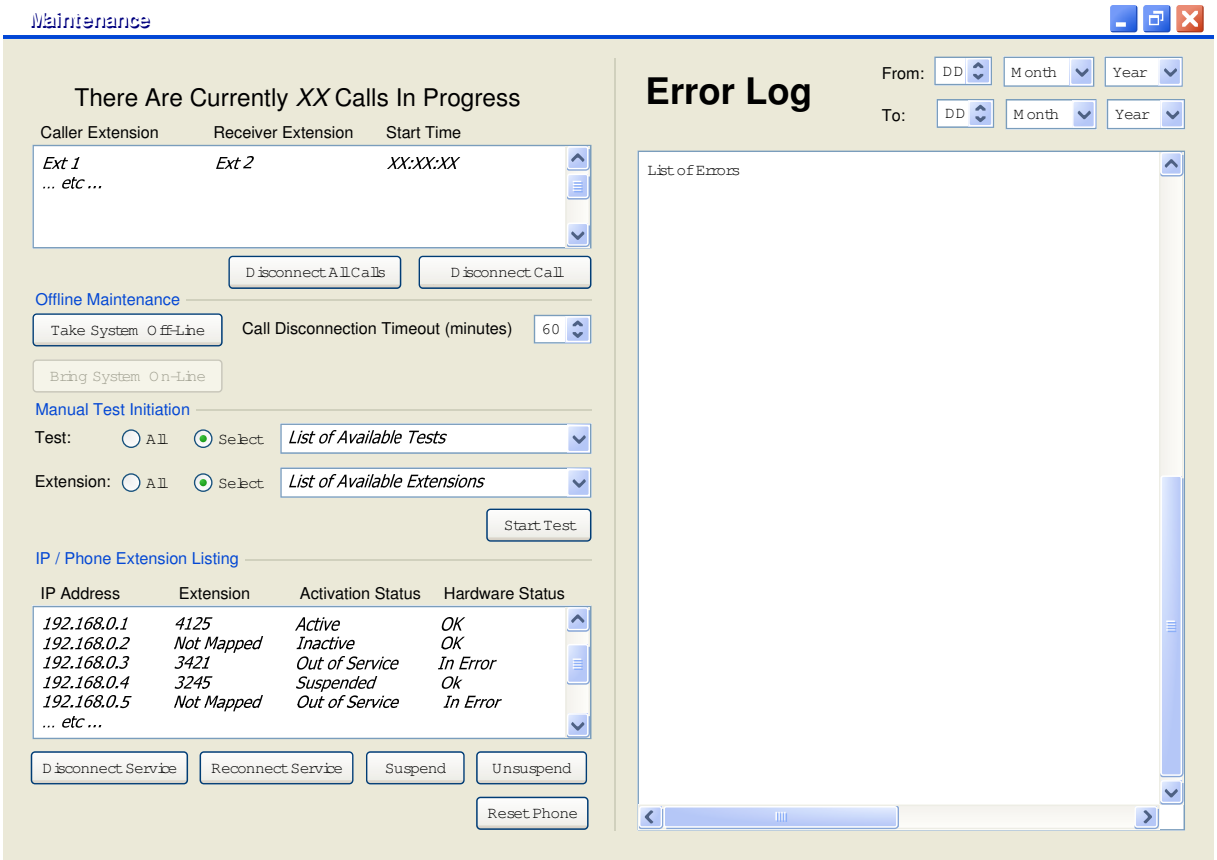


Graphical UI 1: Billing Plans Management Form (Form 4)

- When a Discount Period is selected in the list, the details appear in the fields below. This raises a SelectPeriod event.
- Clicking on the "+" button directly above the list of discount periods will create a new period. This raises a SelectPeriod event.
- Clicking on the period level "Save" button will save any changes that have been made to discount periods. This raises a SavePeriod event.
- Clicking on the period level "Undo Changes" button rolls back any changes to discount periods since the last time they were saved. This raises an UndoPeriods event.
- Clicking on the period level "Delete" button deletes the currently selected discount period. This raises a DeletePeriod event.
- Clicking the "X" at the top right returns to Form 2 (High Level Menu). This raises a GoBack event.

System Maintenance

The Maintenance form (Graphical UI 2) allows an administrator to view and manage ongoing calls, take the system offline for maintenance and put it back online, run hardware tests, view all phone accounts by extension, and view an error log. Events raised on this form are sent to the Performing Maintenance state machine.



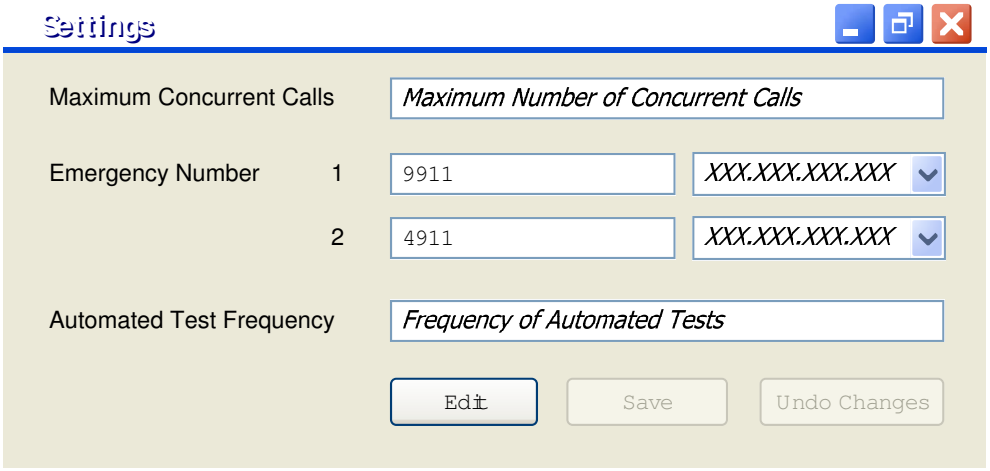
Graphical UI 2: System Maintenance Form (Form 5)

- The list of calls in the upper left of the form updates dynamically as calls begin and end.
- The "Disconnect All Calls" button terminates all ongoing calls. This raises a DisconnectCall event for every entry in the list of calls.
- The "Disconnect Call" button terminates the selected entry in the list of calls. This raises a DisconnectCall event.
- The "Take System Off-Line" button takes the system offline, setting the maximum number of calls to zero and terminating all existing calls after the Call Disconnection Timeout. This raises a SetSystemOffline(t) event, where t is the value contained in the Call Disconnection Timeout box.
- The "Bring System On-Line" button brings the system online, setting the maximum number of calls to the previous value. This raises a SetSystemOnline event.

- The “Start Test” button begins a hardware test as selected by the test and extension inputs directly above it. This raises a StartTest event.
- The “Disconnect Service” button will take the phone account selected in the listing offline. This raises a Disconnect event.
- The “Reconnect Service” button will put the phone account selected in the listing online. This raises a Reconnect event.
- The “Suspend” button will set the phone account selected in the listing to suspended. This raises a Suspend event.
- The “Unsuspend” button will set the phone account selected in the listing to unsuspended. This raises an Unsuspend event.
- The “Reset Phone” button will reset the software on the phone selected in the listing. This raises a Reset event.
- The “From” and “To” date fields allow the Administrator to filter the error log for the time period of interest.
- Clicking the “X” at the top right causes a transition to Form 2 (High Level Menu). This raises a GoBack event.

System Settings

The Settings form (Graphical UI 3) allows administrators to change the current system settings: the maximum number of calls, emergency number extensions and IPs, and automated test frequency. All events raised by this form are sent to the Changing Settings state machine.

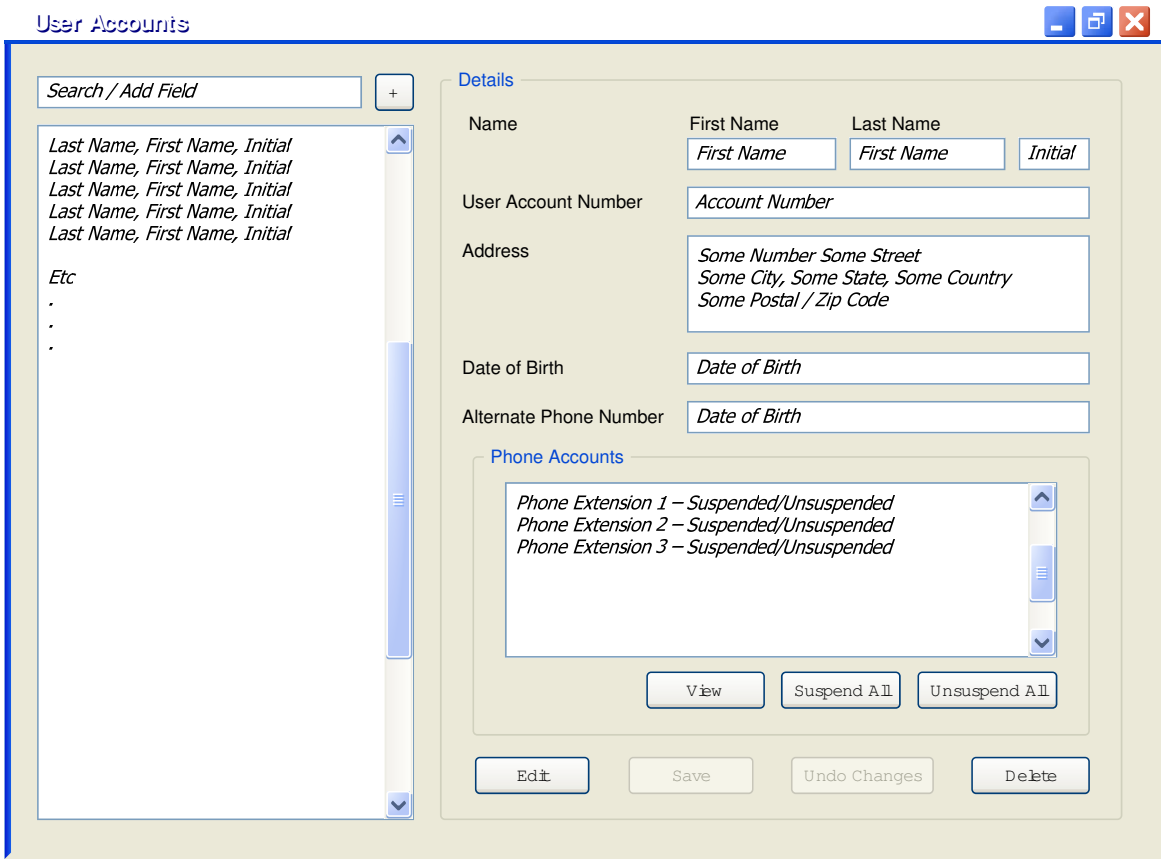


Graphical UI 3: System Settings Form (Form 6)

- Clicking the "Edit" button makes all fields writable (they are read-only by default). This raises an Edit event.
- Clicking the "Save" button validates all form values. This raises a Save event.
- Successful validation resets the form to read-only and commits the values.
- Failing validation prompts the administrator to enter valid values and leaves fields writable.
- Clicking the "Undo Changes" button resets all fields to their values before the "Edit" button was pressed. This raises an UndoChanges event.
- Clicking the "X" at the top right causes a transition to Form 2 (High Level Menu). This raises a GoBack event.

User Account Management

The User Accounts form (Graphical UI 4) allows an administrator to edit user accounts. It allows searching by user name, setting user data, and viewing phone accounts. All events raised by this form are sent to the Managing User Accounts state machine.



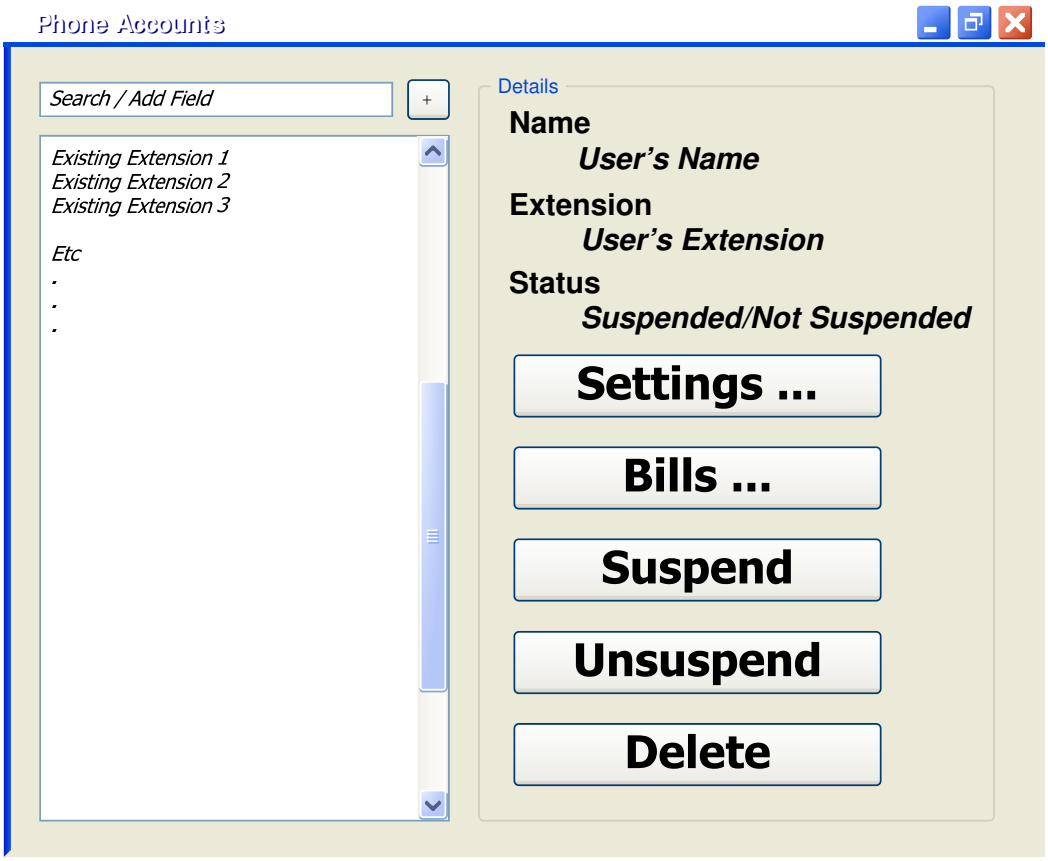
Graphical UI 4: User Account Management Form (Form 7)

- Typing in the input box in the upper left will filter the values in the list of users.
- Selecting a user in the list will display their information in the right. This raises a SelectUser event.
- Clicking on the "+" button will add a new User Account to the list. This raises a SelectUser event, then immediately raises an Edit event.
- Clicking the "Edit" button will make the user information writable (it is read-only by default). This raises an Edit event.
- Clicking the "Save" button will commit the changes to the user's data, and will return the user information to read-only. This raises a Save event.
- When a user account is selected in the left pane, the details appear in the right pane
- Clicking the "Undo Changes" button will undo any changes that have been made since the user clicked the "Save" button. This raises an UndoAll event.

- Clicking the "Delete" button will prompt the admin for confirmation, then close the user account and all associated phone accounts. This raises a DeleteUser event.
- Clicking the "View" button will cause a transition to Form 8 (Phone Accounts Form), and will automatically fill in the search filter on that form with the the phone account selected in the list. This raises a ViewPhoneAccount event.
- Clicking the "Suspend All" button will suspend all phone accounts for the selected user. This raises a SuspendAll event.
- Clicking the "Unsuspend All" button will unsuspend all phone accounts for the selected user. This raises an UnsuspendAll event.
- Clicking the "X" at the top right returns to Form 3 (Account Management Form). This raises a GoBack event.

Phone Accounts Main Form

The Phone Accounts form (Graphical UI 5) allows administrators to select phone accounts, and move to other forms where they can be edited. Any events that are raised from this form are sent to the Managing Phone Accounts state machine.



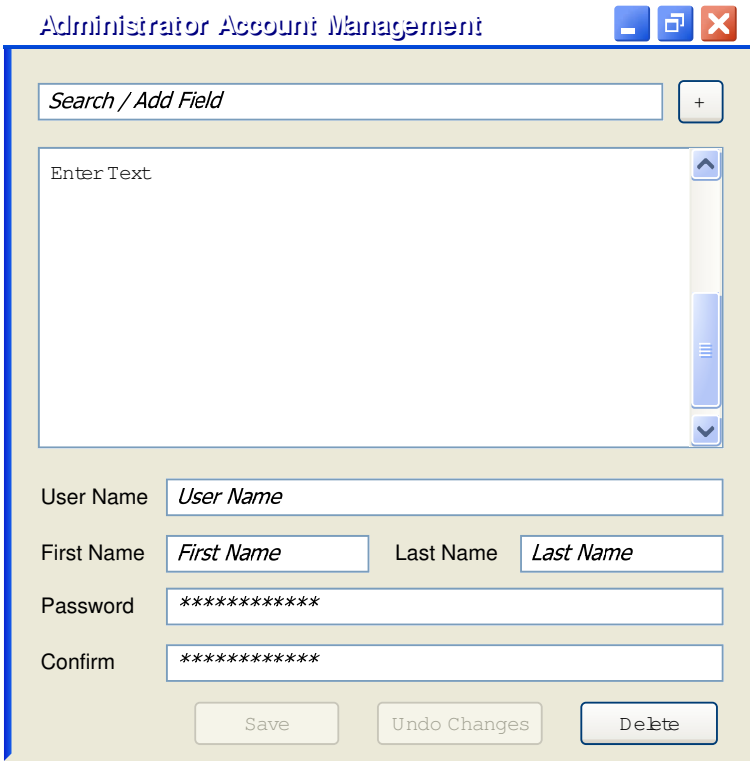
Graphical UI 5: Phone Accounts Main Form (Form 8)

- Entering data into the search field in the upper left will filter the data in the list of extensions.
- Clicking on an entry in the list of extensions will bring up its data in the list of right pane. This raises a SelectAccount event.
- Clicking the "+" button will add a new Phone Account to the list. This raises a SelectAccount event.
- Clicking the "Settings..." button causes a transition to Form 10 (Phone Account Settings). This raises a ViewPhoneAccount event.
- Clicking the "Bills..." button causes a transition to Forms 11 (Phone Account Billing). This raises a ViewPhoneAccountBilling event.
- Clicking the "Suspend" button suspends the selected phone acocunt. This raises a Suspend event.

