CS 445 Software Requirements Specification OAM Software for SX4

Troy Gonsalves 97083748 tgonsalves@student.math.uwaterloo.ca
Chris Mennie 97024327 camennie@student.math.uwaterloo.ca
Dave Tapuska 97084449 dftapusk@student.math.uwaterloo.ca

December 2, 2000

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This document is a specification for the software system that controls Operations, Administration and Maintenance (OAM) of a small telephone exchange (called SX4). It is intended to bring the customers and the developers of the system together to a commit on a functional design.

The document will use Unified Modeling Language (UML) to specify various components of the software and how these components interact. The intended audience to this document is developers, customers and maintainers of the software. It is assumed that the readers of this document are familiar with UML and standard conventions used with UML.

1.2 Scope

The document will specify the functionality of the OAM software system. A brief interface description of the hardware communication protocol is provided. For further information the hardware interface description document should be consulted. The OAM software is responsible for maintaining customer account and billing information as well as storing call processing information. It will also include a description of a user interface the operator will use to perform operations and administrative tasks.

1.3 Definitions, Notational Conventions

To simplify the document the following acronyms are used:

Acronym/Name	**Definition**
CU	Control Unit
OAM	Operations, Administration and Maintenance
O&A	Operations and Administration
UML	Unified Modeling Language
COS	Class of Service
GUI	Graphical User Interface
DN	Dialed Number
EX	Telephone Exchange
Shelf	A particular shelf within an exchange
Slot	Location (slot) within a shelf (within an exchange)
STAT	Bit field describing the status of of a hardware device
SX4	The telephone exchange that the OAM works with
LD	Long Distance

Table 1: Table of Acroynms

The following naming conventions were used for return values:

Acronym	Definition
OK	Function returned without an error
OK(msg, arg)	Function returned without an error. (msg, arg) should be displayed to operator
NOK	Function returned with an error
NOK(error, arg)	Function returned with an error. (error, arg) should be displayed to operator

Table 2: Naming Conventions

The following UML notation conventions have been used that deviate from UML Toolkit:
0..n represents a finite number of multiplicity
0..* represents a infinite number of multiplicity
$ in front of a operation represents a static method
The “creates” keyword is used to indicate creation of an object

In the specification of the and the system calls, associated with certain GUI widgets; if a function takes parameters, and those parameters are found in another widget on the same screen, a {} with the index number between the curly braces is used to indicate where the parameter is filled in from. If a widget has a context dependent function (like a toggle), then the first function will be labeled 1a, 2a, etc, and the next function will be labeled 1b, 2b, etc, and so forth.

1.4 References

The following list is a list of documents specifically reference by this document. Inline reference numbers will map to the list of documents here.

1. Software Requirements and Specification Overview of the Course Project
▷ http://www.student.math.uwaterloo.ca/~cs445/project/intro.ps
2. Software Requirements and Specification Software Interface Description
▷ http://www.student.math.uwaterloo.ca/~cs445/project/soft.ps
3. Software Requirements and Specification Hardware Interface Description
▷ http://www.student.math.uwaterloo.ca/~cs445/project/hwif.ps
4. UML Toolkit

1.5 Overview

There are three main sections to the document. Section two specifies the general description of the OAM software. It specifies assumptions, product functions, constraints, and general interaction with the environment. Section three discusses specific system requirements, including external interface, behavioral, performance, accuracy, and non-behavioral requirements. Section four is the Data Dictionary which specifies all classes, attributes, operations, events and activities in the system.

2 General Description

The OAM system maintains and manages, a number of phone exchanges. The two primary functions of this system are, to maintain mappings between phone numbers and line cards, and to provide an interface for operators to perform OAM tasks.

2.1 Product Perspective

The OAM system is accessed by several Control Units (CU). Each CU communicates with the OAM through message passing. These messages indicate when synchronization events occur (e.g. updateDB, init). The OAM maintains a master copy of the exchange database. This database stores mappings between phone numbers, and line cards. Cached copies of this database are maintained on each CU.

There is one Operator console that users can interface with, to perform operations and administrative tasks. These operations and administrative tasks include, the storage of call processing information and client information. The OAM system can be seen as a collaboration of three entities, O&A, billing and maintenance.

A general overview of the interaction between the Operator Console, the OAM Computer and the individual Control Units is shown below.

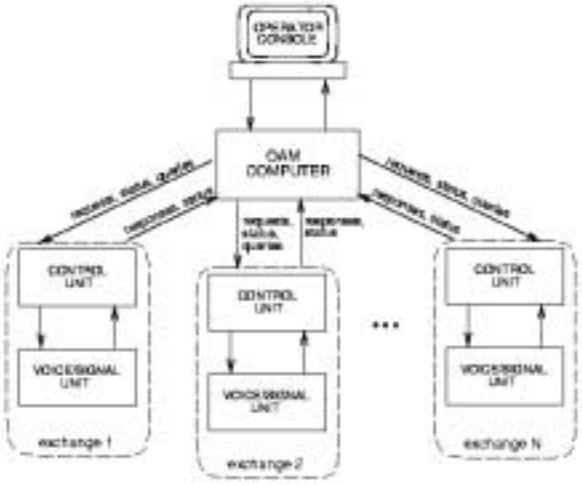

Figure 1: Overview of OAM/CU interface

2.2 Product Functions

The Operator can create, modify, view and remove customer accounts, as well as update and query billing information and issue hardware tests.

The modification of customer accounts includes the addition, modification and removal of subscriptions. These subscriptions associate a phone number to a line card, which is in turn, leased by a customer. The addition of subscriptions requires the OAM to inform each CU of the changes to the exchange database (ie: the new mapping between phone number and line card).

The CU sends messages to the OAM system when it starts up, and when it makes changes to its copy of the exchange database. The system tracks calls a customer makes by interfacing with the CU. The interface with the CU controls the hardware cards and bills calls when completed.

2.3 Characteristics of eventual users

The Operator will be the only human user of the OAM system. The Operator will be required to understand the SX4 exchange system, be capable of operating the GUI. In order to use the system efficiently, the Operator may require training.

2.4 Constraints

The following are constraints of the OAM software.

- Only one Operator will interface with the OAM software at a time.
- There is a maximum of nine exchanges.
- The operator must be able to easily interface with the system.

2.5 Assumptions & Dependencies

The following assumptions were made about the system:

- The machine on which the OAM software runs, will be capable of handling the requests of all CUs, and the console, in an efficient and timely manner [49];
- Phone numbers are unique [50];
- Each phone has at most one phone number mapped to it [51];
- A unique identification number (CustID) will be provided for each customer [52];
- A suspended subscription will not maintain an association with a line card [53];
- A suspended subscription will maintain an association with a phone number [54];
- The GUI will be dialog modal based [55];
- System integrity will be maintained, regardless of environmental interference (eg: hardware failure will not corrupt system) [56];
- A user of the system is an authorized OAM operator with valid login and password [14];
- There is only one operator using the system at any given time [57];
- Local calls are billed at a fixed rate per month [58];
- Each phone line has a unique bill [59];

- Cancelling a subscription will cause billing of monthly service to be prorated for the month (monthly rate $\times$ # of days/31) [16];

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interface

SX4 OAM Overview

Presented below, is a prototype for the OAM system GUI. From any screen, it is implicit that the Operator may go back to the previous window; this is done using the “Back” button. Thus for screens which may be reached from multiple points (“Add New Subscription” for example), the function of the “Back” button is dependent on which screen was previous to the current one.

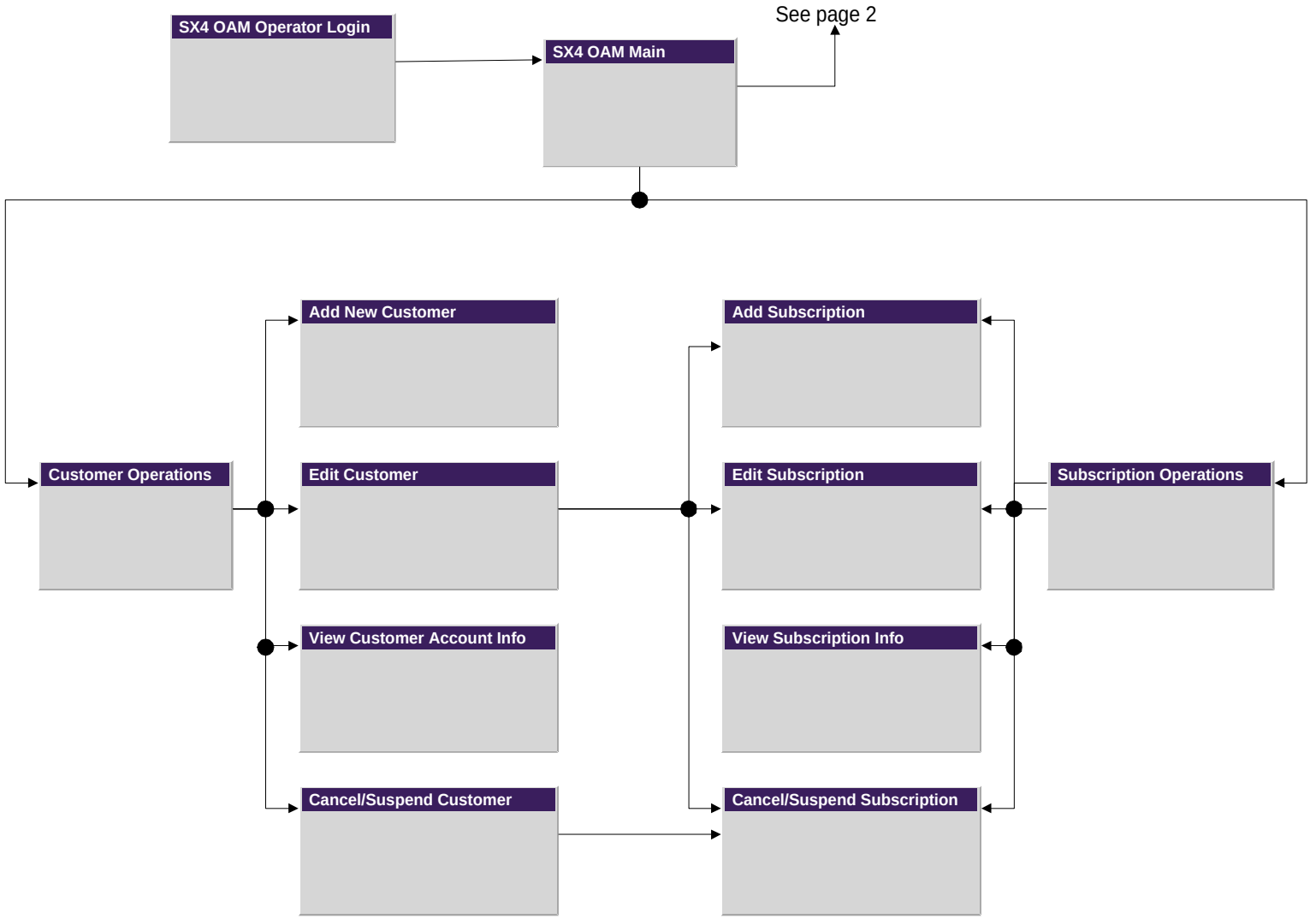

Figure 2: SX4 OAM Overview Diagram (part 1 of 2)

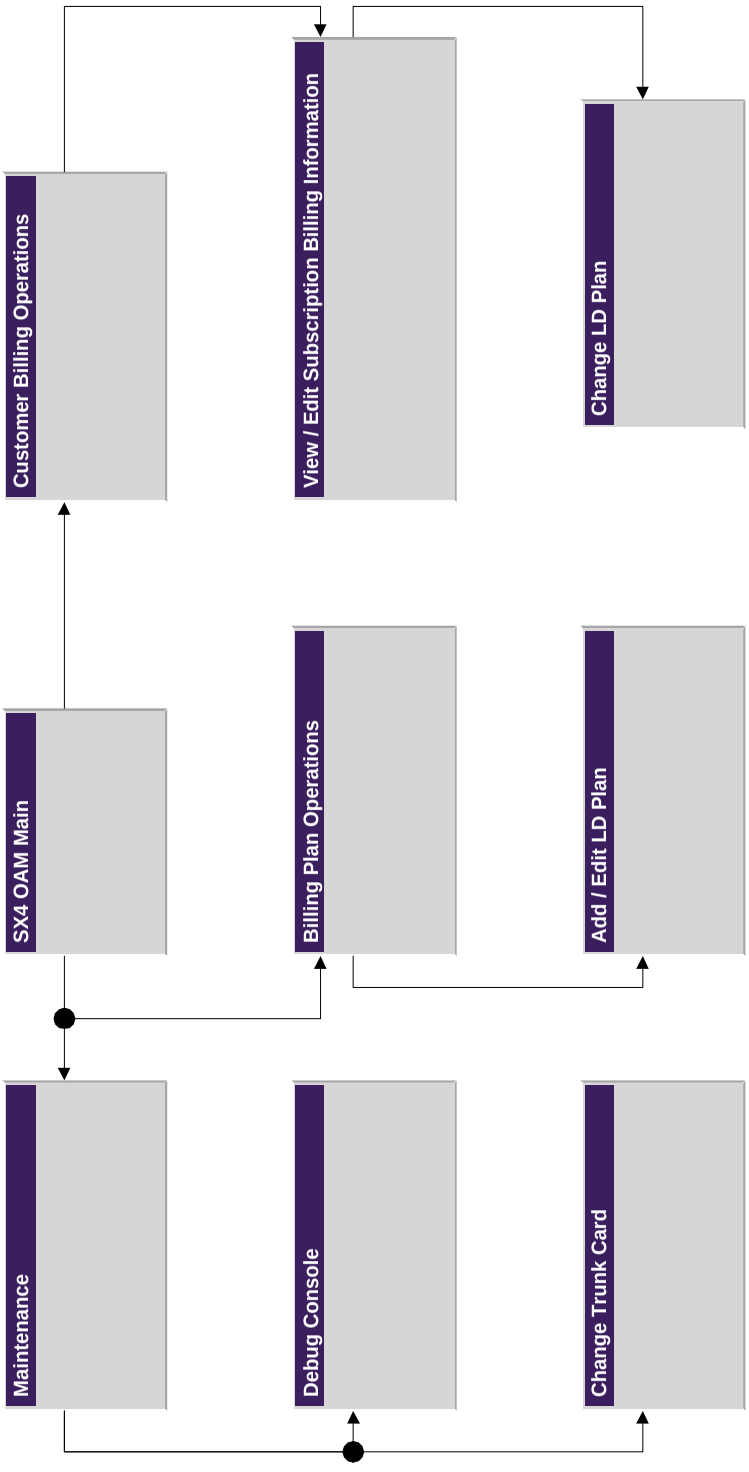

Figure 3: SX4 OAM Overview Diagram (part 2 of 2)

SX4 OAM Operator Login

Initially the OAM awaits the authentication of an Operator. Currently there is a single username and password required to enter the system. Once authenticated, the Operator has full access to the OAM system.[14]

Index	Type	Purpose	Effect
1	Text fields	Username and password data is entered here	None
2	Button	To validate login and (if successful) grant Operator access to OAM	When pressed, the username and password are validated, and if valid, the "SX4 OAM Main" dialogue is opened

Table 3: SX4 OAM Operator Login Window Description

Index	Sequence of associated system calls
2	1) OAMConsole::login({1}name, {1}pass)

Table 4: SX4 OAM Operator Login Window Widget Mappings0.5cm

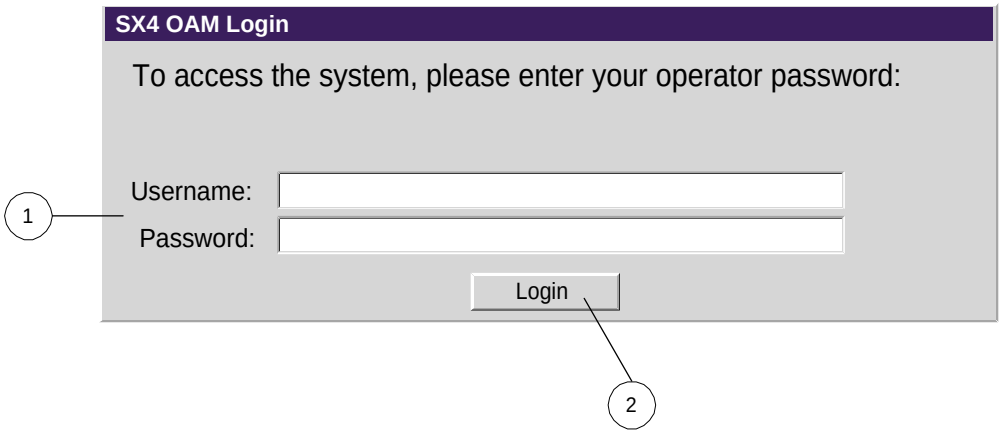

Figure 4: SX4 OAM Operator Login

This is the main menu to the OAM. From here, the Operator may focus on either customer operations, or subscription operations. The Operator may perform maintenance actions, or billing actions. The Operator may also log off the system from this menu.