- Clicking the "Unsuspend" button unsuspends the selected phone account. This raises an Unsuspend event.
- Clicking the "Delete" button prompts the admin for confirmation, then closes the phone account. This raises a DeletePhoneAccount event.
- Clicking on the "X" at the top right causes a transition to the previous form (either Form 3 (Account Management) or Form 7 (User Account Management)). This raises a GoBack event.

Administrator Account Management

The Administrator Account Management form (Graphical UI 6) allows administrators to edit data for other administrators. It also allows the deletion of admin accounts that are no longer being used. Any events raised by this form are sent to the Managing Admin Accounts state machine.

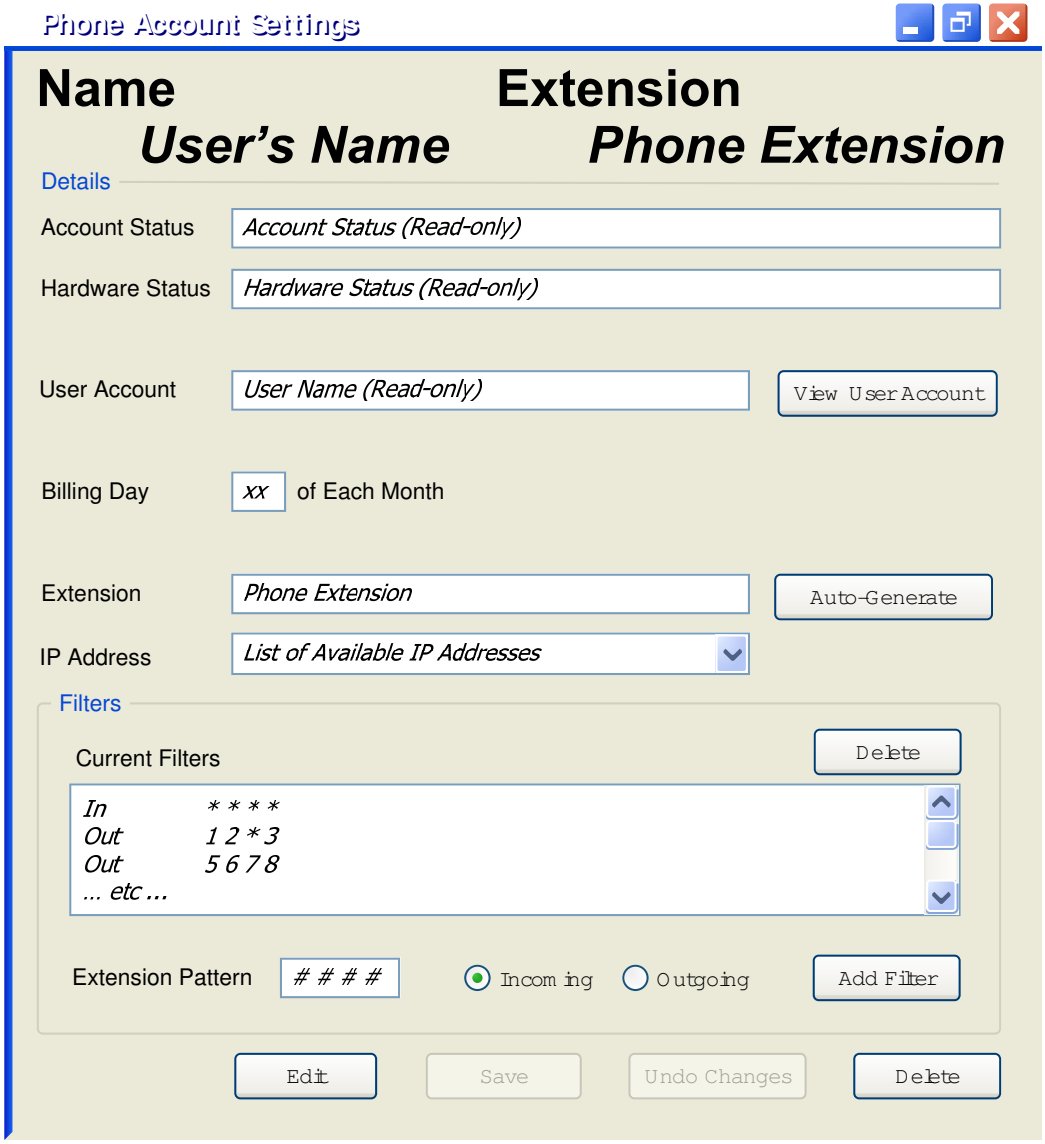


Graphical UI 6: Administrator Account Management Form (Form 9)

- When the data is entered into the field at the top, the list below it will update.
- When the user selects an item from the list, that admin's details are displayed in the inputs below that. This raises a SelectAdmin event.
- Clicking on the "+" button will add a new admin account. This raises a SelectAdmin event.
- Clicking on the "Save" button commits all changes. This raises a Save event.
- Clicking on the "Undo Changes" button will revert all changes made since the last save. This raises an UndoChanges event.
- Clicking on the "Delete" button will ask the admin to confirm his/her intent, and will remove the selected admin account. This will raise a Delete event.
- Editing any field will raise a ChangeData event.
- Clicking on the "X" in the upper right will cause a transition to Form 3 (Accounts Management). This raises a GoBack event.

Phone Account Settings

The Phone Account Settings form (Graphical UI 7) allows administrators to view the account and hardware status of accounts, and edit billing days, extension/IP mappings, and filters. Any events raised on this form are sent to the Phone Account Settings state machine.



Graphical UI 7: Phone Account Settings Form (Form 10)

- Clicking on the "Edit" button makes the billing day, extension/IP mapping, and filters to become writable (they are read-only by default). This raises an Edit event.
- Clicking on the "Save" button commits changes, and makes the form read-only again. This raises a Save event.
- Clicking on the "Undo Changes" button undoes any changes since the last save. This raises an UndoAll even.

- Clicking on the "Delete" button prompts the admin for confirmation, then closes the current phone account. This raises a Delete event.
- Clicking on the "View User Account" button causes a page transition to Form 7 (User Accounts), and automatically populates the search field with the phone account holder's name. This raises a ViewUserAccount event.
- Clicking on the "Auto-Generate" button searches for a free extension in the system, and populates the phone extension input. If there are no free extensions, the administrator is notified. This raises an Auto-Generate event.
- The IP Address input can drop down to show available IPs. Entering part of a number in the input will filter the drop down.
- When the user selects an entry in the "Current Filters" list, the corresponding filter's data is displayed in the fields below the list. This raises a SelectFilter event.
- Clicking the "Add Filter" button adds a new filter, and selects it. This raises a SelectFilter event.
- Clicking the "Delete Filter" button deletes the currently selected filter. This raises a Delete-Filter event.
- Clicking on the "X" in the upper right corner causes a transition to the previous screen (either Form 7 (User Accounts) or Form 8 (Phone Accounts)). This raises a GoBack event.

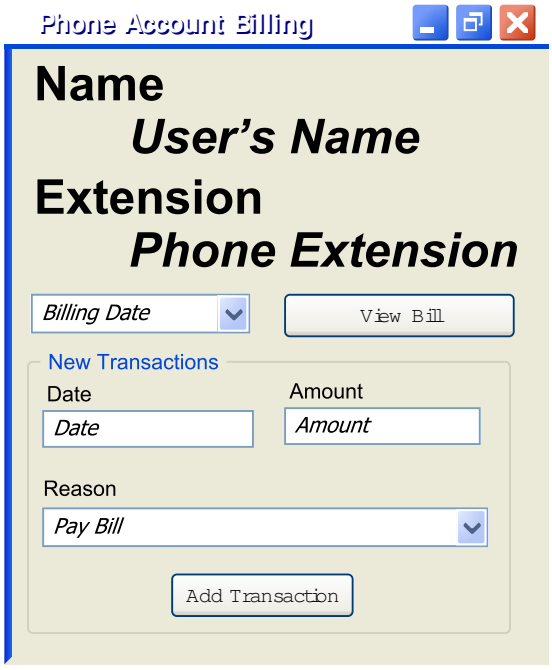
Phone Account Billing

The Phone Account Billing form (Graphical UI 8) allows administrators to view billing information for, and add credits to phone accounts. Events raised on this form get sent to the Viewing Billing Info state machine.

- Clicking the "View Bill" button causes a transition to Form 12 (View Bill Details). This raises a ViewBill event.
- Clicking the "Add Transaction" button causes a credit to be applied to the phone account. This raises an AddTransaction event.
- Clicking the "X" at the top right causes a transition to Form 8 (Phone Accounts). This raises a GoBack event.

View Bill Details form allows administrators to view a bill. It displays the start, end, and due dates for the bill, the name, extension, and address of the customer, and data for each call made in the billing period. The call data includes the recipient, start date, start time, length, minutely rate, and total cost.

- Clicking the "X" at the top right causes a transition to Form 11 (Phone Account Billing). This raises a GoBack event.



Graphical UI 8: Phone Account Billing Form (Form 11)

3.1.2 Hardware Interface

All hardware events raised are sent to one of the Call Processing state machines or to the Hardware Testing state machine.

The hardware interface output events listed below are received and processed by the system. The associated system components which may process each event are included. Each event is processed by only one state machine.

- DigitPressed - Dialing an Extension (State Machine 13), Connect Call (State Machine 14), Edit Call Blocking (State Machine 16)
- DigitReleased - Dialing an Extension (State Machine 13), Connect Call (State Machine 14), Edit Call Blocking (State Machine 16)
- OnHook - Make a Call (State Machine 12), Connect Call (State Machine 14), Receive a Call (State Machine 15)
- OffHook - Make a Call (State Machine 12), Connect Call (State Machine 14), Receive a Call (State Machine 15)

The hardware interface input events listed below are generated by the system and sent to a hardware device. The associated system components which generate each event are included.

- DisplayString - Dialing an Extension (State Machine 13), Edit Call Blocking (State Machine 16)
- PlayTone - Make a Call (State Machine 12), Dialing an Extension (State Machine 13), Connect Call (State Machine 14), Receive a Call (State Machine 15), Edit Call Blocking (State Machine 16)
- StartRinging - Connect Call (State Machine 14), Receive a Call (State Machine 15)
- StopRinging - Connect Call (State Machine 14), Receive a Call (State Machine 15)
- StartAudioSend - Connect Call (State Machine 14)
- StopAudioSend - Connect Call (State Machine 14)
- StartAudioReceive - Connect Call (State Machine 14)
- StopAudioReceive - Connect Call (State Machine 14)
- TestOnHook - Make a Call (State Machine 12), Hardware Testing (State Machine 18)
- TestOffHook - Make a Call (State Machine 12), Hardware Testing (State Machine 18)

3.2 Functional Requirements

3.2.1 Use Cases

The use case diagram in Figure 3 shows a summary of all of the use cases. Not all of the use cases listed in this diagram have been modelled, as described in Section 2.2 - Product Features.

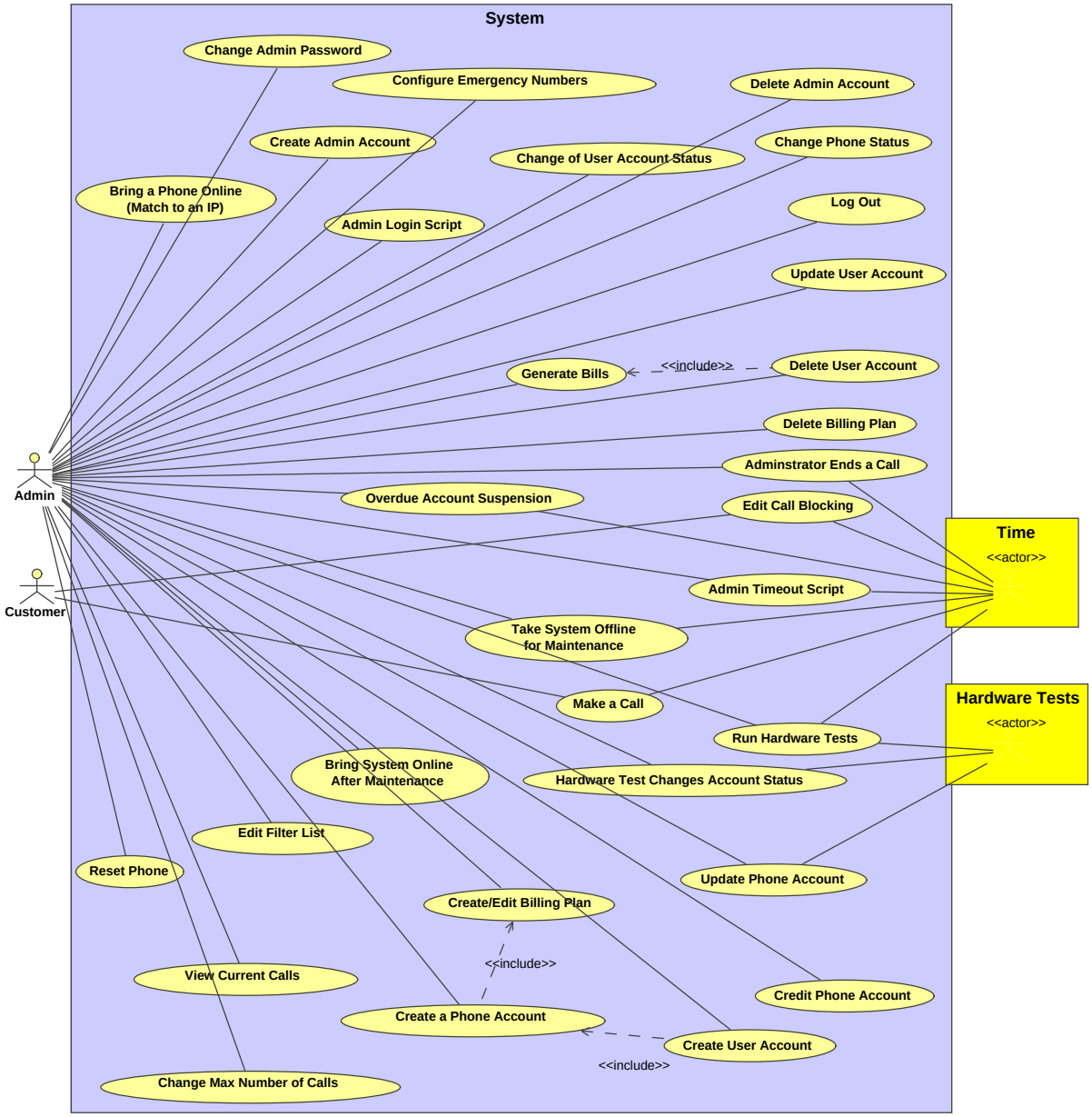


Figure 3: Overview of System Use Cases

Use Case Description: Make a Call

Name: Make a Call

Use Case Number: UC 1

Author: Colin Rhodes

System: Call Processing

Actors:

- User (Caller)
- Second User (Callee)
- Call Processing Subsystem (System)
- Time

Event/Precondition:
Caller's phone is on the hook.
Caller's phone is enabled and online.
Caller's phone account is not suspended.
System is running.

Overview/Postcondition:
Caller and callee are connected in a phone call.

References: R6-8, R13, R15, M1-15, M1-23, M1-27. M1-41, M1-83, M2-23

Related Use Cases:

- UC 03 - If the user dials a special number (call blocking), then UC 03 applies instead of UC 01

Typical Process Description			
Caller	*Callee*	*System*	*Time*
1 Caller picks up phone			
		2 System prompts caller with dial tone	
3 Caller dials four-digit extension			
		4 System prompts callee by ringing his phone. System gives feedback to caller by playing the ringing sound.	

	5 Callee picks up phone		
		6 System establishes audio path between caller and callee	
7 Caller hangs up phone			
		8 System gives feedback to Callee with no audio signal	
		9 System bills the call to Caller's phone account	
	10 Callee hangs up phone		
Alternate Flow 1: Callee doesn't pick up phone			
	5 Callee doesn't pick up phone within 10 rings		
		6 System terminates call attempt. System stops ringing callee's phone and plays fast busy tone on caller's phone.	
7 Caller hangs up phone			
Alternate Flow 2: Callee hangs up phone before Caller			
	7 Callee hangs up phone		
		8 System gives feedback to Caller with no audio signal	
		9 System bills the call to Caller's phone account	
10 Caller hangs up phone			

Use Case Description: Edit Call Blocking

Name: Edit Call Blocking

Use Case Number: UC 3

Author: Colin Rhodes

System: Call Processing

Actors:

- User
- Call Processing Subsystem (System)
- User Account Subsystem
- Time

Event/Precondition:
User's phone is on the hook.
User's phone is enabled and online.
User's phone account is not suspended.
System is running.

Overview/Postcondition:
User has edited call blocking and his phone is on the hook.

References: M1-84, M2-12, M2-16, M2-19, M2-74, M2-75

Related Use Cases:

- UC 01 - starts in the same way as UC 03, except that in UC 03, user dials a special number

Typical Process Description		
User	*System*	*Time*
1 User picks up phone		
	2 System prompts caller with dial tone	
3 User dials special number (#70)		
	4 System queries User Account Subsystem to check if user's phone account has call blocking enabled	
	5 User Account Subsystem determines that user's phone account has call blocking enabled and sends this information to System	

	6 System allows user to enter call blocking system	
7 User dials 1 to add blocked numbers		
	8 System enters add mode	
9 if User dials * 9.2 User hangs up phone	9.1 System terminates call and gives feedback to user with no audio signal	
10 User dials 4-digit extension		
11 User dials #		
	12 if System is in add mode, extension is valid, and extension is not in block list 12.1 System adds extension to block list	
	13 if System is in remove mode, extension is valid, and extension is in block list 13.1 System removes extension from block list	
	14 System returns to step 9	
Alternate Flow 1: User dials 2 to remove blocked numbers		
7 User dials 2 to remove blocked numbers		
	8 System enters remove mode and returns to step 8	

Use Case Description: Reset Phone Connection

Name: Reset Phone Connection

Use Case Number: UC 30

Author: Adam Nace

System: Hardware Testing and Control Subsystem

Actors:

- Administrator (Admin)
- Hardware Testing and Control Subsystem (System)

Event/Precondition:
Administrator requests reset of phone software

Overview/Postcondition:
Phone Software for selected phone is reset

References: N/A

Related Use Cases:
N/A

Typical Process Description	
Administrator	*System*
1 Admin requests reset of phone software	
	2 System prompts for phone extension to be reset
3 Admin provides phone extension to be reset	
	4 System prompts Admin for confirmation
5 Admin confirms reset of selected phone's software	
	6 System resets software of selected phone, and logs that activity.