Index	Type	Purpose	Effect
1	Button	Begin performing customer operations	Open “Customer Operations” dialogue
2	Button	Begin performing subscription operations	Open “Subscription Options” dialogue
3	Button	Log Operator out of OAM system	Log out Operator and return to “SX4 OAM Operator Login” dialogue
4	Button	Begin performing maintenance operations	Open “Maintenance” dialogue
5	Button	Begin performing billing plan operations	Open “Billing Plan Operations” dialogue
6	Button	Begin performing customer billing operations	Open “Customer Billing Operations” dialogue

Table 5: SX4 OAM Main Window Description

Index	Sequence of associated system calls
3	1) OAMConsole::logout()

Table 6: SX4 OAM Main Window Widget Mappings

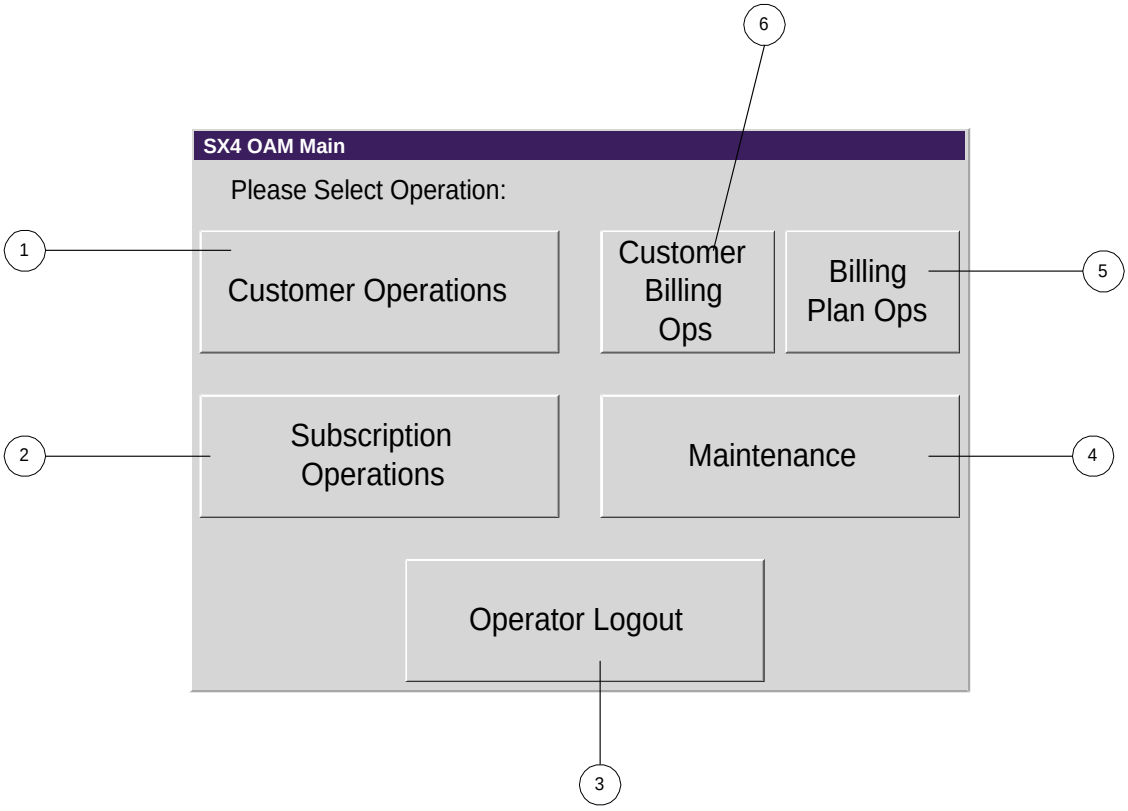

Figure 5: SX4 OAM Main

From this screen, the Operator may view, modify, or create a customer account.

Index	Type	Purpose	Effect
1	Button	Begin adding a new customer account	Open “Add New Customer Account” dialogue
2	Button	Begin editing a customer account	Open “Edit Customer Account” dialogue
3	Button	Return to Main dialogue	Close dialogue, return to “SX4 OAM Main” dialogue
4	Button	Begin canceling or suspending a customer account	Open “Cancel or Suspend Customer Account”
5	Button	Begin viewing a customer account	Open “View Customer Account” dialogue

Table 7: Customer Operations Window Description

Index	Sequence of associated system calls
	None

Table 8: Customer Operations Window Widget Mappings

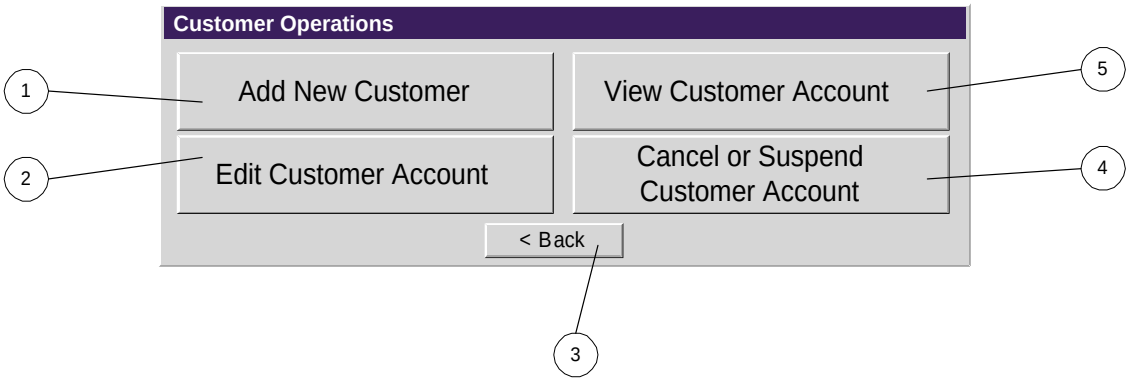

Figure 6: Customer Operations

bold Add New Customer Account

This form allows the Operator to create a new customer account. [1] Initially the form is blank. Once the Operator fills in the customer information, the “Add” button is pressed. This attempts to add the new account to the system. If the account can not be added (for example, because another account with the same information already exists), the OAM will notify the Operator. This notification is in the form of a message in the “Account Status” area of the form. If the new account was successfully created, a notification message is written to the “Account Status” area of the form.

Index	Type	Purpose	Effect
1	Text Fields	Allows Operator to enter in customer information	None
2	Text	When an add is attempted, this area is updated, to tell the Operator if the add was successful or, if not, why it failed.	None
3	Button	Close dialogue, and return to “Customer Operations” dialogue	Dialogue is closed, control returns to “Customer Operations” dialogue
4	Button	Attempt to add a new customer	System attempts to add a new customer using the data from (1). If an existing record containing the customer information already exists, then the add fails. Otherwise, the add succeeds.

Table 9: Add New Customer Account Window Description

Index	Sequence of associated system calls
4	1) Customer::create({1}info)

Table 10: Add New Customer Account Window Widget Mappings

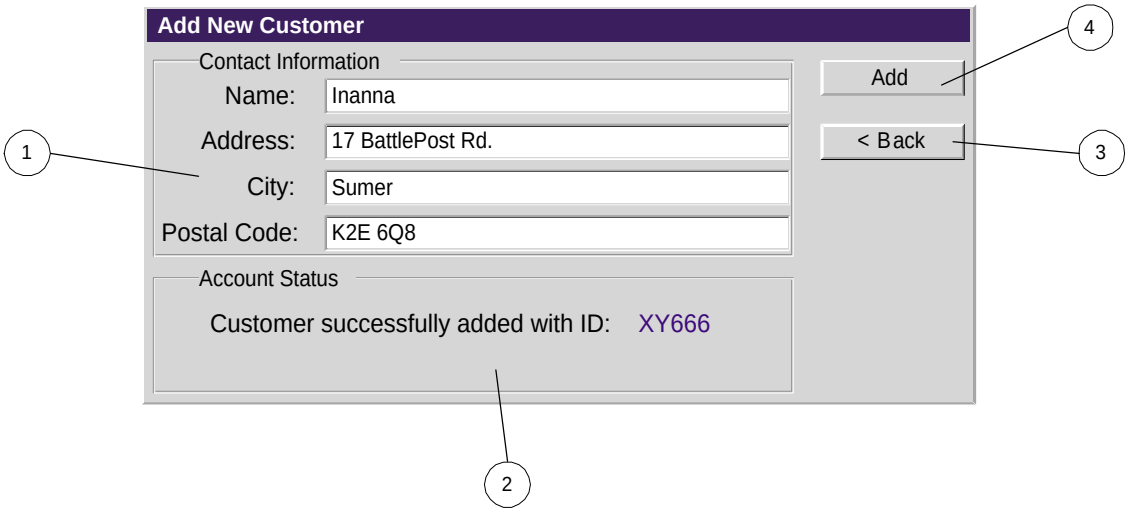

Figure 7: Add New Customer Account

To edit a customer account, the Operator uses this form. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the “Find” button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account’s subscriptions. [36] On a successful find, the “Account Status” area indicates that the account was successfully retrieved. If the “Find” fails, a message is written to the “Account Status” area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Once an account has been found, the Operator may make modifications to it. The Operator may freely modify any of the “Contact Information” fields. In addition, the Operator may perform modifications to the account’s list of subscriptions. Once the desired modifications are made, the Operator must press the “Update” button to commit the changes. [2] If the “Update” was successful, a success message will be written to the “Account Status” area. On failure, a message detailing the failure and possible suggestions to remedy the problem, will be placed into the “Account Status” area.

When the Operator wishes to leave, by pressing the “Back” button, a check is made to see if changes have occurred since the last “Update”. If so, a message indicating this is written to the “Account Status” area, and the Operator must either “Update” the changes, or press “Back” a second time.

Index	Type	Purpose	Effect

1	Text Fields	Allows operator to enter in customer information	None
2	List Boxes	Subscriptions associated with customer account, are displayed here	Only a single subscription may be selected at a time.
3	Text	Whenever the Operator attempts to modify the customer information, or subscription information (add, edit, cancel, or suspend), the result of the operation is displayed here. The results of a search for a customer record are also displayed here.	None
4	Button	Begin adding a new subscription	Open “Add Subscription” dialogue
5	Button	Allow Operator to cancel a subscription	Cancel subscription, by dissociating, and freeing phone number, and dissociating and freeing line card (if associated). Final bill is sent to customer. [10, 16]
6	Button	Allow Operator to suspend a subscription, or reactivate a suspended subscription	Suspend subscription, by dissociating, and freeing line card [17]. If subscription is already suspended, this will reactivate it.
7	Button	Begin editing a subscription	Open “Edit Subscription” (sub dialogue), passing to it the selected subscription’s information
8	Button	Close dialogue, and return to “Customer Operations” dialogue	Dialogue is closed, and control returned to “Customer Operations” dialogue.

9	Button	Update the customer account with the modifications made	If a change is made, update the customer account to reflect the changes. This will fail if the ID is changed to one that already exists.
10	Button	From the given information, find the associated customer account	From the partial information the Operator enters into (1), this will find the customer account which best matches. If a unique account is found, the rest of the fields in (1) and (2) are updated. Otherwise a message is displayed in (3) indicating the number of matching accounts. [36]

Table 12: Edit Customer Account Window Description

Index	**Sequence of associated system calls**
5	1) Subscription::cancel() 2) PhoneNumber::release() 3) LineCard::release()
6	1a) Subscription::suspend() 2a) LineCard::release() 1b) Subscription::activate 2b) LineCard::findFree({2}DN) 3b) LineCard::initialize({2}DN)
9	1) Customer::setCInfo({1}cInfo)
10	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)

Table 13: Edit Customer Account Window Widget Mappings

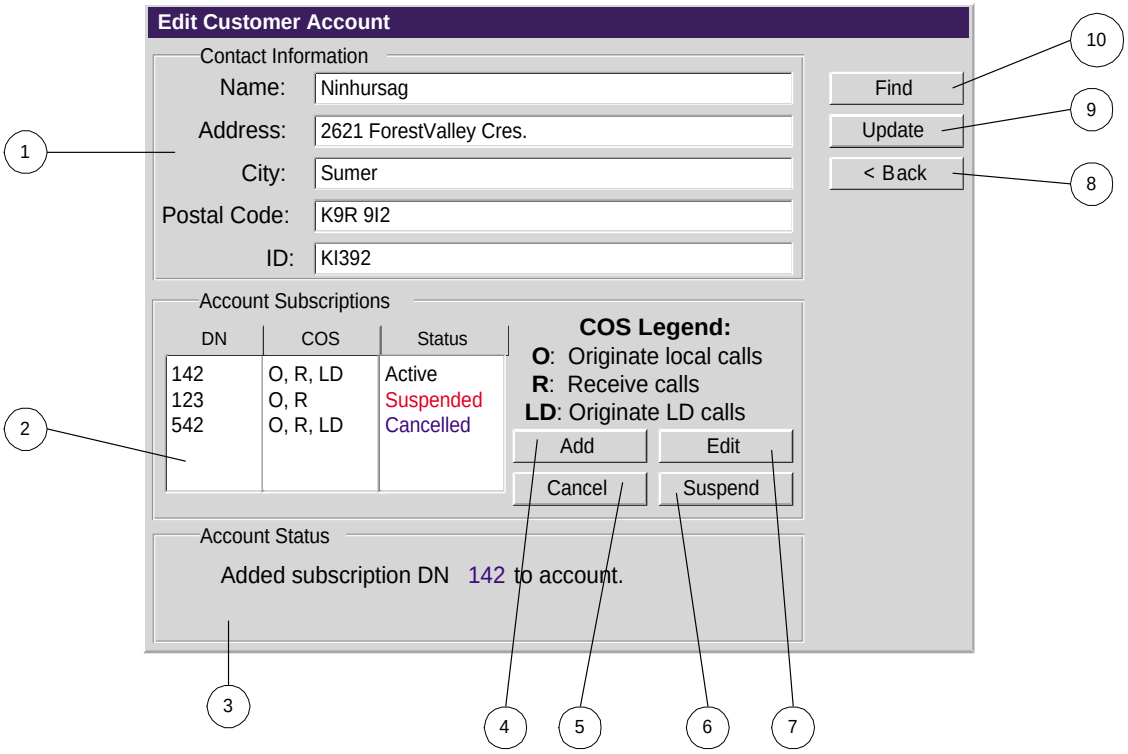

Figure 8: Edit Customer Account

If the Operator wishes to simply view account details, this form may be used. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the “Find” button. [4]If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account’s subscriptions. On a successful find, the “Account Status” area indicates that the account was successfully retrieved. If the “Find” fails, a message is written to the “Account Status” area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Once a match is found, the status of the account (whether it is active, suspended, or canceled), is indicated in the “Account Status” area.

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in customer information	None
2	List Boxes	Subscriptions associated with customer account, are displayed here	Only a single subscription may be selected at
3	Text	Whenever the Operator attempts to find a customer account, the results of the search are displayed here	None
4	Button	Close dialogue, and return to	Dialogue is closed, and control returned to
5	Button	From the given information, find	From the partial information the Operator enters into (1), this will find the customer account which best matches. If a unique account is found, the rest of the fields in (1) and (2) are updated. Otherwise a message is displayed in (3) indicating the number of matching accounts. [36]

Table 14: View Customer Account Window Description

Index	Sequence of associated system calls
5	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)

Table 15: View Customer Account Window Widget Mappings

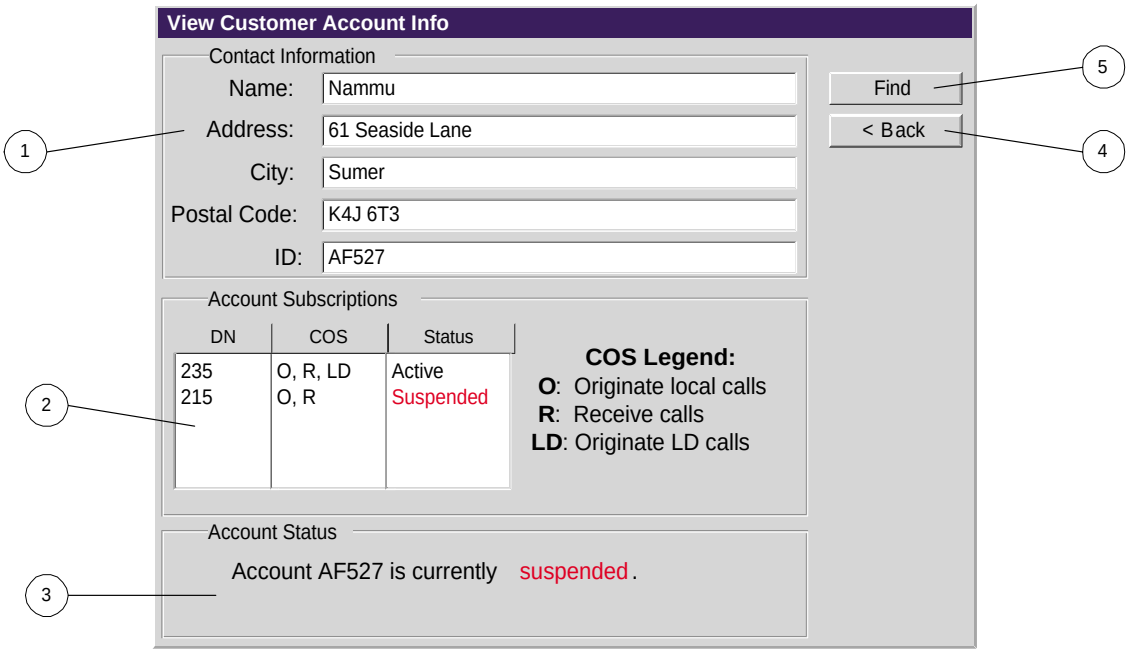

Figure 9: View Customer Account

To cancel or suspend a customer account, the screen is used. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the “Find” button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account’s subscriptions. On a successful find, the “Account Status” area indicates that the account was successfully retrieved. If the “Find” fails, a message is written to the “Account Status” area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Once a match is found, the Operator may suspend the account by pressing on the “Suspend” button, or cancel the account by pressing on the “Cancel” button. On cancel or suspend of an account, all the account’s subscriptions are canceled, or suspended, respectively. [3, 35] The results of the suspend or cancel operations are placed in the “Account Status” area.