Use Case Description: Create a User Account

Name: Create a User Account

Use Case Number: UC 12

Author: Adam Nace

System: Administrative Control Subsystem

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
Administrator requests a new user account to be created.

Overview/Postcondition:
The new user account exists with all essential contact information

References: R21, R22

Related Use Cases:

- UC 13
- UC 14

Typical Process Description	
Administrator	*System*
1 Admin requests creation of a new user	
	2 System prompts Admin to enter contact information
3 Admin submits contact information	
	4 System commits user contact information to persistent storage
	5 System prompts Admin to (optionally) add telephones to user profile
6 If Admin wished to add phones to user account 6.1 Include UC 13	

Use Case Description: Delete User Account

Name: Delete User Account

Use Case Number: UC 14

Author: Adam Nace

System: Administrative Control Subsystem

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
Administrator requests deletion of a user

Overview/Postcondition:
The selected user account is flagged as deleted, but remains in persistent storage.

References: R50

Related Use Cases:

- UC 12
- UC 29

Typical Process Description	
Administrator	*System*
1 Admin requests deletion of a user	
	2 System presents a list of users that can be deleted
3 Admin selects a user to delete	
	4 System flags as deleted any phone accounts that do not have an outstanding balance
	5 If any phone accounts linked to the user account have outstanding balances 5.1 System generates a bill for phone accounts (include UC 19) else 5.2 User Record is Flagged as Deleted

Use Case Description: Create Phone Account

Name: Create Phone Account

Use Case Number: UC 13

Author: Adam Nace

System: Administrative Control Subsystem

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
Administrator requests the creation of a phone account.

Overview/Postcondition:
A phone account is created, attached to a user, and paired with a billing plan.

References: N/A

Related Use Cases:

- UC 12
- UC 15

Typical Process Description	
Administrator	*System*
1 Admin requests the creating of a new phone account	
	2 System presents a list of users to add the phone account to
3 Admin chooses a user from the list	
	4 If there is no active billing plans 4.1 Prompt Admin to enter new billing plan (include UC 16)
	5 System provides a list of unassigned IP Addresses
	6 System prompts Admin to select an IP Address for the phone
7 Admin provides IP Address for the new phone	
	8 System prompts Admin to assign an extension to the new phone
9 Admin provides an extension for the new phone	

	10 System validates new extension 10.1 If phone extension is invalid or already in use, return to 5
	11 System presents a list of billing plans for the phone account to use.
12 Admin chooses a billing plan	
	13 System enables incoming and/or outgoing calls based on the selected billing plan

Use Case Description: Administrator credits a phone account

Name: Administrator credits a phone account

Use Case Number: UC 20

Author: Steven Robertson

System: Billing Subsystem

Actors:

- Administrator (Admin)

Event/Precondition:
An administrator needs to credit the balance of a phone account.

Overview/Postcondition:
The balance on the phone account has been decreased by the appropriate amount.

References: R51, M1-16, M1-31

Related Use Cases:

- UC 13 - Creation of a phone account determines when UC 20 should happen.
- UC 14 - Deletion of a phone account changes when UC 20 should happen.

Typical Process Description	
Administrator	*System*
1 Admin requests to credit a phone account	
	2 System displays a list of phone accounts
3 Admin chooses a phone account from the list	
	4 System prompts admin for the amount of the credit
5 Admin enters amount of credit	
	6 System prompts admin for a reason
7 Admin enters the reason (e.g. paying a bill)	
	8 System deducts credit from the outstanding balance
	9 If the phone account has been suspended for overdue payment, and its balance is 0. 9.1 System unsuspends the phone account. Else if the phone account has been cancelled, and its balance is 0. 9.2 System flags the phone account as deleted.

Use Case Description: Create Phone Account Filter

Name: Create Phone Account Filter

Use Case Number: UC 26

Author: Brett Lounsbury

System: Phone Account Management Subsystem

Actors:

- Administrator
- Phone Account Management Subsystem (System)

Event/Precondition:
Administrator is logged in.
Administrator is currently viewing a phone account.
Administrator has the edit lock for the phone account.
System is running.

Overview/Postcondition:
Administrator has successfully added a filter to the account.

References: R25, M1-3, M1-52, M1-53, M1-54

Related Use Cases:
UC 03 - The concept of filters is similar to that of blocked numbers.

Typical Process Description	
Administrator	*System*
1 Administrator requests to add a filter to the current phone account	
	2 System prompts for filter direction and extension pattern
3 Administrator selects a direction for the filter (in/out) and enters an extension pattern	
	4 System verifies that extension pattern is valid (series of 4 digits {0-9,})
	5 If the filter does not already exist
	5.1 System adds the filter to the list of filters for the phone account

Use Case Description: Administrator Creating a Billing Plan

Name: Administrator Creating a Billing Plan

Use Case Number: UC 16

Author: Steven Robertson

System: Billing Subsystem

Actors:

- Administrator (Admin)

Event/Precondition:
The administrator has requested to enter data for a billing plan.

Overview/Postcondition:
The billing plan the administrator chose has been created.

References: R21, R54, R56, R58, M1-7, M1-42, M2-15

Related Use Cases:

- UC 13 - If an admin tries to create a phone account (UC 13) when there are no billing plans, UC 16 will be invoked.
- UC 21 - Deleting a billing plan (UC 21) is the opposite of UC 16's outcome.

Typical Process Description	
Administrator	*System*
	1 System provides a list of plans to choose from
2 Admin chooses to create a new plan	
	3 System prompts admin for the new billing plan's monthly rate, discount periods, and whether or not the plan allows call blocking
4 Admin enters a new monthly rate, discount periods, and whether or not the plan allows call blocking	
	5 System validates that the discount periods do not overlap
	6 System requests confirmation of the updated values from the admin
7 Admin confirms the updated values.	
	8 System commits updated plan.
	9 System confirms successful completion of setting the values.

Use Case Description: Delete Billing Plans

Name: Delete Billing Plans

Use Case Number: UC 21

Author: Adam Nace

System: Billing Subsystem

Actors:

- Administrator (Admin)
- Billing Subsystem (System)

Event/Precondition:
Admin requests a billing plan to be deleted.

Overview/Postcondition:
The billing plan is flagged as deleted, but remains in persistent storage

References: R27, R28, R54

Related Use Cases:

- UC 16 - Creating a billing plan (UC 16) is the opposite of UC 21's outcomes.

Typical Process Description	
Administrator	*System*
1 Admin requests deletion of billing plan	
	2 System displays list of active billing plans
3 Admin selects billing plan from list	
	4 System requests confirmation
	5 Remove billing plan from list of active plans, but keep it in the system

Use Case Description: Bills Are Generated

Name: Bills Are Generated

Use Case Number: UC 31

Author: Steven Robertson

System: Billing Subsystem

Actors:

- Time

Event/Precondition:
It is 5pm.

Overview/Postcondition:
A bill has been printed.

References: R44, R45, R48, R49, R52, M1-80, M1-82

Related Use Cases:

- UC 19 - Sometime after bills are generated, an administrator will print them.

Typical Process Description	
Time	*System*
1 The use case starts when it is 5pm	
	2 The system finds all bills that need to be generated on the current day (monthly entries for all phone accounts that were opened or closed on the current day of the month, and have outstanding balances).
	3 For each bill found
	3.1 System prints the customer's name, address, monthly fee, and total owed. 3.2 If the bill is overdue, the system marks it as such. 3.3 If the bill is more than 3 months overdue, suspend the account. 3.4 For each call in the bill's time span 3.4.1 System prints the call's recipient, start time, length, minutely rate, and total charge.

Use Case Description: Administrator Login Script

Name: Administrator Login Script

Use Case Number: UC 4

Author: Jesse Bishop

System: Administrative Control

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
System requires Admin to be logged in before performing a requested an action

Overview/Postcondition:
Admin is logged in

References: R18, M1-85, M2-45, M2-82

Related Use Cases:

- UC 5 - Admin is logged out of System after ten minutes of inactivity

Typical Process Description	
Administrator	*System*
1 Admin requests an action which requires authentication	
2.2 Admin enters login information	2 While Admin has not entered valid login information and Admin is not logged in with a different 2.1 System prompts Admin to enter login information 2.3 If Admin with same user ID is logged in with a different administrator session 2.3.1 System notifies Admin that a different administrator session must be logged out before Admin can log in 2.4 If Admin entered invalid login information 2.4.1 System notifies Admin that Admin entered invalid login information
	3 System starts tracking time between Admin's requests to System

Use Case Description: Administrator Changes an Administrator's Password

Name: Administrator Changes an Administrator's Password

Use Case Number: UC 6

Author: Jesse Bishop

System: Administrative Control

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
Admin requests that an administrator account's password be changed

Overview/Postcondition:
The administrator account selected by Admin has a new valid password entered by Admin. The selected administrator account is logged off.

References: R18, R19, R20, R21, M1-85, M2-2, M2-50, M2-81

Related Use Cases:

- UC 04 - Requires Administrator to be logged in before changing an administrator account's password

Typical Process Description	
Administrator	*System*
1 Admin requests the password of a selected administrator account be changed	
	2 include(UC 04 - Administrator Login Script)
	3 While the Admin has not entered a valid new password for the selected administrator account 3.1 System prompts Admin for a new password for the selected administrator account
3.2 Admin enters a new password for the selected administrator account	
	3.3 System validates[1] new password
	4 System stores new password

[1]Validation of Password entails ensuring it meets the required format (which characters cannot be used, etc.). See Functional Requirement O16.

	5 If the selected administrator account is logged in 5.1 System logs out the selected administrator account

Use Case Description: Administrator Creates an Administrator Account

Name: Administrator Creates an Administrator Account

Use Case Number: UC 7

Author: Jesse Bishop

System: Administrative Control

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
Admin requests a new administrator account be created

Overview/Postcondition:
A new administrator account exists with login information specified by Admin

References: R18, R21, M1-65, M1-85

Related Use Cases:

- UC 04 - Requires Admin to be logged in before creating a new administrator account

Typical Process Description	
Administrator	*System*
1 Admin requests that a new administrator account be created	
	2 include(UC 04 - Administrator Login Script)
3.2 Admin enters login information for the new account	3 While Admin has not entered valid login information for the new admin account 3.1 System prompts Admin for login information for the new account 3.3 System validates login information
	4 System creates and stores a new administrator account with the given login information

Use Case Description: Administrator Removes an Administrator Account

Name: Administrator Removes an Administrator Account

Use Case Number: UC 8

Author: Jesse Bishop

System: Administrative Control

Actors:

- Administrator (Admin)
- Administrative Control Subsystem (System)

Event/Precondition:
Admin requests an administrator account be deleted

Overview/Postcondition:
The administrator account selected by Admin is deleted

References: R18, M1-66

Related Use Cases:

- UC 04 - Requires Admin to be logged in before deleting an administrator account

Typical Process Description	
Administrator	*System*
1 Admin requests that an administrator account be deleted	
	2 include(UC 04 - Administrator Login Script)
	3 System prompts Admin to select an administrator account to delete
4 Admin selects an administrator account to delete	
	5 System deletes the selected administrator account
	6 If the deleted administrator account is logged in 6.1 System logs out the deleted administrator account

Use Case Description: Performance of Hardware Testing

Name: Performance of Hardware Testing

Use Case Number: UC 22

Author: Adam Nace

System: Maintenance Subsystem

Actors:

- Administrator (Admin)
- Time
- Maintenance Subsystem (System)

Event/Precondition:
Administrator Requests Hardware Tests on All Phones; or
Administrator Requests Hardware Tests on a Specific Extension; or
Elapsed Time since last Scheduled Hardware Test Triggers Hardware Test on All Phones.

Overview/Postcondition:
Hardware Tests have been performed on one or more phones
Some phones may switch status from out-of-service to in-service if they pass all tests
Some phones may switch status from in-service to out-of-service if they fail all tests

References: R27, R28, R29, R32, R35

Related Use Cases:
N/A

Typical Use Case Flow		
Time	*Admin*	*System*
1 Specified Elapsed Time between Hardware Tests has elapsed, so Time triggers execution of hardware tests on all phones		
		2 For All Phones (if initiator requires test on all phones) or for selected phone (if initiator requires test on just one phone): 2.1 System performs hardware connectivity test

		2.2 System verifies that hardware test passed and phone is currently in-services OR that hardware test failed and phone is currently out-of-service
Alternate Flow 1: Admin Requests Test on all Phones		
	1 Admin Requests Test on all Phones	
	Return to Main Flow at step 2	
Alternate Flow 2: Admin Requests Test on a single phone		
	1 Admin Requests Test on a Single Phone	
		2 System prompts for extension of phone to be tested
	3 Admin provides extension of phone to be tested	
	Return to Main Flow at Step 2	

Use Case Description: System Goes Off Line for Maintenance

Name: System Goes Off Line for Maintenance

Use Case Number: UC 23

Author: Adam Nace

System: Maintenance Subsystem

Actors:

- Administrator (Admin)
- System
- Time

Event/Precondition:
Administrator Requests the system to go offline for maintenance

Overview/Postcondition:
The system is offline and all on-going calls are terminated

References: N/A

Related Use Cases:
UC 24 - System Comes On Line after Maintenance

Typical Process Description		
Admin	*System*	*Time*
1 Admin requests system to go off line		
	2 System displays a list of on-going phone calls, with their start times	
	3 System temporarily changes max concurrent calls to zero, disallowing all new calls	
	4 System prompts for on-going call timeout duration	
5 Admin provides a timeout duration in minutes		
	6 If timeout duration is between 0 and 60 minutes 6.1 System schedules disconnection of all calls after timeout duration Else if timeout duration is less then 0	

	6.2 System schedules disconnection of all calls after 0 minutes (immediately) Else if timeout duration is greater then 60 6.3 System schedules disconnection of all calls after 60 minutes (immediately)	
		7 Timeout duration has been reached, triggering the disconnection of any remaining ongoing calls
	8 System disconnects all calls	
	9 System becomes off-line	

3.2.2 Domain Model

A domain model representing the VoIP phone system is presented below as Figure 4. Each class is shown with applicable attributes.

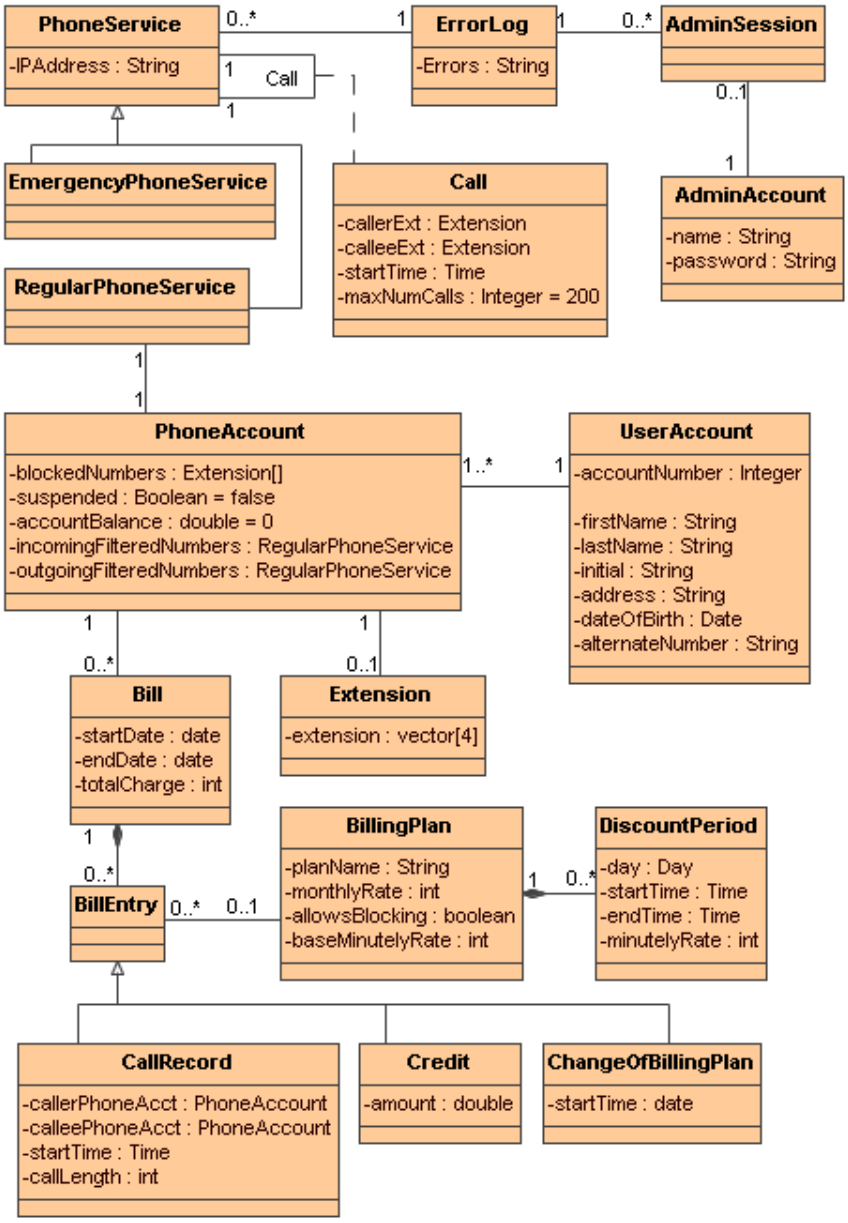


Figure 4: Domain Model of VoIP Phone System

3.2.3 Functional Specifications

This section presents function tables for functions that operate on sets, classes, and attributes specified in the domain model (see Figure 4). Not all functions are included; only those that significantly impact the domain are presented here.