Alternatively, the Operator may suspend or cancel individual subscriptions from this screen by using the “Cancel” or “Suspend” buttons located in the “Account Subscriptions” area.

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in customer information	None
2	List Boxes	Subscriptions associated with customer account, are displayed here	Only a single subscription may be selected at

3	Text	Whenever the Operator attempts to find a customer account, the results of the search are displayed here. The results of an attempt to cancel or suspend a customer account, or subscription, are displayed here.	None
4	Button	Allow Operator to cancel a	Cancel subscription, by dissociating, and freeing
5	Button	Allow Operator to suspend a subscription, or reactivate a suspended subscription	Suspend subscription, by dissociating, and freeing line card [17]. If subscription is already suspended, this will reactivate it.
6	Button	Close dialogue, and return to	Dialogue is closed, and control returned to “Customer Operations” dialogue.
7	Button	Allow Operator to cancel the customer account	Cancel all subscriptions associated with account, then set account to canceled
8	Button	Allow Operator to suspend the	Suspend all subscriptions associated with account. then set account to suspended
9	Button	From the given information, find	From the partial information the Operator enters into (1), this will find the customer account which best matches. If a unique account is found, the rest of the fields in (1) and (2) are updated. Otherwise a message is displayed in (3) indicating the number of matching accounts. [36]

Table 17: Cancel or Suspend Customer Account Window Description

Index	Sequence of associated system calls
4	1) Subscription::cancel() 2) PhoneNumber::release() 3) LineCard::release()
5	1a) Subscription::suspend() 2a) LineCard::release() 1b) Subscription::activate 2b) LineCard::findFree({2}DN) 3b) LineCard::initialize({2}DN)
7	1) Customer::cancel() 2) For each subscription, perform steps associated with Index 4 (cancel all subscriptions)
8	1) For each subscription, perform steps associated with Index 5 (suspend all subscriptions)
9	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)

Table 18: Cancel or Suspend Customer Account Window Widget Mappings

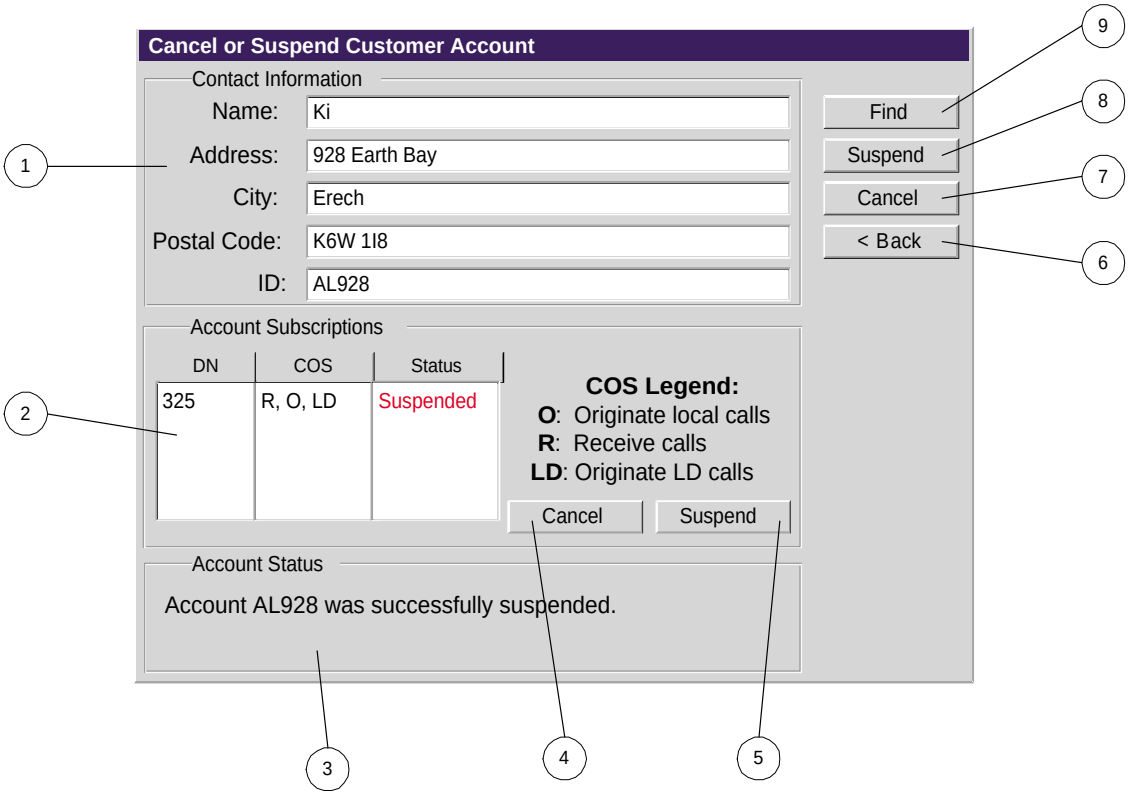

Figure 10: Cancel or Suspend Customer Account

From the screen, the Operator may add, view, edit, cancel, or suspend subscriptions.

Index	Type	Purpose	Effect
1	Button	Begin adding new subscriptions	Open “Add Subscription” dialogue
2	Button	Begin editing subscriptions	Open “Edit Subscription” dialogue
3	Button	Return to Main dialogue	Close dialogue, return to “SX4 OAM Main”
4	Button	Begin canceling or suspending subscriptions	Open “Cancel or Suspend Subscription” dialogue
5	Button	Begin viewing subscriptions	Open “View Subscription” dialogue

Table 19: Subscription Operations Window Description

Index	Sequence of associated system calls
	None

Table 20: Subscription Operations Window Widget Mappings

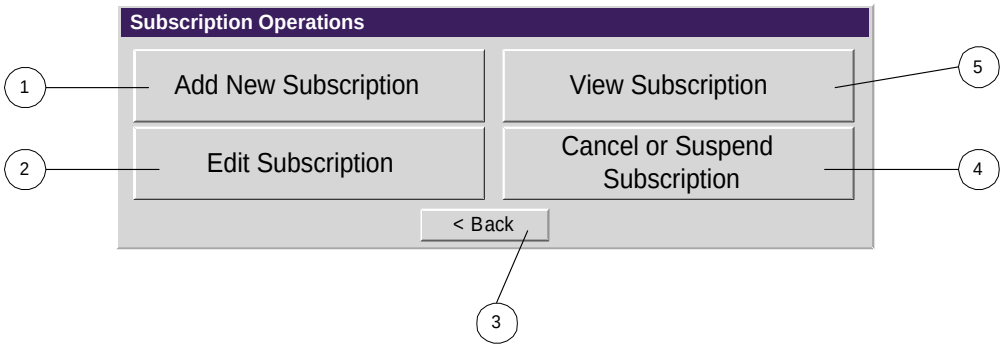

Figure 11: Subscription Options

This form allows the Operator too add subscriptions to a customer’s account. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the “Find” button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account’s subscriptions. On a successful find, the “Account Status” area indicates that the account was successfully retrieved. If the “Find” fails, a message is written to the “Account Status” area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Once a match is made, a new subscription may be added by pressing on the “Add New” button in the “Account Subscriptions” area. This brings up a smaller dialogue box, which allows the class of service and account status to be selected. By default, the class of service is fully permissive, and the account status is “Active”. [48] This dialogue allows the Operator to request from the OAM either a specific number (by directly entering it in, into the DN field), or any available number (by entering an “*” into the DN field). To add the subscription, the Operator clicks on the “Ok” button. [21] To abort the operation, the “Cancel” button is pressed. On success or failure, an

information message is written to the “Account Status” area of the “Add Subscription” screen.

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in customer information	None
2	List Boxes	Subscriptions associated with customer account, are displayed here	Only a single subscription may be selected at
3	Text	Whenever the Operator attempts to find a customer account, the results of the search are displayed here. The results of an attempt to add a new subscription are displayed here.	None
4	Button	Begin adding a new subscription to the customer account.	Open up “Add Subscription” sub-dialogue (shown in figure, as the second dialogue). If successful, the new subscription is added to (2).

Index	Type	Purpose	Effect
5	Button	Close dialogue, and return to "Customer Operations" dialogue or "Subscription Operations" dialogue (depending on which opened this dialogue)	Dialogue is closed, and control returned to
6	Button	From the given information, find	From the partial information the Operator enters into (1), this will find the customer account which best matches. If a unique account is found, the rest of the fields in (1) and (2) are updated. Otherwise a message is displayed in (3) indicating the number of matching accounts. [36]
7	Text Field	The DN to be used for the new subscription.	This can be manually entered by the Operator, to suggest specific DNs, or in conjunction with (12), a random DN can be created.
8	Check Boxes	Set the COS for the new subscription	None

Index	Type	Purpose	Effect
9	Button	Add new subscription with the attributes given	Add new subscription with the attributes given [21, 48]
10	Button	Abort the addition of a new subscription	Close "Add Subscription" sub-dialogue, and return control to main "Add Subscription" dialogue
11	Radio Buttons	Set the activation status of the new subscription	None

12	Button	Generate a random, unused DN	Given an exchange number in (7), an unused DN (if available) will be generated, and be placed in (7)

Table 23: Add Subscription Window Description

Index	**Sequence of associated system calls**
6	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)
9	1) Subscription::initialize({7}phoneNum) 2) PhoneNumber::allocate() 3) Subscription::setCOS({8}cos) 4-a) If {11} is set to Cancel, then cancel subscription 4-b) If {11} is set to Suspend, then suspend subscription 4-c) If {11} is set to Active, then activate subscription
12	1) PhoneNumber::findFree({7}ex)

Table 24: Add Subscription Window Widget Mappings

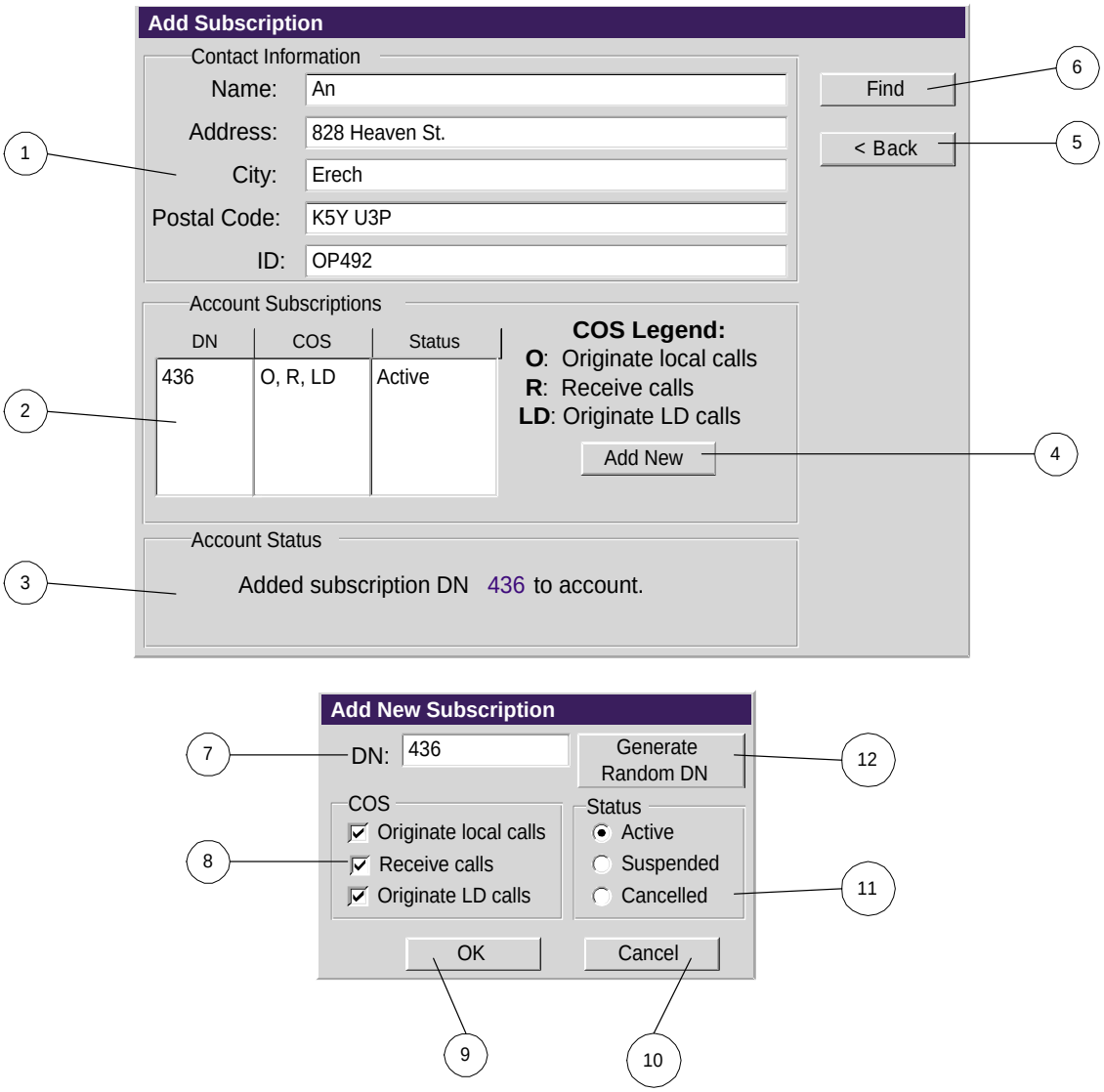

Figure 12: Add Subscription

If the Operator wishes to simply view an account's subscriptions, this screen is used. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the "Find" button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account's subscriptions. [23, 36] On a successful find, the "Account Status" area indicates that the account was successfully retrieved. If the "Find" fails, a message is written to the "Account Status" area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in customer information	None
2	List Boxes	Subscriptions associated with customer account, are displayed here	Only a single subscription may be selected at
3	Button	Close dialogue, and return to "Customer Operations" dialogue or "Subscription Operations" dialogue (depending on which opened this dialogue)	Dialogue is closed, and control returned to
4	Button	From the given information, find	From the partial information the Operator enters into (1), this will find the customer account which best matches. If a unique account is found, the rest of the fields in (1) and (2) are updated. Otherwise a message is displayed in (3) indicating the number of matching accounts. [36]

Table 25: View Subscription Window Description

Index	Sequence of associated system calls
4	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)

Table 26: View Subscription Window Widget Mappings

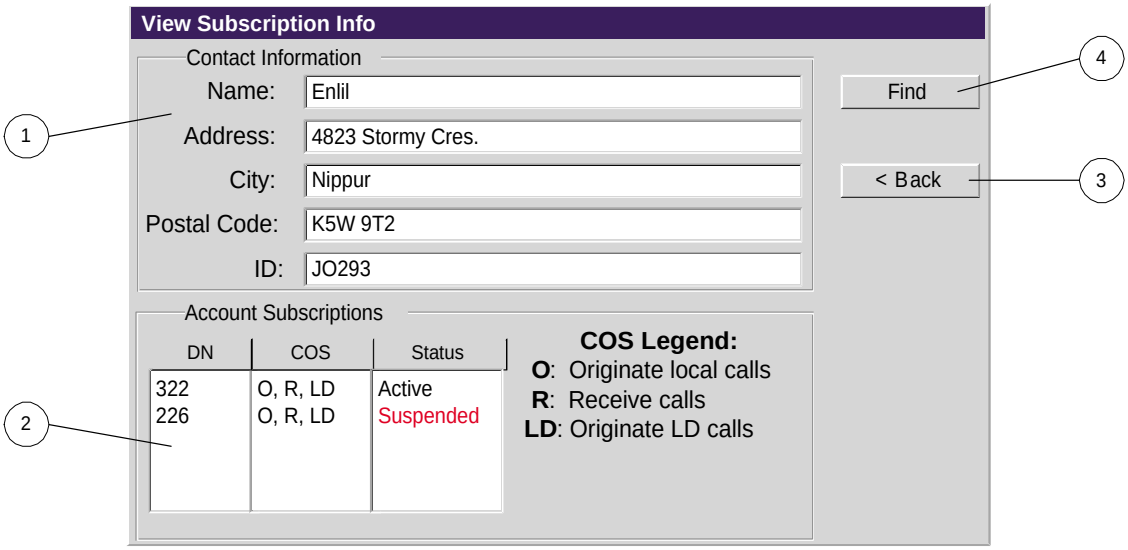

Figure 13: View Subscription

To edit a subscription, this form is used. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the "Find" button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account's subscriptions. On a successful find, the "Account Status" area indicates that the account was successfully retrieved. If the "Find" fails, a message is written to the "Account Status" area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Once a match is found, the Operator selects the subscription to edit, then clicks on the "Edit" button. This brings up a new dialogue box which allows modification of the subscription. This new dialogue box allows the Operator to modify the subscriptions class of service, status, or DN. To confirm the modifications, the "Ok" button is pressed. [22] Otherwise the "Cancel" button is used to close the dialogue box.