Sets Modified by Functions

The following sets are assumed to exist before the functions below execute. This list is not a comprehensive listing of all required sets; the below are sets which are specifically modified by the functions included in this document.

- BillEntries - a set of bill entries associated with a single bill
- BillingPlans - a set of billing plans
- BlockedNumbers - a set of extensions from which a single PhoneAccount will not accept incoming calls
- CallRecords - a set of billed calls
- Calls - a set of ongoing calls between VoIP phones
- CurrentBills - a set of bills that are in the process of having CallRecords associated with them
- DeletedPhoneAccounts - a list of phone accounts that have been deleted
- DeletedUserAccounts - a list of user accounts that have been deleted
- DiscountPeriods - a set of discount periods associated with a single billing plan
- IncomingFilteredNumbers - a set of extensions from which a PhoneAccount will not accept incoming calls
- OutgoingFilteredNumbers - a set of extensions for which a PhoneAccount will not allow outgoing calls
- PhoneAccounts - a set of active telephone accounts, each of which belong to a user account
- PhoneConnections - a set of open connections to connected VoIP telephones
- UnpaidBills - a set of bills which have not been paid
- UserAccounts - a set of active user accounts

Phone Hardware and Call Processing Functions

The Phone Hardware and Call Processing Functions include the main system operations involved in managing calls between phones and managing the connections between the phones and the server. Only the important functions that modify the domain model are included.

Initialize Call	**ID:** O1	**Importance:** E

Overview: Add new Call to the domain **Inputs:** CallerExt:Extension; CalleeExt:Extension **Preconditions:** Caller's phone is enabled, in service, and not suspended; Caller's phone does not currently have an ongoing call; System is online. **Modifies:** Calls **Postconditions:** Calls' = Calls ∪ {(callerExt, calleeExt, startTime)} **Exceptions:** if calleeExt is not valid or is blocked or filtered by the caller and calleeExt is not an emergency number, play error tone on caller's phone if phone associated with calleeExt is out of service or suspended, then play busy tone on caller's phone if Callee has an ongoing call, then play busy tone on caller's phone if callerExt extension is blocked or filtered by Callee, then play busy tone on caller's phone if |Calls| ≥ maxNumCalls, then play error tone on the caller's phone if Callee does not pick up the phone within 10 rings, then play fast busy tone on caller's phone if Caller hangs up before Callee picks up the phone, cancel the call
References: UC1, M1-5, M1-13, M1-15, M1-24, M1-27, M1-41, M1-67, M1-68, M1-77

Terminate Call	**ID:** O2	**Importance:** E

Overview: Remove a Call from the domain **Inputs:** calleeExt:Extension; callerExt:Extension **Preconditions:** There is a single ongoing call with the extension as either the caller or callee **Modifies:** CallRecords, Calls **Postconditions:** Calls' = Calls - { (calleeExt, callerExt, *) } CallRecords' = CallRecords ∪ { (callerPhoneAcct, calleePhoneAcct, Calls[calleeExt, callerExt].startTime, currentTime - Calls[calleeExt, callerExt].startTime) } **Exceptions:** none
References: UC1, M1-38

Reset Phone Connection	**ID:** O3	**Importance:** E

Overview: This function is used to reset a phone's connection to the system, allowing it to re-establish proper communication, and allowing it to pass the hardware tests. If the phone connection was in an ongoing call at the time, that call is disconnected. **Inputs:** IPAddress:String **Preconditions:** $\exists$phoneIPAddress, phoneIPAddress $\in$ PhoneConnections **Modifies:** Calls, PhoneConnections **Postconditions:** IPAddress $\in$ PhoneConnections if IPAddress $\in$ Calls then IPAddress $\notin$ Calls' **Exceptions:** Cannot Locate IPAddress at all to reconnect to phone, then return error condition Can Locate IPAddress, but cannot re-open connection to phone, then return error condition
References: UC 30

User and Phone Account Management Functions

The User and Phone Account Management Functions include the main system operations involved in creating and deleting user and phone accounts and modifying their properties. Only the important functions that modify the domain model are included.

Create User Account	**ID:** O4	**Importance:** D

Overview: Creates a User Account **Inputs:** accountNumber:Integer; firstName:String; lastName:String; initial:String; address:String; dateOfBirth:Date; alternateNumber:String **Preconditions:** $\nexists$ acct $\in$ UserAccounts : acct(accountNumber) = accountNumber **Modifies:** UserAccounts **Postconditions:** UserAccounts' = UserAccounts $\cup$ { (accountNumber, firstName, lastName, initial, address, dateOfBirth, alternateNumber) } **Exceptions:** none
References: UC12, M1-19

Delete User Account	**ID:** O5	**Importance:** D

Overview: Moves a user account to the list of deleted user accounts and cancels all of the user's phone accounts **Inputs:** accountNumber:Integer **Preconditions:** $\neg$ UserAccounts[accountNumber](lockedByAdmin) $\forall$phoneacct $\in$ UserAccounts[accountNumber](phoneAccounts), $\neg$phoneacct(lockedByAdmin) $\exists$ acct $\in$ UserAccounts : acct[accountNumber] = accountNumber **Modifies:** DeletedPhoneAccounts, DeletedUserAccounts, PhoneAccounts, UserAccounts **Postconditions:** DeletedUserAccounts' = DeletedUserAccounts $\cup$ { $\forall x \in UserAccounts : x.acctNum = acctNum$ } UserAccounts' = UserAccounts - { (acctNum, *) } DeletedPhoneAccounts' = DeletedPhoneAccounts $\cup$ { $\forall x \in PhoneAccounts : x.acctNum = acctNum$ } PhoneAccounts' = PhoneAccounts - { (acctNum, *) } **Exceptions:** none
References: UC14, M1-38, M1-63, M2-72, M3-4

Create Phone Account	**ID:** O6	**Importance:** E

Overview: Creates a Phone Account **Inputs:** userAcct:UserAccount, extension:Extension, IPAddress:String, billingPlan:BillingPlan **Preconditions:** $\nexists$ phAcct $\in$ PhoneAccounts : phAcct(extension) = extension, billingPlan $\in$ BillingPlans, IPAddress exists **Modifies:** PhoneAccounts **Postconditions:** PhoneAccounts' = PhoneAccounts $\cup$ { (userAcct, extn, IP, billingPlan) } **Exceptions:** if (*, extn, *, *) $\in$ PhoneAccounts || (*, *, IP, *) $\in$ PhoneAccounts then error code "Invalid input" is returned
References: UC13, M3-2, M4-12

Suspend Phone Account	**ID:** O7	**Importance:** D

Overview: Suspends a phone account **Inputs:** extension:Integer **Preconditions:** ¬PhoneAccounts[extension](lockedByAdmin) **Modifies:** PhoneAccounts **Postconditions:** p = PhoneAccounts[extension] : p(suspended) = true **Exceptions:** if $\nexists$ PhoneAccounts[extension], then error code "invalid extension" is returned
References: UC17, M2-49

Create Blocked Number	**ID:** O8	**Importance:** E

Overview: Given valid input, creates a new blocked number for a given phone account. **Inputs:** accountNumber:String; numToBlock:String **Preconditions:** BillingPlans[PhoneAccounts[accountNumber](billingPlan)](AllowsBlocking) **Modifies:** PhoneAccounts[accountNumber](BlockedNumbers) **Postconditions:** PhoneAccounts[accountNumber](BlockedNumbers') = PhoneAccounts[accountNumber](BlockedNumbers) $\cup$ numToBlock **Exceptions:** if ¬ BillingPlans[PhoneAccounts[accountNumber](billingPlan)](allowsBlocking), then PhoneAccounts[accountNumber](BlockedNumbers') = PhoneAccounts[accountNumber](BlockedNumbers) and error code "operation failed" is returned
References: UC3, M1-84, M2-12, M2-16, M2-19, M2-74, M2-75

Delete Blocked Number	**ID:** O9	**Importance:** E

Overview: Given valid input, deletes a blocked number for a given phone account. **Inputs:** accountNumber:String; numToUnblock:String **Preconditions:** BillingPlans[PhoneAccounts[accountNumber](billingPlan)](AllowsBlocking) **Modifies:** PhoneAccounts[acctNum](blockedNumbers) **Postconditions:** PhoneAccounts[acctNum](blockedNumbers') = PhoneAccounts[acctNum](blockedNumbers) - numToUnblock **Exceptions:** if ¬ BillingPlans[PhoneAccounts[accountNumber](billingPlan)](allowsBlocking), then PhoneAccounts[accountNumber](BlockedNumbers') = PhoneAccounts[accountNumber](BlockedNumbers) and error code "operation failed" is returned
References: UC3, M1-84, M2-12, M2-16, M2-19, M2-74, M2-75

Create Filter	**ID:** O10	**Importance:** E

Overview: Given valid input, creates a new filter for a given phone account. **Inputs:** accountNumber:String; filterDirection:String; numToFilter:String **Preconditions:** ∃acct ∈ PhoneAccounts : acct(accountNumber) = accountNumber, ¬acct(lockedByAdmin) filterDirection = "input" ‖ filterDirection = "output" **Modifies:** PhoneAccounts[accountNumber](IncomingFilteredNumbers), PhoneAccounts[accountNumber](OutgoingFilteredNumbers) **Postconditions:** If filterDirection = "input", then PhoneAccounts[accountNumber](IncomingFilteredNumbers') = PhoneAccounts[accountNumber](IncomingFilteredNumbers) ∪ numToFilter If filterDirection = "output", then PhoneAccounts[accountNumber](OutgoingFilteredNumbers') = PhoneAccounts[accountNumber](OutgoingFilteredNumbers) ∪ numToFilter **Exceptions:** none
References: UC25

Credit Account	**ID:** O11	**Importance:** E

Overview: Given valid input, credits a given phone account. **Inputs:** accountNumber:String; creditValue:Double; reasonForCredit:String **Preconditions:** ∃acct ∈ PhoneAccounts : acct(accountNumber) = accountNumber, ¬acct(lockedByAdmin), creditValue ≥ 0 **Modifies:** PhoneAccounts[accountNumber](BillEntries) **Postconditions:** PhoneAccounts[accountNumber](accountBalance)' = PhoneAccounts[acctNum](accountBalance) + creditValue ∃ credit : BillEntries[numEntries] = Credit(creditValue) **Exceptions:** none
References: UC20, R51, M1-16, M1-31

Billing and Billing Plan Management Functions

The Billing and Billing Plan Management Functions include the main system operations involved in managing billing plans and generating bills. Only the important functions that modify the domain model are included.

Create Billing Plan	**ID:** O12	**Importance:** E

Overview: Given valid input, creates a new billing plan record **Inputs:** planName:String; monthlyRate:Integer; allowsBlocking:boolean; baseMinutelyRate:Integer **Preconditions:** $\nexists$plan $\in$ BillingPlans : plan(planName) == planName **Modifies:** BillingPlans **Postconditions:** BillingPlans' = BillingPlans $\cup$ { (planName, monthlyRate, allowsBlocking, baseMinutelyRate) } **Exceptions:** if monthlyRate $<$ 0 $\|$ baseMinutelyRate $<$ 0, then BillingPlans' = BillingPlans
References: UC16, M2-15, M2-16

Delete Billing Plan	**ID:** O13	**Importance:** D

Overview: Given the name of a valid billing plan, deletes it. **Inputs:** planName:String **Preconditions:** $\exists$plan $\in$ BillingPlans : plan(planName) == planName, $\neg$plan(lockedByAdmin) **Modifies:** BillingPlans, InactivePlans **Postconditions:** BillingPlans' = BillingPlans - BillingPlans[planName], if subscribesTo[planName] $\neq$ NULL then InactivePlans' = InactivePlans $\cup$ BillingPlans[planName] **Exceptions:** If the preconditions are not met, BillingPlans' = BillingPlans and InactivePlans' = InactivePlans.
References: UC21, M2-21

Generate Bill	**ID:** O14	**Importance:** E

Overview: Given a bill in the system, this calculates its total and sets up a new bill
Inputs: b:Bill; today:Date
Preconditions: $\exists$ cbp $\in$ b(billEntries), today is the current date
Modifies: CurrentBills, UnpaidBills
Postconditions: $\exists$b' $\in$ Bills : b'.totalCharge = (($\sum$ cbp : ChangeOfBillingPlan $\in$ BillEntries[b] : cbp.BillingPlan.monthlyRate * percentage of month cbp was active * percentage of active period b.PhoneAccount was unsuspended and enabled) + ($\sum$ cr : CallRecord $\in$ BillEntries[b] : cr.callLength * appropriate minutelyRate from cr.BillingPlan) - ($\sum$ c : Credit $\in$ BillEntries[b] : c.amount))
b'.endDate = today
CurrentBills' = (CurrentBills - b) $\cup$ { (today, NULL, 0) }
UnpaidBills' = UnpaidBills $\cup$ { b' }
Exceptions: If any preconditions are not met, CurrentBills' = CurrentBills and UnpaidBills' = UnpaidBills
References: UC19, R49, R50, M1-7, M1-43

Create Discount Period	**ID:** O15	**Importance:** E

Overview: Given valid data for a discount period, and a billing plan to add it to, this adds it.
Inputs: day:Day; endTime:Time; minutelyRate:Integer; planName:String; startTime:Time
Preconditions: $\exists$plan $\in$ BillingPlans : plan(planName) = planName, $\neg$plan(lockedByAdmin)
Modifies: DiscountPeriods
Postconditions: DiscountPeriods[planName]' = DiscountPeriods[planName] $\cup$ { (day, startTime, endTime, minutelyRate) }
Exceptions: if startTime < endTime $\|$ minutelyRate < 0 $\|$ ($\exists$ p : DiscountPeriod : (startTime < p.startTime && endTime > p.startTime) $\|$ (startTime < p.endTime && endTime > p.endTime)), then DiscountPeriods' = DiscountPeriods
References: UC16, R56, R57, M5-5

Administration Account and Session Management Functions

The Administration Account and Session Management Functions include the main system operations involved in managing the set of administrator accounts and authenticating administrators using the administrator interface. Only the important functions that modify the domain model are included.

Change Admin Password	**ID:** O16	**Importance:** D

Overview: Change the password of an administrator account **Inputs:** name:String; password: String **Preconditions:** $\exists adminName : admins[adminName] = (name)$ **Modifies:** **Postconditions:** AdminAccounts[name](password) = password **Exceptions:** if $\nexists adminName : AdminAccounts[adminName] = (name) \| \neg password \in \{a-z, A-Z, 0-9, !, @, \#, \$, \%, \hat{}, \&, *, (,)\}\{8, \}$, then $admins' = admins$ and error code "operation failed" is returned
References: UC6, M1-85, M2-2, M2-60

Administrator Login	**ID:** O17	**Importance:** D

Overview: Administrator Logs In **Inputs:** name:String; password:String **Preconditions:** true **Modifies:** AdminSessions **Postconditions:** $AdminSessions' = AdminSessions \cup \{(AdminAccounts[name])\}$ **Exceptions:** if $AdminAccounts[name](password) \neq password$, then error code "invalid login" is returned
References: UC4

Create Admin Account	**ID:** O18	**Importance:** D

Overview: Create a new administrator account **Inputs:** name:String; password: String **Preconditions:** $\nexists adminName : admins[adminName] = (name)$ **Modifies:** AdminAccounts **Postconditions:** $AdminAccounts' = AdminAccounts \cup \{(name, password)\}$ **Exceptions:** if $\exists adminName : admins[adminName] = (name) \| \neg password \in \{a-z, A-Z, 0-9, !, @, \#, \$, \%, \hat{}, \&, *, (,)\}\{8, \}$, then $admins' = admins$ and error code "operation failed" is returned
References: UC7, M1-85, M2-2, M2-60

Create Admin Account	**ID:** O19	**Importance:** D

Overview: Delete an existing administrator account **Inputs:** name:String **Preconditions:** $\exists a : a = admins[name]$ **Modifies:** AdminAccounts, AdminSessions **Postconditions:** AdminSessions' = AdminSessions - { (AdminAccounts[name]) }, AdminAccounts' = AdminAccounts - { (name, *) } **Exceptions:** if $\nexists a : a = AdminAccounts[name]$, then $AdminAccounts' = AdminAccounts$ and error code "invalid operation" is returned
References: UC8, M1-66

System Level Functions

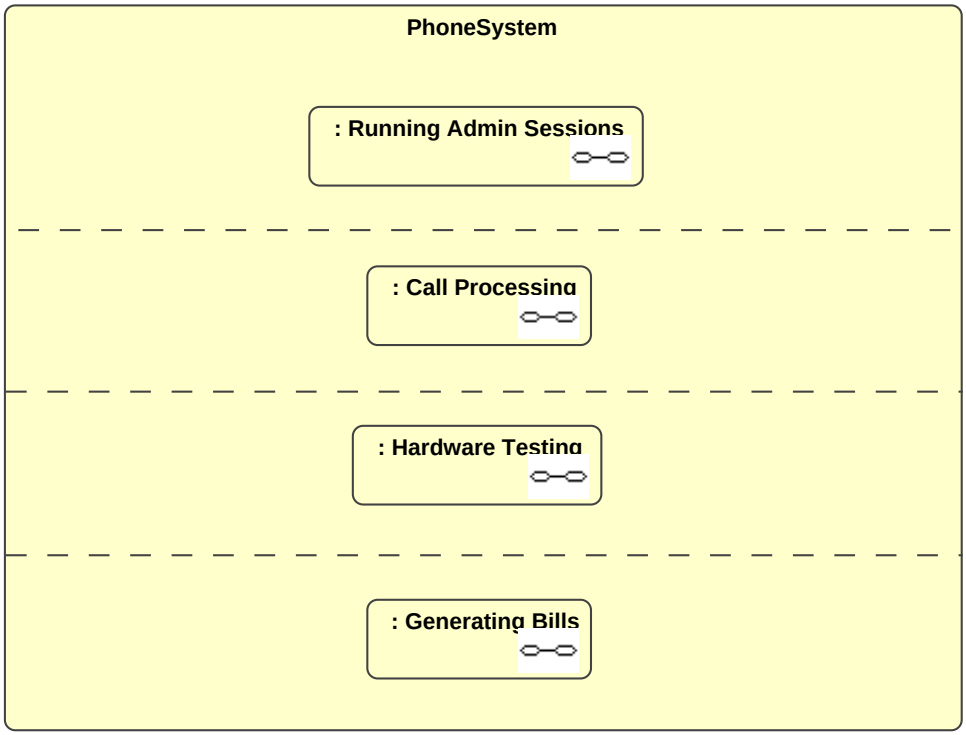
The System Level Functions include the main system operations involved in system-wide actions that are not covered in previous sections. Only the important functions that modify the domain model are included.