Upon exit, a message is written to the "Account Status" area, indicating the operation performed, and whether or not it was successful (and other relevant details).

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in customer information	None
2	List Boxes	Subscriptions associated with customer account, are displayed here	Only a single subscription may be selected at

3	Text	Whenever the Operator attempts to find a customer account, the results of the search are displayed here. The results of an attempt to edit a subscription are displayed here.	None
4	Button	Begin editing a new subscription	Open up “Edit Subscription” sub-dialogue, passing to it the information of the currently selected subscription (if one is selected). If a subscription isn’t selected, the sub-dialogue is not opened, and an error message is displayed in (3)
5	Button	Close dialogue, and return to “Subscription Operations” dialogue	Dialogue is closed, and control returned to
6	Button	From the given information, find	From the partial information the Operator enters
7	Text Field	The DN to be used for the new	This can be manually entered by the Operator, to
8	Check Boxes	Set the COS for the new	None
9	Button	Apply changes to the subscription, using the settings in the sub-dialogue	Apply changes to the subscription, using the settings in the sub-dialogue [22]
10	Button	Abort the editing of the	Close “Edit Subscription” sub-dialogue, and return control to main “Edit Subscription” dialogue
11	Radio Buttons	Set the activation status of the	None
12	Button	Generate a random, unused DN	Given an exchange number in (7), an unused DN

Table 28: Edit Subscription Window Description

Index	Sequence of associated system calls
6	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)
9	1) (If DN changed) PhoneNumber::release() 2) (If DN changed) Subscription::initialize({7}phoneNum) 3) (If DN changed) PhoneNumber::allocate() 4) Subscription::setCOS({8}cos) 5-a) If {11} is set to Cancel, then cancel subscription 5-b) If {11} is set to Suspend, then suspend subscription 5-c) If {11} is set to Active, then activate subscription
12	1) PhoneNumber::findFree({7}ex)

Table 29: Edit Subscription Window Widget Mappings

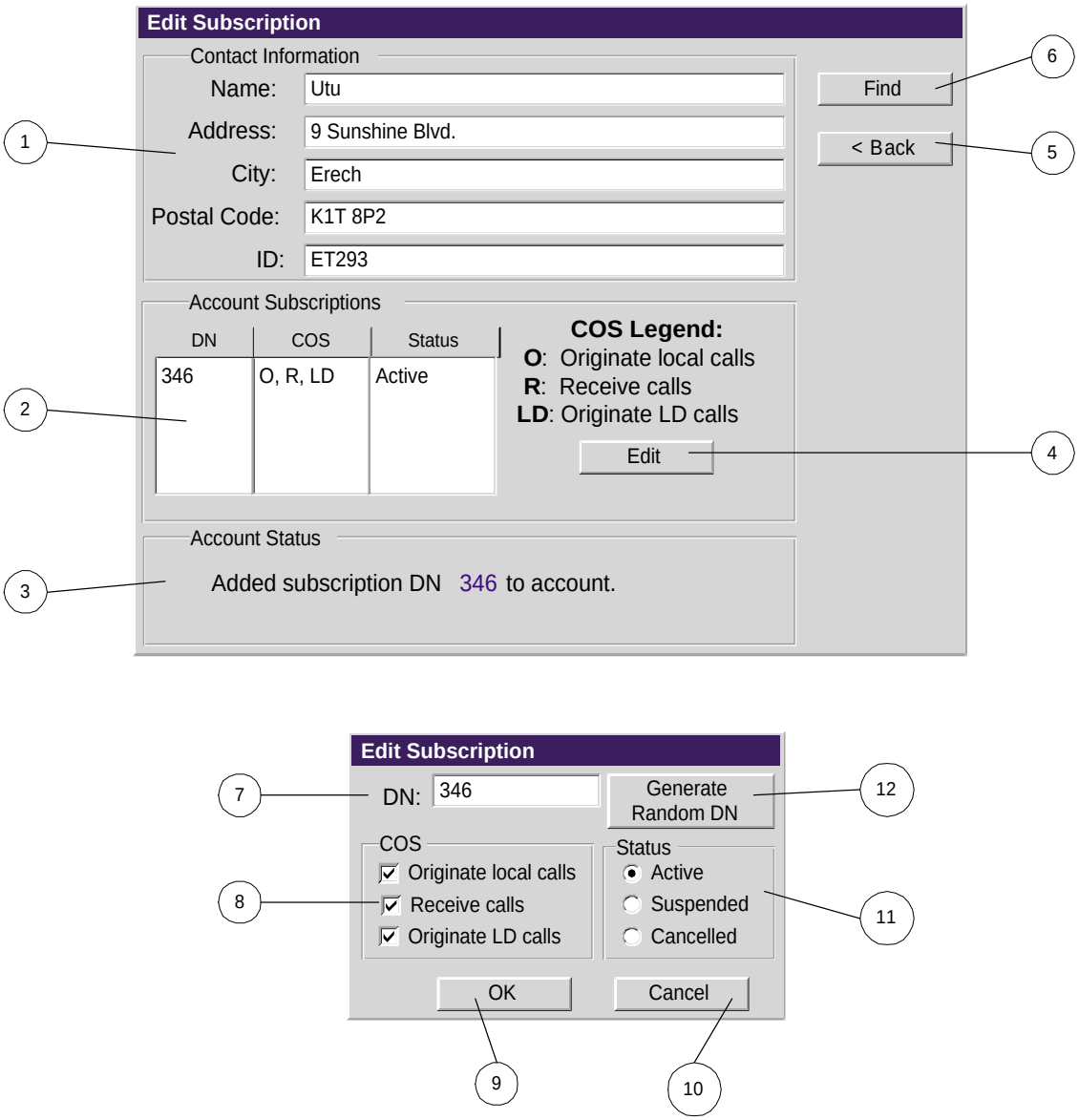

Figure 14: Edit Subscription

...

To cancel or suspend/activate a subscription, this form is used. Initially the form is blank. The Operator then enters data into some of the fields, and presses on the "Find" button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account's subscriptions. On a successful find, the "Account Status" area indicates that the account was successfully retrieved. If the "Find" fails, a message is written to the "Account Status" area indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

When a match is found, the Operator may select a subscription and cancel or suspend/activate it, by using the "Cancel" or "Suspend/Activate" buttons, respectively. On success, or failure, a message is written to the "Account Status" area indicating the operation performed and the operation's success. The "Suspend/Activate" button is a toggle and will only display either "Suspend"

or “Activate” depending on the status of the subscription. [10, 17, 18]

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in	None
2	List Boxes	Subscriptions associated with	Only a single subscription may be selected at
3	Text	Whenever the Operator attempts to find a customer account, the results of the search are displayed here. The results of an attempt to cancel or suspend a subscription are displayed here.	None
4	Button	Allow Operator to cancel a	Cancel subscription, by dissociating, and freeing
5	Button	Allow Operator to suspend a subscription, or reactivate a	Suspend subscription, by dissociating, and freeing line card [17]. If subscription is already suspended, this will reactivate it.
6	Button	Close dialogue, and return to	Dialogue is closed, and control returned to
7	Button	From the given information, find	From the partial information the Operator enters into (1), this will find the customer account which best matches. If a unique account is found, the rest of the fields in (1) and (2) are updated. Otherwise a message is displayed in (3) indicating the number of matching accounts. [36]

Table 31: Cancel or Suspend Subscription Window Description

Index	Sequence of associated system calls
4	1) Subscription::cancel() 2) PhoneNumber::release() 3) LineCard::release()
5	1a) Subscription::suspend() 2a) LineCard::release() 1b) Subscription::activate 2b) LineCard::findFree({2}DN) 3b) LineCard::initialize({2}DN)
7	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)