Set System Offline	**ID:** O20	**Importance:** D

Overview: Sets system to offline (for maintenance, etc.). Gives ongoing calls a set amount of time to end before terminating them. **Inputs:** Amount of time before terminating ongoing calls **Preconditions:** System is currently online **Modifies:** Calls **Postconditions:** Calls' = $\{\emptyset\}$, systemOnline = false, for each terminated call, a corresponding CallRecord is created **Exceptions:** If timeout value is not between 0 and 60 minutes, the operation is cancelled.
References: UC23, M1-71, M6-1

3.2.4 State Machine Models

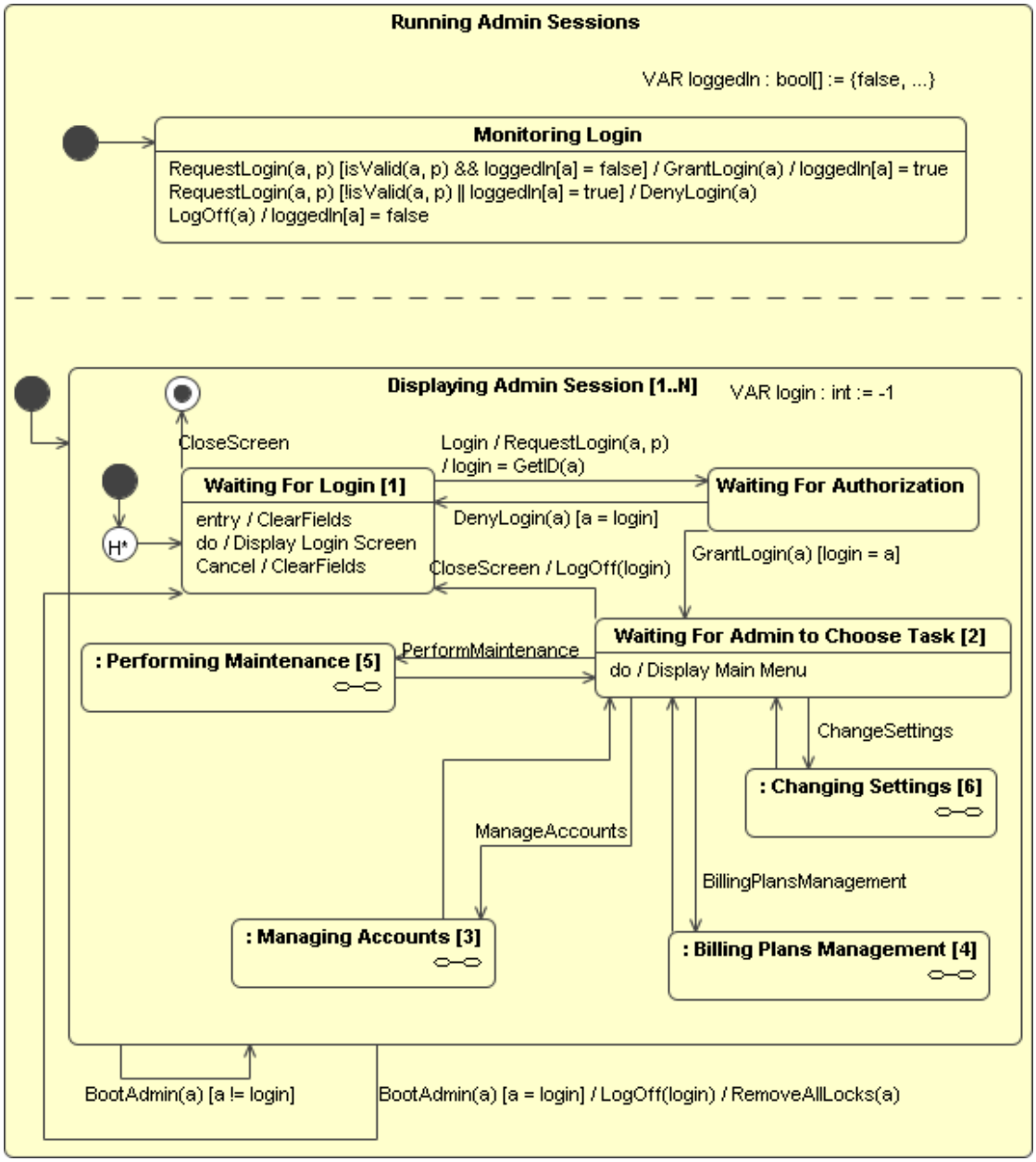
The following state machine (State Machine 1) presents an overview of the four major concurrent responsibilities of the VoIP telephone switching system.



State Machine 1: Overview of Major System Responsibilities

Administrator Session Life Cycle

The Displaying Admin Session state machine (State Machine 2) manages administrators logging in and out. It also acts as a top level gateway to other state machines.



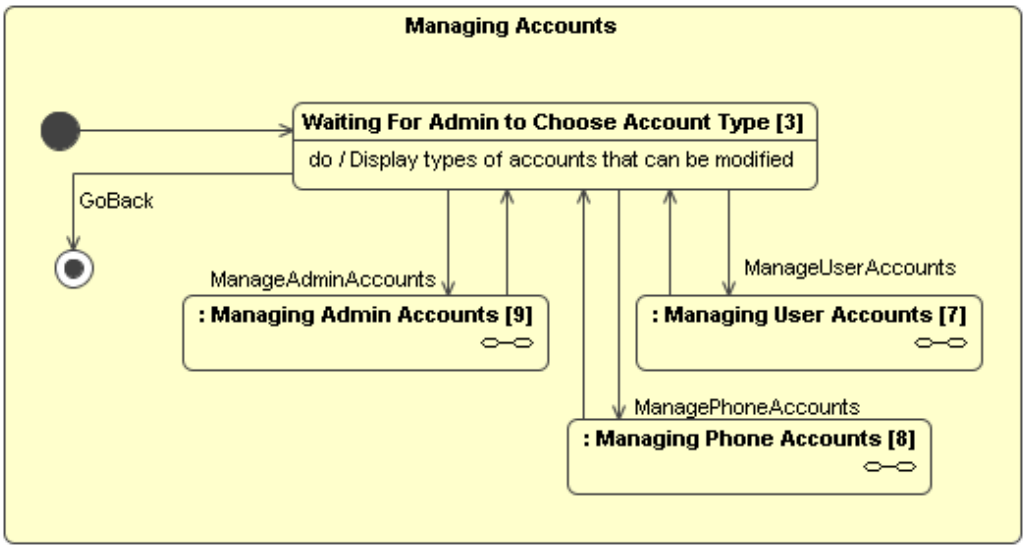
State Machine 2: Administrator Session State Diagram

- RequestLogin(a, p) - This will be produced in response to the Login event. A and p are the admin name, and password respectively (taken from the form's input boxes). This signal is used to keep logging in secure, and to prevent a user from logging in multiple times.
- GetID(a) - Gets an admin's ID based on their login name.
- LogOff(a) - Logs off admin a. This should only ever be sent by admin a.

- isValid(a, p) - Verifies that a and p are valid login credentials.
- GrantLogin(a) and DenyLogin(a) are used to respond to events
- ClearFields - Used to clear the AdminID and password fields on the login screen when the admin either fails to enter the correct credentials, or presses the cancel button.
- BootAdmin(a) - This event is received when an admin has deleted another admin's account.
- RemoveAllLocks(a) - This will remove any locks held in submachines by user (a).

Account Management

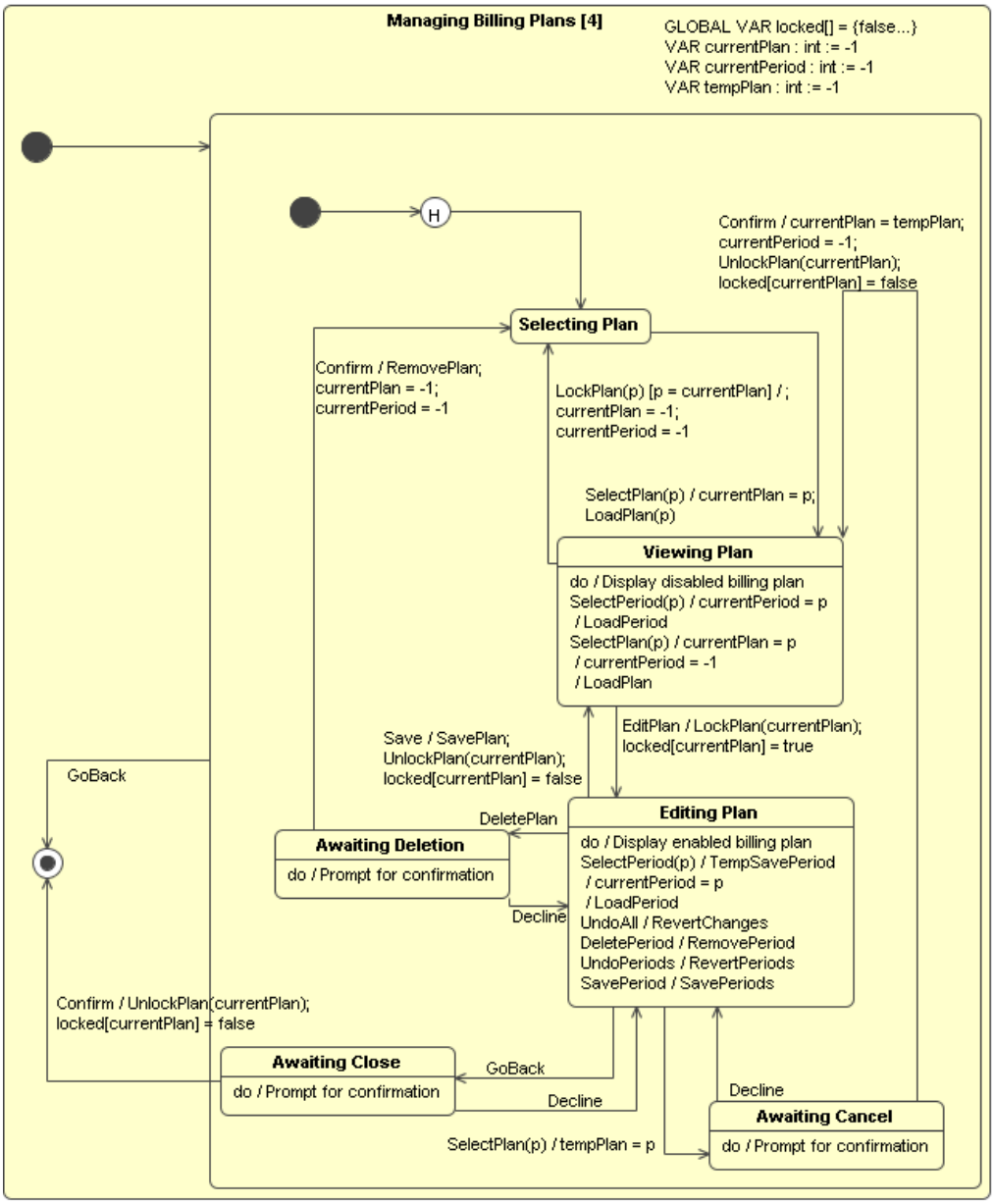
The Manage Accounts state machine (State Machine 3) acts as a top level gateway to a number of composite machines. This state machine will correspond to Form 3 in the GUI.



State Machine 3: Account Management State Diagram

Billing Plans Management

The Managing Billing Plans state machine (State Machine 4) controls how an administrator is able to edit billing plans. The three most important states correspond to not having selected a plan, reading from a plan, and writing to a plan. When an admin is writing to a plan, it prevents them others from reading it.



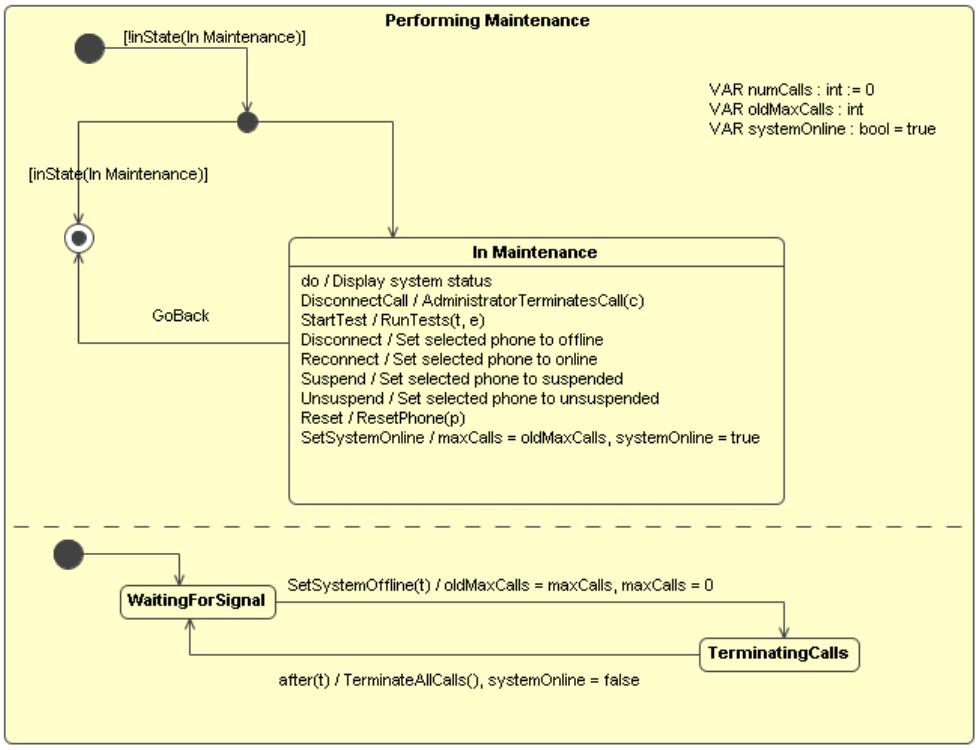
State Machine 4: Billing Plan Management State Diagram

- LockPlan(p) - Sent as a response to the EditPlan event. It causes any other admins' lists to refresh, and kicks off any other admin who was viewing plan p.
- UnlockPlan(p) - Allows admins to once again see plan p.

- LoadPlan - Loads the data for the current plan (i.e. this populates the text boxes and the list of discount periods based on the currentPlan variable). By default, this will put blanks in all the discount periods text boxes.
- LoadPeriod - Loads the data for the current period (based on the currentPeriod variable).
- tempSavePeriod - Temporarily records the contents of the active billing period when the user selects a new one.
- SavePeriod - Saves the contents of any billing plans that have been changed since the last SavePeriod event. (The difference between this and tempSavePeriod is that RevertPeriods will revert as far back as the last SavePeriod, but may span multiple tempSavePeriods). All saving is in a deferred manner. (i.e. the changes will not be permanent until the billing plan itself gets saved)
- SavePlan - Saves the contents of the plan and all of its periods.
- RevertChanges - This action will reload the original values of a record that is being edited.
- RevertPeriods - This action will reload the original value of all billing plans as of the point of the last SavePeriod action.
- RemovePeriod - This actually removes the period from the plan.
- RemovePlan - This actually removes the plan from the system.
- GoBack - This is a signal sent when the user clicks on the screen's "X".
- Confirm - The user confirms when they are being prompted for confirmation on something.
- Decline - The user changes their mind when they are being prompted for confirmation on something.

System Maintenance

The Performing Maintenance state machine (State Machine 5) responds to a number of events, and prevents more than one administrator from performing maintenance at a time.

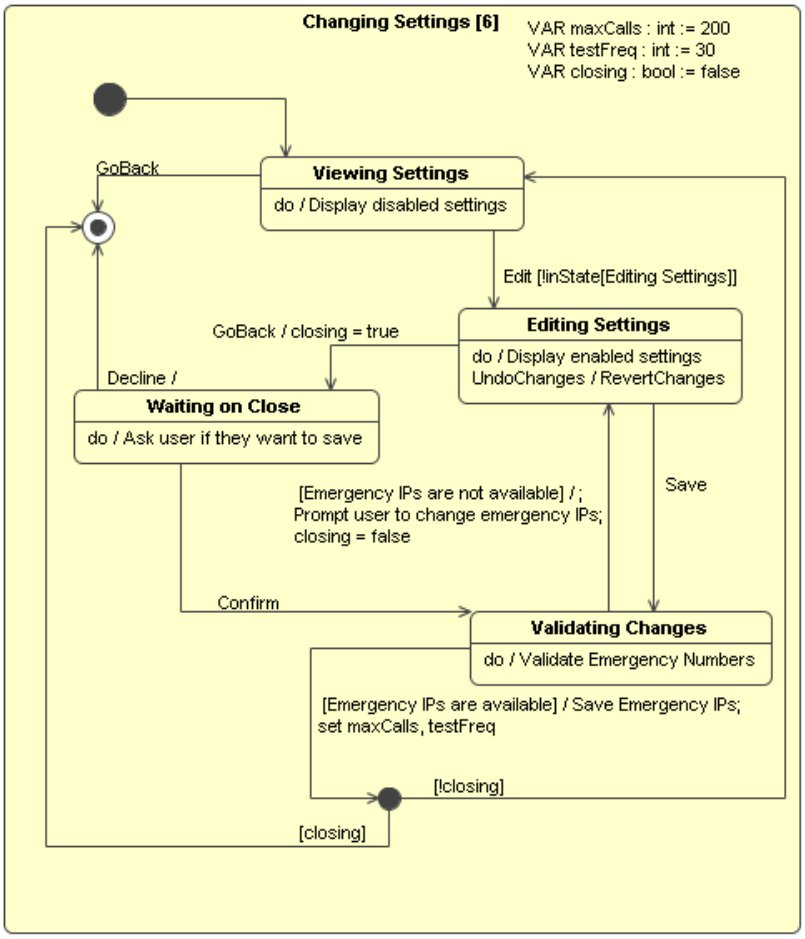


State Machine 5: System Maintenance State Diagram

- In Maintenance activity - The system status to be displayed on this screen includes the status of in-progress calls, an IP/Phone extension listing, and an error log.
- AdministratorTerminatesCall(c) - This signal is sent to the call processing submachine in reaction to DisconnectCall. The parameter 'c' represents the selected phone in the list directly above the button.
- RunTests(t, e) - This function runs the selected hardware tests on the selected phone extensions. The parameter 't' is a list of the tests to run (and could include hardware messages for TestOnHook, TestOffHook, TestDigitPressed, TestDigitReleased, and a request for confirmation of receipt of a message). The parameter 'e' is a list of the extensions to run the test(s) on.
- ResetPhone(p) - This signal causes the phone selected in the IP/Phone Extension Listing to be reset (i.e. active calls are terminated, and the software on the phone restarts).
- TerminateAllCalls() - This signal raises a DisconnectCall event for every entry in the list of calls.

System Settings

The Changing Settings state machine (State Machine 6) shows behaviour for changing system settings. It allows multiple administrators to view the settings, but only one can edit it at a time.

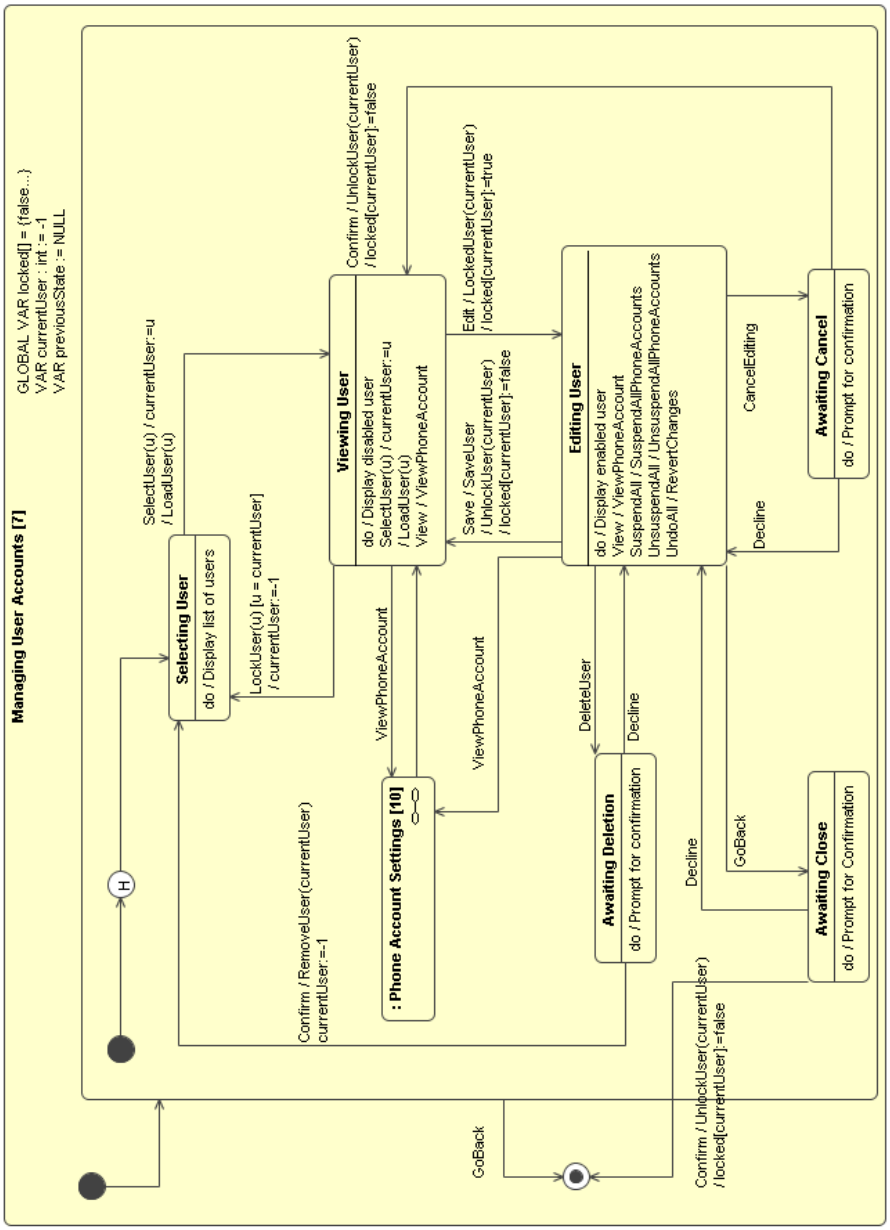


State Machine 6: System Settings State Diagram

- RevertChanges - In response to UndoChanges, this will reload the previous values of all settings.
- Validate Emergency Numbers - Before saving, the system will need to verify that a phone wasn't assigned to either of the proposed emergency IPs while the settings were being edited. Also, the system will need to validate that the emergency extensions are different, and their IPs are different.
- Confirm - This event occurs when the user is asked if they want to save, and say yes.
- Decline - This event occurs when the user is asked if they want to save, and say no.

User Account Management

The Managing User Accounts state machine (State Machine 7) governs how administrators are able to edit user accounts. The three most important states indicate not having selected a user yet, reading from a user account, and writing to a user account. Also, when an administrator is writing to a user account, no other administrators are able to read from it.



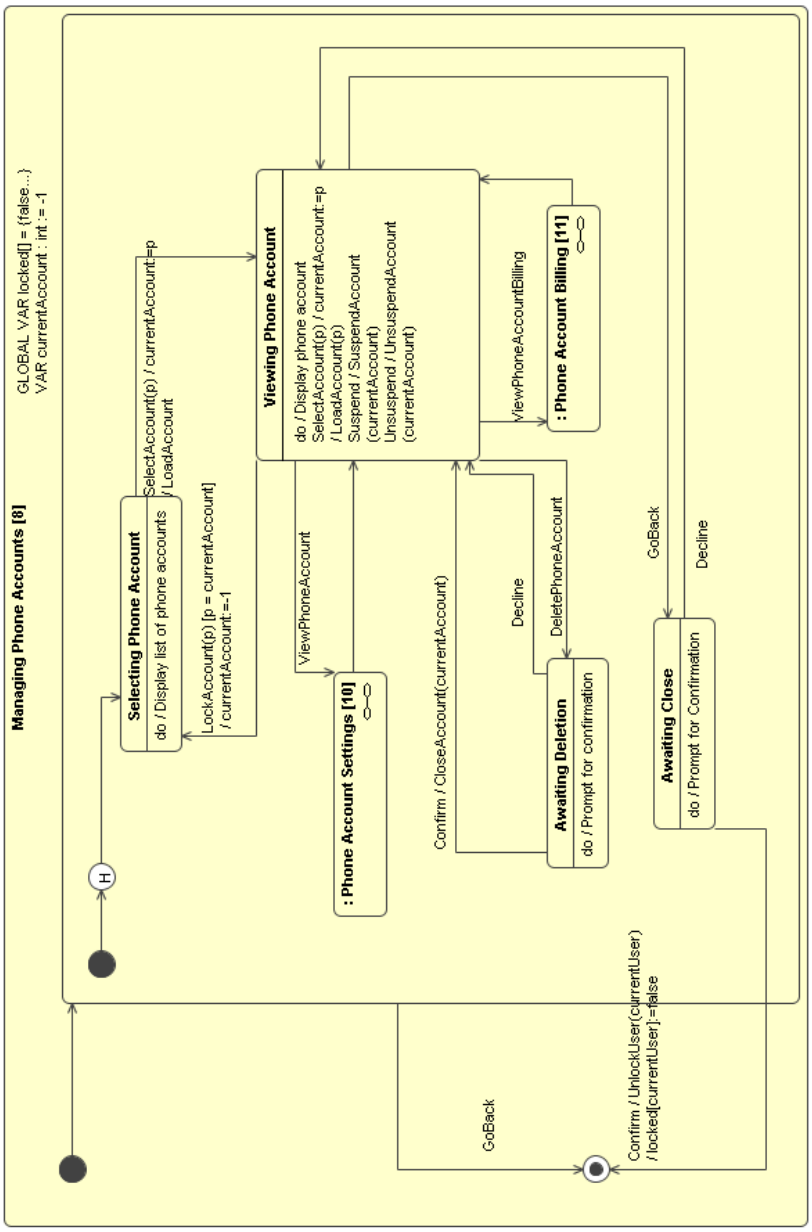
State Machine 7: User Account Management State Diagram

- LockUser(u) - Sent as a signal when a user clicks the edit button. It causes any other admins' lists to refresh, and kicks off any other admin who was viewing user u.
- UnlockUser(u) - Allows admins to once again see user u.

- SelectUser(u) - This signal is caused by the user clicking on an entry in the list to the left (u is already known), or clicking on the '+' button next to the list (to add a new user with a new value of u).
- LoadUser - Loads the data for the current user (i.e. this populates the text boxes and the list of phone accounts based on the currentuser variable).
- SaveUser - Saves the contents of the user and all of its phone accounts.
- UndoAll - This is a signal sent from the "Undo All Changes" button.
- RevertChanges - This action will reload the original values of a record that is being edited.
- SuspendAllPhoneAccounts - This actually suspends all of the user's phone accounts.
- UnsuspendAllPhoneAccounts - This actually unsuspends all of the user's phone accounts.
- RemoveUser - This closes the user's accounts (including phone accounts).
- GoBack - This is a signal sent when the user clicks on the screen's "X".
- Confirm - The user confirms when they are being prompted for confirmation on something.
- Decline - The user changes their mind when they are being prompted for confirmation on something.

Phone Accounts Main Form

The Managing Phone Accounts state machine (State Machine 8) governs how administrators can view and edit phone accounts. It acts as a top level gateway with some functionality, and allows access to more specialised functions in the phone account settings and phone account billings state machines.



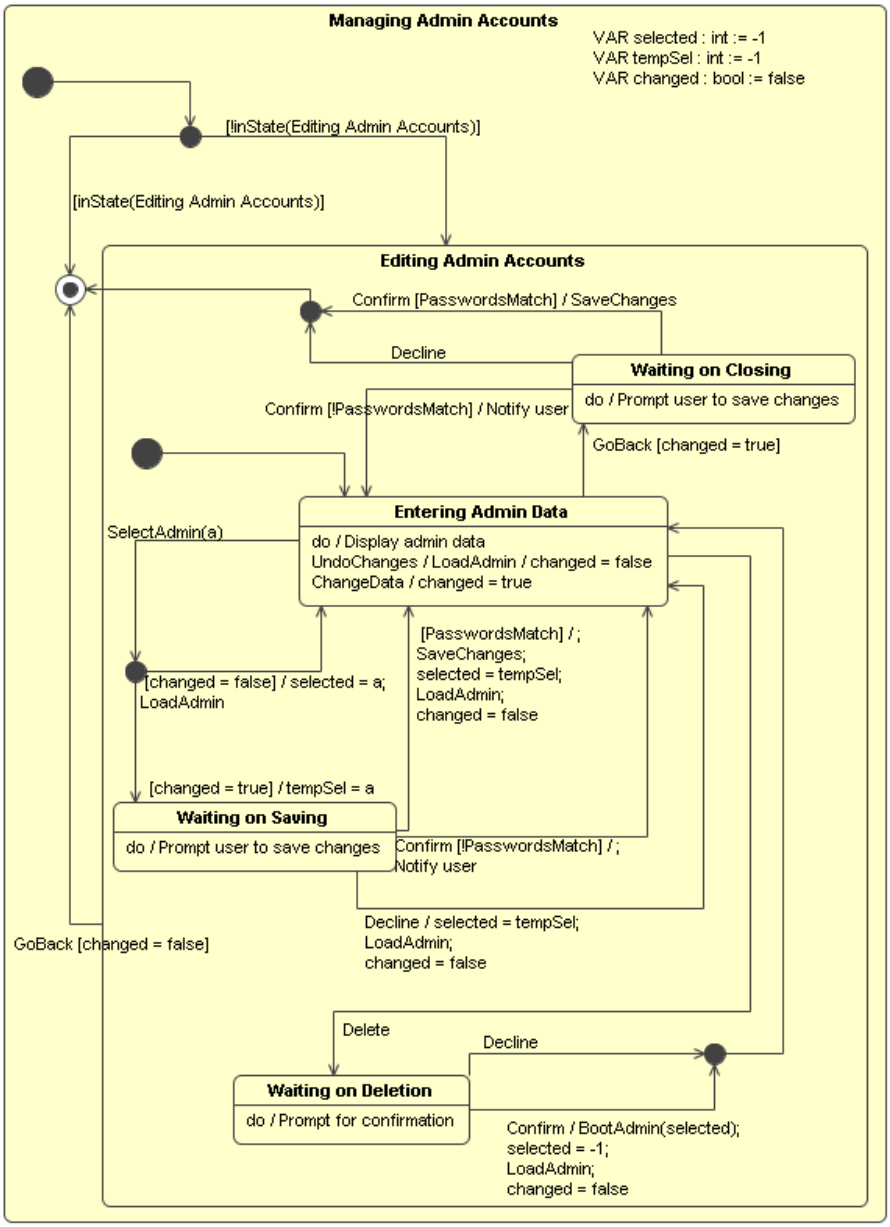
State Machine 8: Phone Account Management State Diagram

- LockAccount(p) - Sent as a signal when a user clicks the edit button. It causes any other admins' lists to refresh, and kicks off any other admin who was viewing Phone Account p.
- UnlockAccount(p) - Allows admins to once again see Phone Account p.

- LoadAccount(p) - Loads the data for phone account p (i.e. this populates the two labels showing the user and extension associated with a phone account).
- SuspendAccount(p) - this action will suspend phone account p.
- UnsuspendAccount(p) - this action will unsuspend phone account p.
- CloseAccount(p) - This action marks phone account p as closed. It sets the billing date on the account to the current day, or the following day if it is past 5 pm.
- Confirm - The user confirms when they are being prompted for confirmation on something.
- Decline - The user changes their mind when they are being prompted for confirmation on something.

Administrator Account Management

The Managing Admin Accounts state machine (State Machine 9) governs how administrators can edit admin accounts. The key feature of the state machine is that it prevents multiple administrators from editing admin accounts simultaneously.



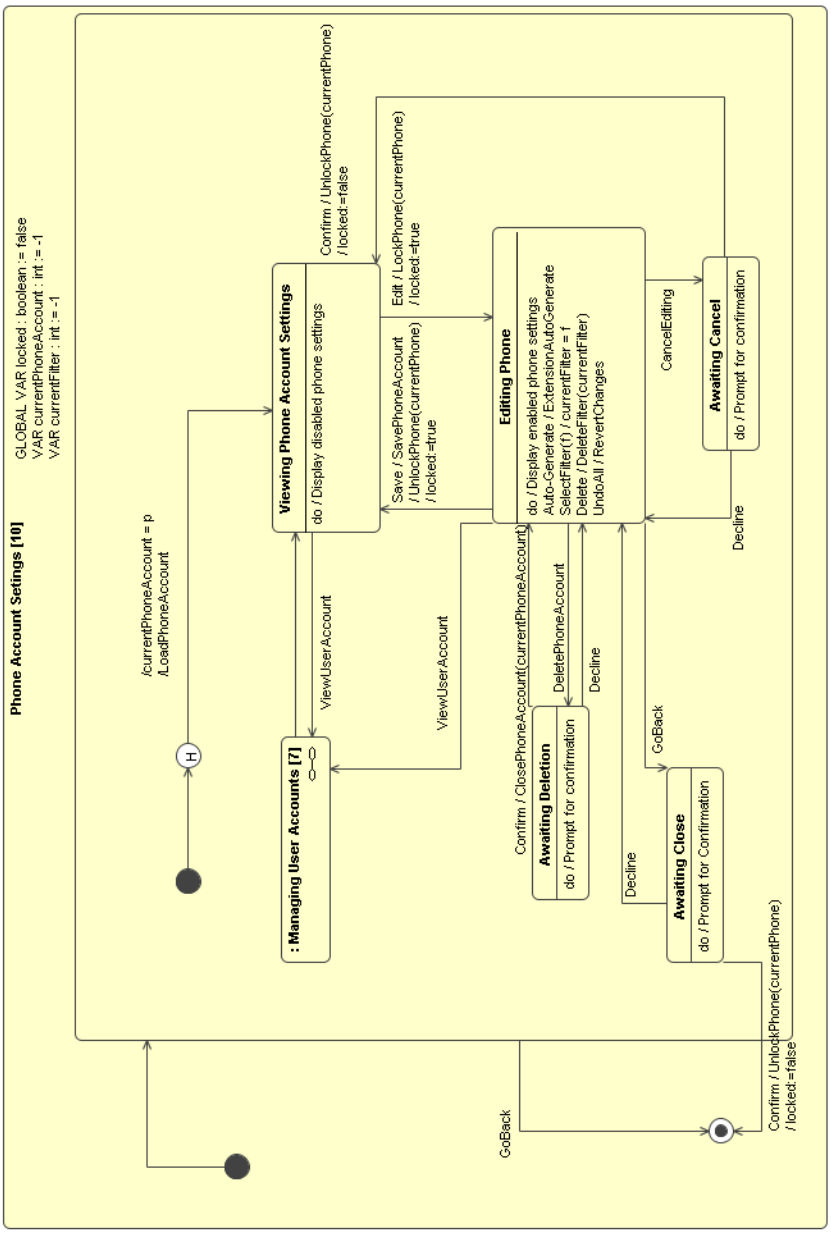
State Machine 9: Administrator Account Management State Diagram

- LoadAdmin - Loads the values of the currently selected admin.
- SaveChanges - Saves the current values of the currently selected admin.
- PasswordsMatch - Determines if the contents of the two password fields match.

- BootAdmin(a) - This is a signal to the rest of the system to boot the admin that was just deleted if they are currently in the system.

Phone Account Settings

The Phone Account Settings state machine (State Machine 10) governs how administrators can edit phone accounts. There are two main states, viewing the phone account, and editing the phone account. This screen can only be accessed directly by the Phone Accounts state machine, so there will always be a phone account selected (unlike the user account and billing plan machines).



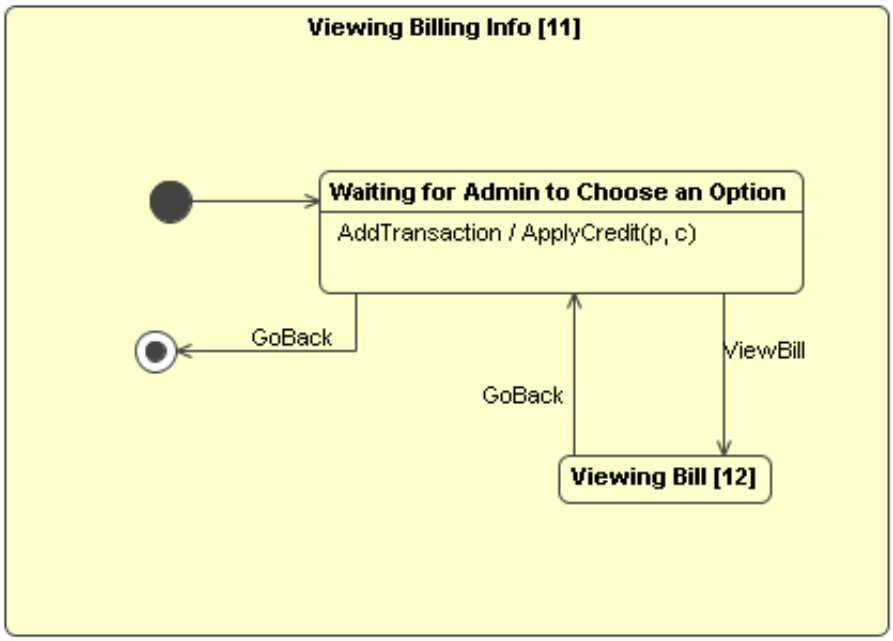
State Machine 10: Phone Account Management State Diagram

- LockPhone(p) - Sent as a signal when a user clicks the edit button. It causes any other admins' lists to refresh, and kicks off any other admin who was viewing Phone p.
- UnlockPhone(p) - Allows admins to once again see Phone p.

- LoadPhoneAccount - Loads the data for the current Phone Account (i.e. this populates the text boxes and the list of filters.
- SavePhoneAccount - Saves the contents of the PhoneAccount and all of its filters.
- SavePhoneAccount - this action saves the phone account and its filters.
- RevertChanges - This action will reload the original values of a record that is being edited.
- DeletePhoneAccount - This is a signal from the "Delete" button on the Phone account section
- ClosePhoneAccount(p) - This marks phone account p as closed. It will also adjust the billing date to the current day, or the following day if it is past 5 pm.
- ExtensionAutoGenerate - This sets the phone account's extension to some extension that is available in the system. If there are not any extensions available, it displays an error to the user.
- Confirm - The Phone confirms when they are being prompted for confirmation on something.
- Decline - The Phone changes their mind when they are being prompted for confirmation on something.

Phone Account Billing

The Viewing Billing Info state machine (State Machine 11) governs how administrators can view billing info.

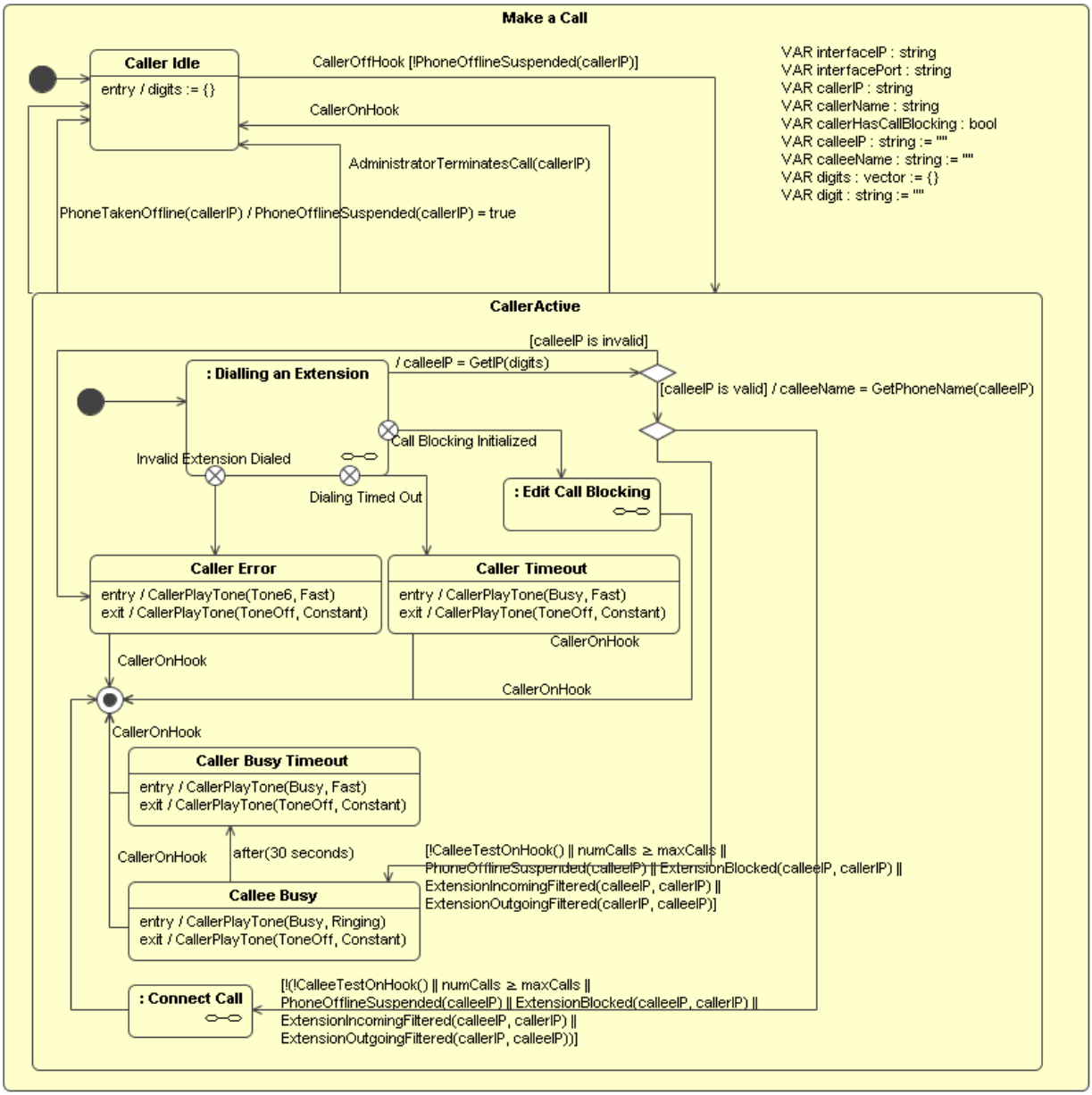


State Machine 11: Phone Account Billing Diagram

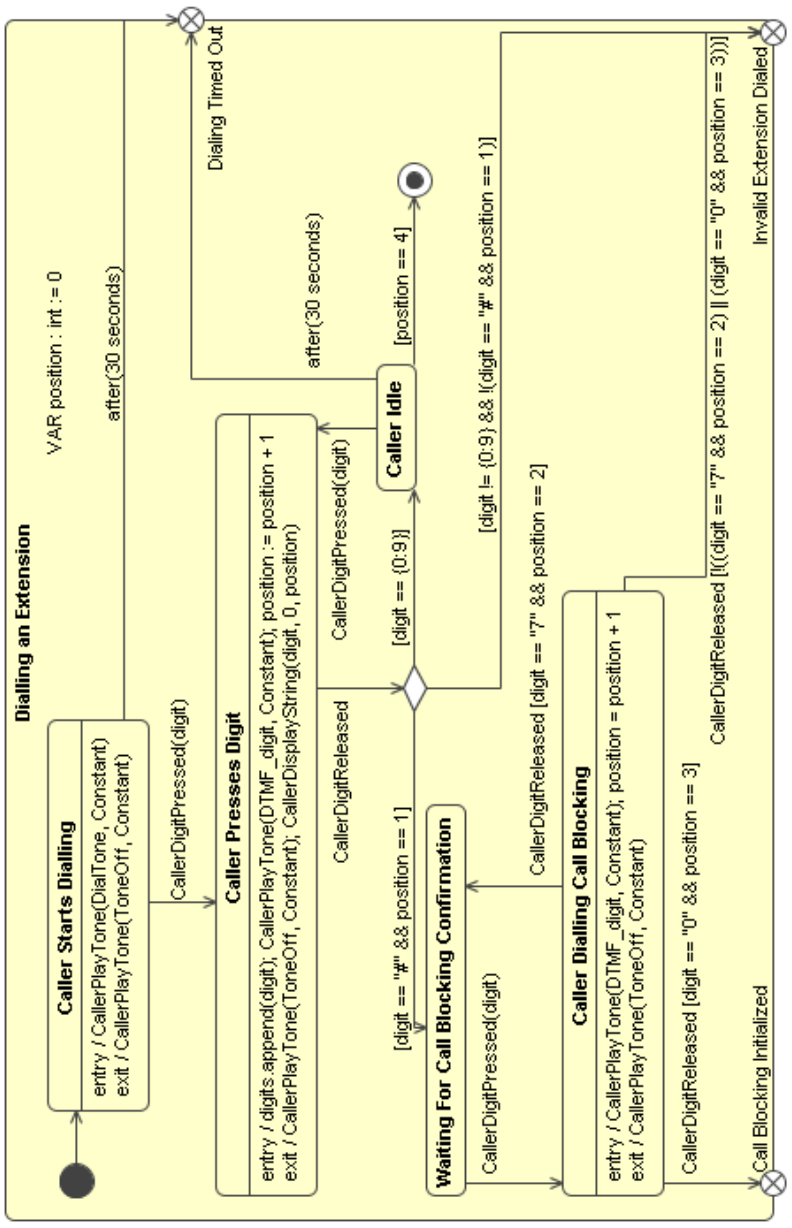
- ApplyCredit(p, c) - This function applies a credit of size c to the balance on the phone account p.

Making a Call

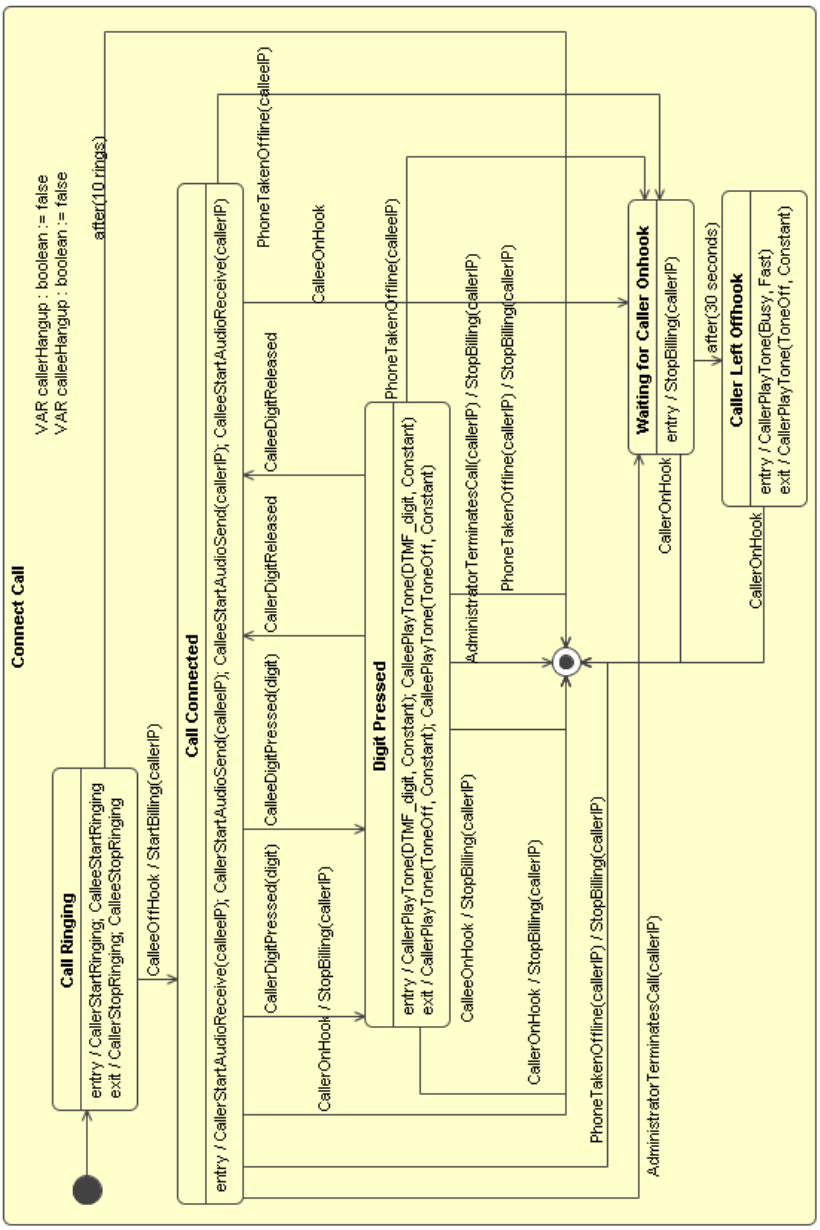
Making a call encompasses four major state machines. The Make a Call state machine (State Machine 12) includes the Sub State Machines for Dialing An Extension (State Machine 13) and for Connecting a Call (State Machine 14), the functions of which follow logically from their names. The fourth state machine is the Receive a Call state machine (State Machine 15), which, of course, describes the receiving phone.



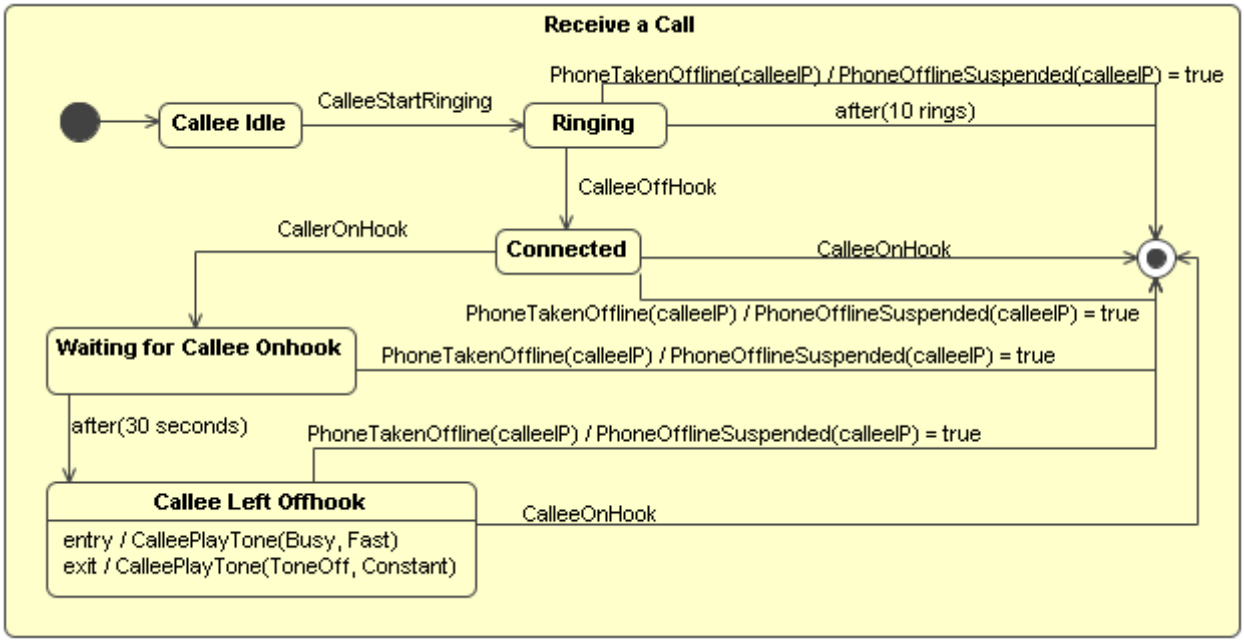
State Machine 12: Make a Call State Diagram



State Machine 13: Dialing an Extension State Diagram



State Machine 14: Connecting a Call State Diagram



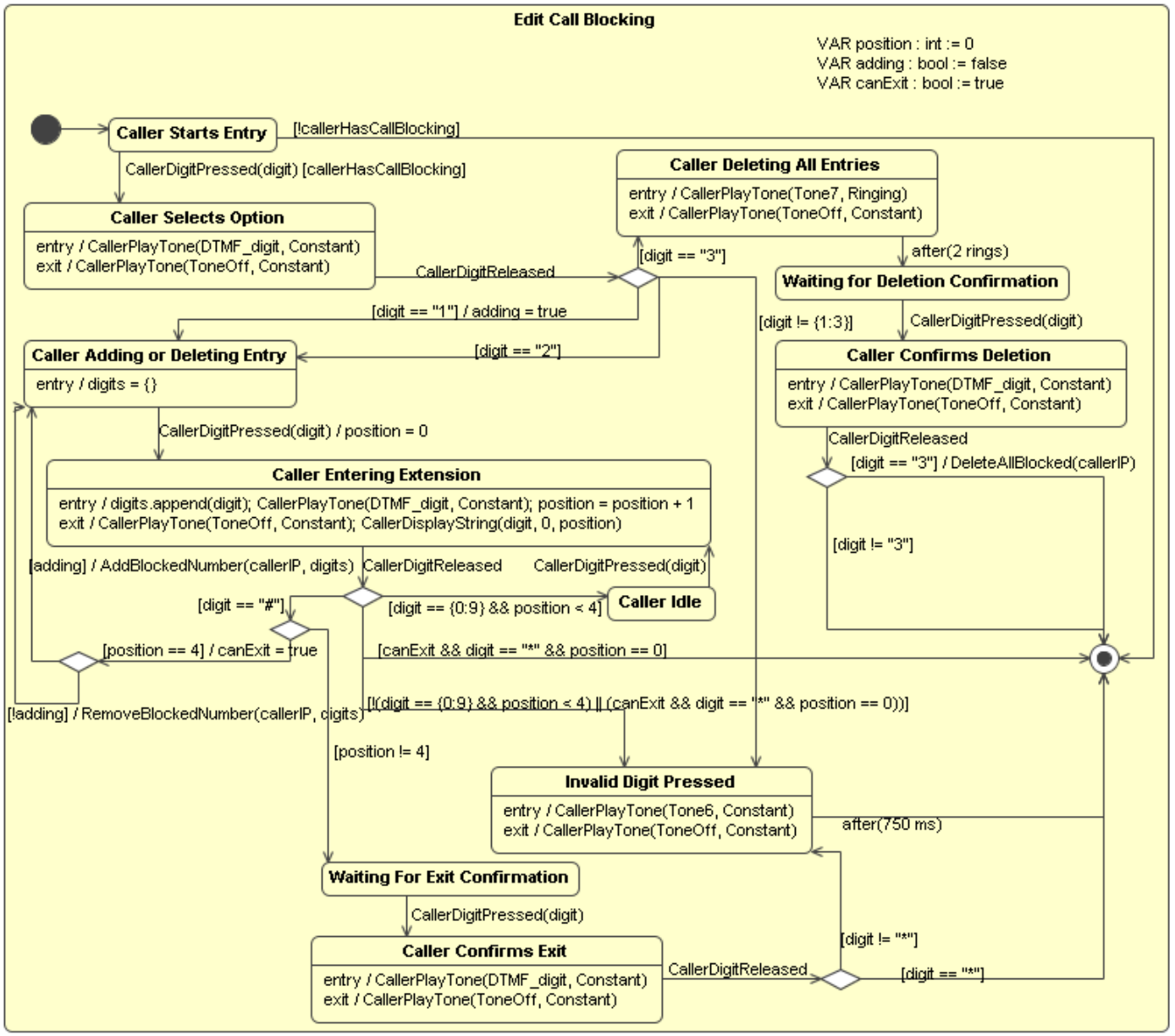
State Machine 15: Receive a Call State Diagram

The following actions and events are generated and sent to other parts of the state machine:

- GetPhoneName(phoneIP) - Returns the name of the owner of the phone account.
- StartBilling(phoneIP) - Records the start of a call in the billing system.
- StopBilling(phoneIP) - Records the end of a call in the billing system.
- PhoneOfflineSuspended(phoneIP) - Indicates that a currently offline phone (mapped to phoneIP) has been suspended.
- AdministratorTerminatesCall(phoneIP) - Indicates that an admin has forcibly terminated a call involving the phone at phoneIP.
- PhoneTakenOffline(phoneIP) - Indicates that an admin or a hardware test has changed the phone at phoneIP's status to offline.

Edit Call Blocking

The Edit Call Blocking state machine (State Machine 16) describes the functionality of the call blocking section of the system. The actions below are executed and sent to other parts of the state machine:



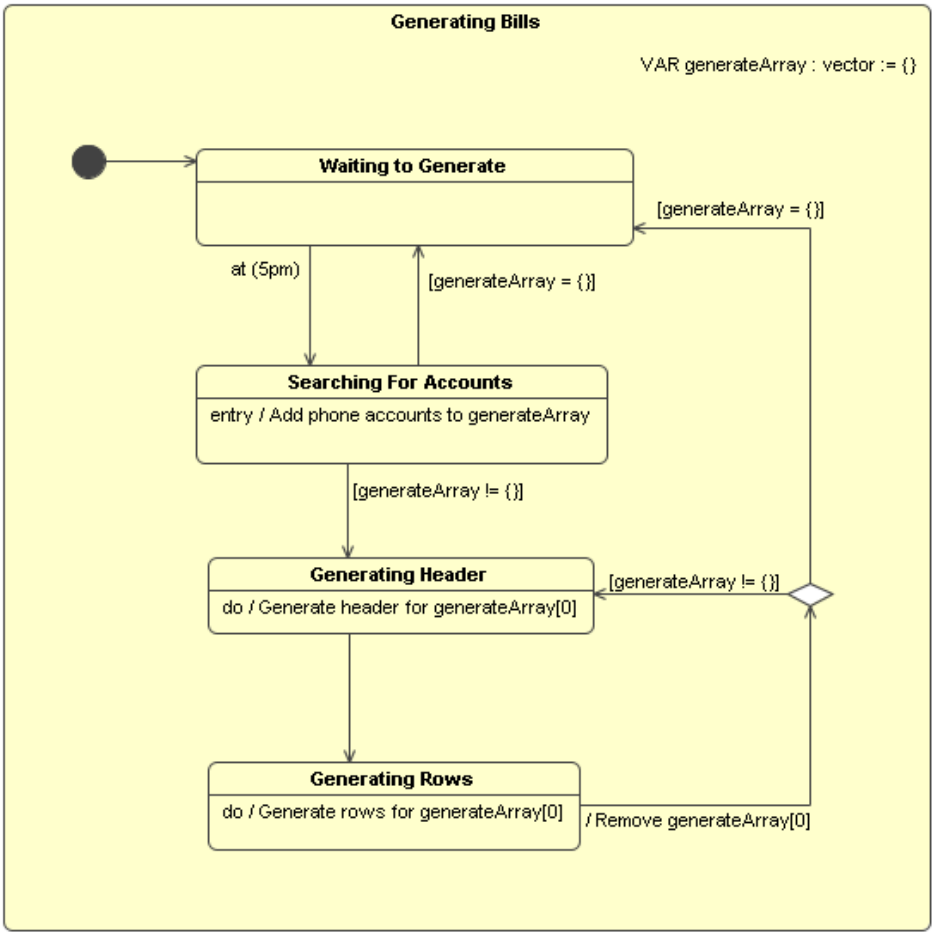
State Machine 16: Editing Call Blocking State Diagram

- AddBlockedNumber(phoneIP, digits) - Adds a blocked number with the given digits to the phone currently mapped to phoneIP.
- RemoveBlockedNumber(phoneIP, digits) - Reverses the previous action.

- DeleteAllBlocked(phoneIP) - Removes all blocked numbers on the phone mapped to phoneIP.

Bill Generation

Bill generation is represented as a separate state machine (State Machine 17), running in parallel to the administrator UI. Whenever an admin requests that a bill be generated, or it is 5 o'clock, this state machine activates, and generates a bill for any accounts that it is billing day for.

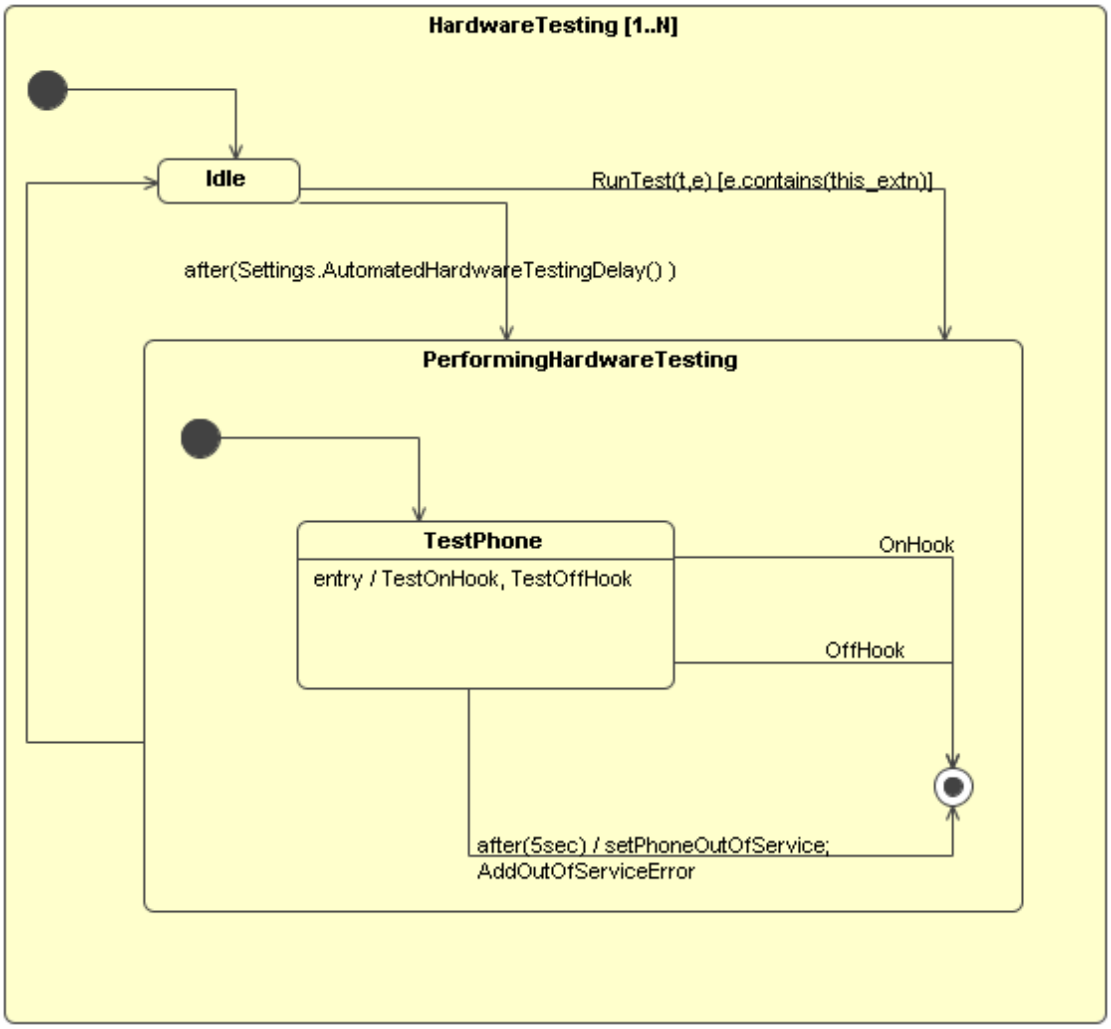


State Machine 17: Monthly Generation of Phone Bills State Diagram

- Searching for accounts action - Each phone account will have a billing day associated with it. When this activity runs, it searches through all the accounts and adds any that have a billing day of the current day to generateArray.
- Generating header activity - This will generate a header for a bill to be mailed to the owner of the phone account. Details include the user's name and address, and if their bill is overdue.
- Generating rows activity - This will generate rows for the bill. Details are printed for each call in the billing period, including the caller, the callee, the start time, the length, and the cost per minute.

Hardware Testing

The Hardware Testing state machine (State Machine 18) governs the process of performing a hardware test on a single phone.



State Machine 18: Hardware Testing State Diagram

- RunTests(t,e) - This event is received from the administrator UI when an administrator requests a manual hardware test (t is a list of tests to run and e is a list of extensions to test)
- setPhoneOutOfService - Sets the phone being tested to out of service because a test has timed out on it.
- AddOutOfServiceError - Adds an entry to the system's error log, and notifies an administrator if there is one currently logged on.

3.3 Performance

The following requirements describe constraints imposed on the performance of the system and provide fit criteria to be used when evaluating each requirement.

Number of Extensions

The system should be scalable to support 10000 extensions, including emergency numbers, under maximum load, and 3 to 9999 under normal load. All phone accounts associated with any extension should be able to be active at the same time. The system implementation would be considered outstanding in this area if it were able to support 50000 extensions under maximum load (see M7-1).

Number of Administrators

The system is designed to support 15 active administrator accounts, with any 10 administrator sessions concurrently active under maximum load (see M2-65). There should be no distinguishable degradation in administrator interface responsiveness while operating within this range.

Number of Concurrent Calls

The system is intended to support a default maximum load of 200 concurrent calls, and a default normal load of 0 to 199 calls (see M1-45). The maximum number of allowed concurrent calls is configurable by an administrator; the expected performance of the system under load is hardware-dependent and therefore outside the scope of this document.

Information Processing

The system is intended to concurrently support administrator interface requests and call processing requests. The system should be able to concurrently handle any combination of these requests within the bounds specified in the preceding sections.

3.4 Design Constraints

The following list of requirements describe design constraints imposed on the system:

- The system must communicate with the phones over sockets on an IP network via the interface described in the Hardware Interface Description document.
- The system must use a client-server architecture.
- The system must include a graphical user interface (rather than a text-based interface) for the client's personnel.

- Addresses must be treated as one long string instead of a series of strings.
- The GUI must be designed in a hierarchical fashion.

3.5 Quality Attributes

The following nonfunctional requirements describe additional quality constraints that may be used to evaluate the system, and provide quantitative fit criteria that specify the extent to which they must be met. As these are not functional requirements of the system and are therefore imperfectly explicable, upper (outstanding) and lower (minimum) criteria are presented along with a target criterion for each requirement.

Dial Tone Response Speed	**ID:** NF1	**Importance:** E

Overview: When a caller picks up a phones headset, the system should issue a dial tone within the specified deadline.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	0s	0.5s	1s
References: M7-1			

Call Connection Speed	**ID:** NF2	**Importance:** E

Overview: When a callee picks up a phones headset when it is ringing, the system should connect the caller and callee within the specified deadline.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	0s	0.1s	0.5s
References: M7-1			

Error Tone Response Speed	**ID:** NF3	**Importance:** E

Overview: When a caller performs an action requiring an error tone, the system should play the error tone within the specified deadline.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	0s	0.1s	0.5s
References: M7-1			

Imitation of Existing Systems	**ID:** NF4	**Importance:** E

Overview: In a focus group of average customers, a defined percentage of subjects should not be able to differentiate between the VoIP phone system and the Canadian land-line phone system when comparing features that are common to both.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	100%	80%	60%
References: M1-13, M5-3, M7-1			

Bill Readability	**ID:** NF5	**Importance:** D

Overview: In a focus group of average customers, a defined percentage of subjects should have no difficulties reading their phone bill.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	100%	80%	60%
References: M1-34, M7-1			

Error Tone Distinguishability	**ID:** NF6	**Importance:** E

Overview: In a focus group of average customers, a defined percentage of subjects should be able to recognize specific error tones after reading the user manual.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	100%	80%	60%
References: M1-41, M7-1			

Password Security	**ID:** NF7	**Importance:** O

Overview: Administrator passwords should be stored by the system using a secure algorithm with a defined number of bits of encryption.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	512 bits	256 bits (SHA-256)	128 bits (MD5)
References: M2-54, M7-1			

Administrator Interface Usability	**ID:** NF8	**Importance:** O

Overview: Administrators should be able to learn how to perform all necessary functions within a defined amount of time, assuming that they have experience in the domain of administering phone systems and that they are computer-literate.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	1 day	1 week	2 weeks
References: M2-1, M3-8, M3-9, M7-1			

Special Number Response Time	**ID:** NF9	**Importance:** E

Overview: When the caller dials a special number, the system should provide feedback within a defined period of time.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	0s	0.1s	0.5s
References: M2-41, M7-1			

System Reliability	**ID:** NF10	**Importance:** E

Overview: The system should be reliable enough to allow it to run a defined percentage of the time.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	100% uptime	99.9% uptime	99% uptime
References: M2-57, M7-1			

Sound Quality	**ID:** NF11	**Importance:** E

Overview: The average call time between audible flaws in transmission should be greater or equal to a defined value.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	100min	10min	5min
References: M2-58, M2-71, M7-1			

Error Log Readability	**ID:** NF12	**Importance:** D

Overview: In a focus group of computer-literate administrators with all required domain knowledge, a defined percentage of subjects should have no troubles understanding a set of error messages.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	100%	80%	60%
References: M2-64, M7-1			

Time to Learn System Maintenance	**ID:** NF13	**Importance:** D

Overview: Given a finished implementation of the VoIP phone system, a experienced developer should be able to learn enough about the system to effectively maintain it within a defined period of time.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	2 weeks	4 weeks	6 weeks
References: M7-1			

Time to Implement New Minor Feature	**ID:** NF14	**Importance:** D

Overview: Given a finished implementation of the VoIP phone system, an experienced developer should be able to implement and test a minor feature within a defined period of time.			
Fit Criteria:	**Outstanding**	**Target**	**Minimum**
	1 month	2 months	3 months
References: M7-1			

Appendix A: Glossary

Administrator Session: (Administrative Control) An instance of an administrator using the administrator interface GUI, associated with one administrator login on one computer.

Bill: (Output) A printed form containing billing info that is mailed to customers.

Billing Info: (Variables) A set of data that can be used to calculate how much any given customer should be charged for a phone account in a given month. This includes the monthly fee, what percentage of the month the phone was in service (used to pro-rate the monthly fee), and a list of calls (including call start time, call length, person called, and minutely rate).

Billing Plan: (Class) A pricing scheme for phone accounts consisting of a monthly rate, a possible allowance for call filtering, and zero or more discount periods.

Call: (Activity) A connection between two phones that occurs when a customer dials a correct phone extension that is in service, and is not already participating in another call.

Contact Information: (User Account Data) Synonym: Customer Info
The user contact information is the information used to define a customer, including their name, address, and other telephone numbers.

Disabled (Data): (State) Describes data (user, phone, billing plan, etc.) that an administrator can currently see, but cannot edit. The data will become enabled when the admin begins editing.

Disabled (Phone): (State) Synonym: Offline
The opposite of enabled. A disabled phone will have no record in the system, and therefore no extension.

Discount Periods: (Class) A period of time during the day where a billing plan specifies a cost per minute.

Emergency Number: (Phone Number) A phone extension that is monitored as a source of emergency services (similar to 911 in the public phone system). Two emergency numbers are currently planned, 9911 and 4911.

Enabled (Data): (State) Describes data that an administrator is currently editing. Other administrators cannot view this data.

Enabled (Phone): (State) Synonym: Online
A phone that is enabled communicates with the system.

Error Log: (Class) A collection of reports of hardware tests that returned negative results.

Hardware Test: (Action) A request by the server to a phone for data. The request could be to confirm connectivity, to test if the phone is on or off the hook, and to test if a given digit is or is not pressed.

In Service: (State) Normally used to refer to a phone that is enabled, and is passing hardware

tests.

Out of Service: (State) Normally used to refer to a phone that is being monitored by the system, but has failed a hardware test. (Note: Out of service/In service status can be overridden by an administrator, regardless of whether hardware tests pass).

Special Number: (Phone Number) A phone extension that serves a purpose other than connecting a call. There is currently only one, the extension #70, which is used to block numbers.