Table 32: Cancel or Suspend Window Widget Mappings

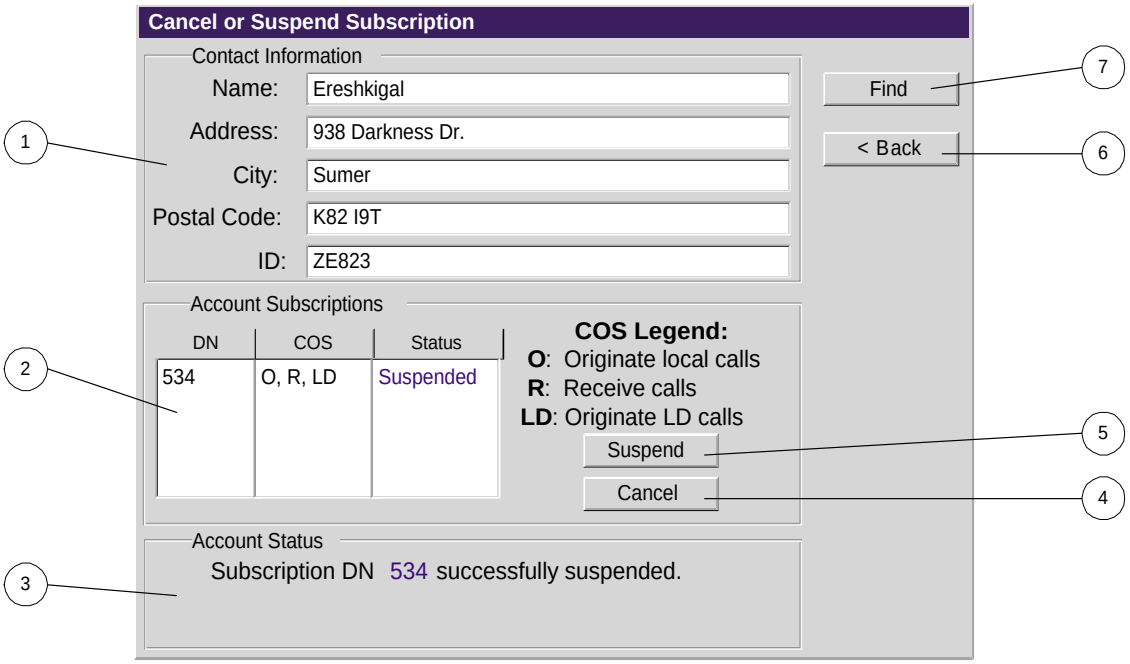

Figure 15: Cancel or Suspend Subscription

The Operator may perform various maintenance functions with the line cards, and CU system. When opened, this dialogue presents a list of all line and trunk cards. Upon selecting a card, the Operator may enable, or disable it. The Operator may request the card's controlling CU to be reset. Tests can be requested of individual cards, or on all cards. Trunk cards can have their associated exchange altered. As well, the Operator may initiate the debug console from here.

Index	Type	Purpose	Effect

1	List Boxes	All line and trunk cards are listed here, with their ENs, DNs, type, and activation status.	Only one card can be selected at a time
2	Button	Request CU to test the selected card	Request testing of the card with the EN from (1) [41]
3	Text	Whenever an action is taken upon a line card (test, disable, etc) this area is updated with a message indicating the action taken, and the result of that action	None
4	Button	Request CU to test all line cards	Request testing of all cards listed in (1) [49]
5	Button	Enable the selected line card	Enable the line card selected in (1) [39]
6	Button	Disable the selected line card	Disable the line card selected in (1) [40]
7	Button	Reset the CU controlling the selected line card	Reset the line card selected in (1) [42]
8	Button	Begin changing trunk card exchange association	If a trunk card is selected, then open “Change Trunk Card Association” sub-dialogue. Otherwise indicate in (3) that a trunk cards needs to be selected.
9	Button	Enter debug console	Open up “Debug Console” dialogue
10	Button	Close dialogue, and return to “SX4 OAM Main” dialogue	Close this dialogue, and return control to “SX4 OAM Main” dialogue
11	Text	Indicate which trunk card is being modified	None
12	Text field	Allow Operator to specify which exchange (DN) the trunk card is to be associated with	This must be a valid exchange number (ie: between 1 and 9, and not the same as the trunk card’s EX)

13	Button	Commit changes to trunk card	Change trunk card's DN to (12)
14	Button	Discard changes to trunk card	Don't update the trunk card. Close dialogue, and return control to "Maintenance" dialogue.

Table 34: Maintenance Window Description

Index	Sequence of associated system calls
2	1) Card::test()
4	1) For each card, do a Card::test()
5	1) Card::enable()
6	1) Card::disable()
7	1) Card::reset()
13	1) PhoneNumber::release() 2) Card::initialize({7}phoneNum) 3) PhoneNumber::allocate()

Table 35: Maintenance Window Widget Mappings

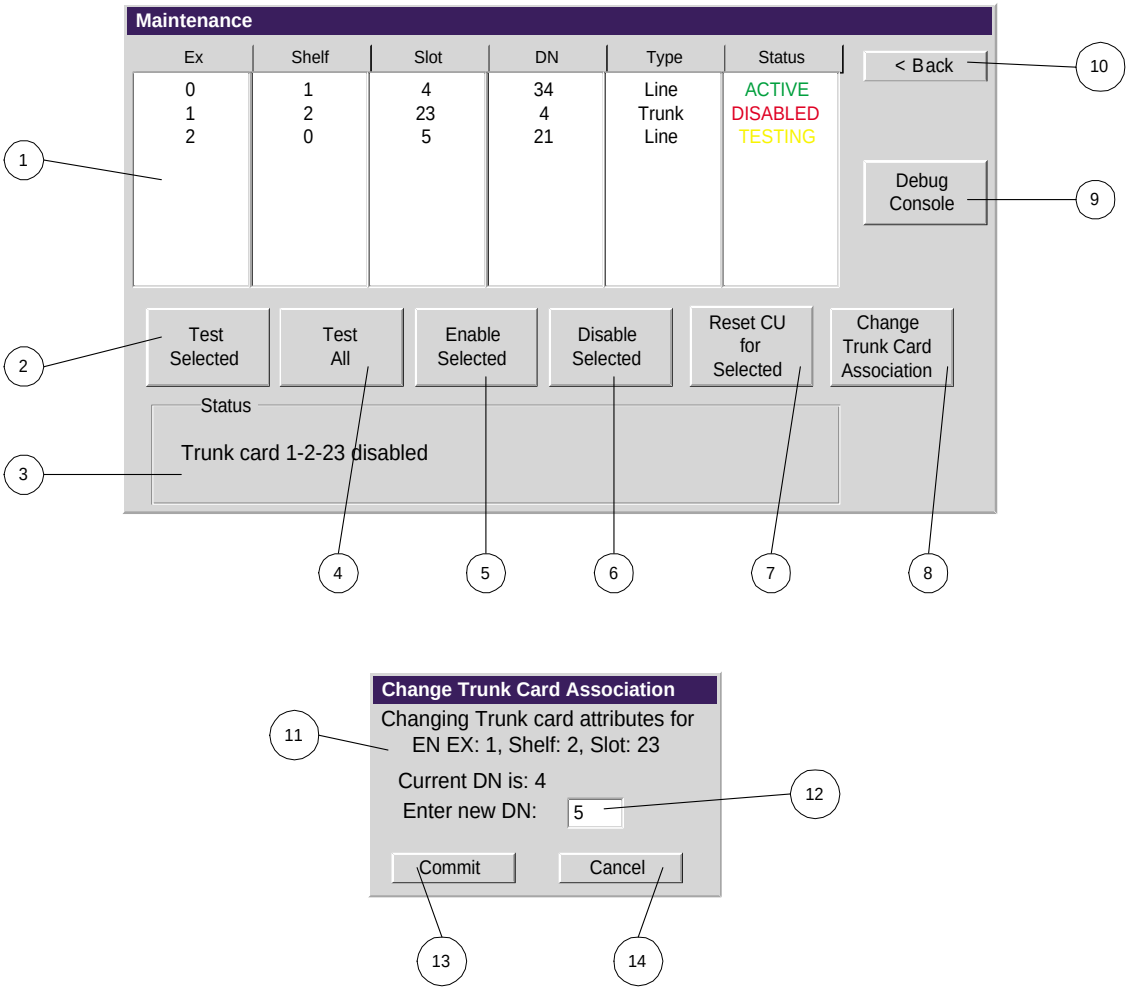

Figure 16: Maintenance

The Operator may request from the CU, information regarding a given card. Whenever a line card fails a self-test, the bottom message is displayed (over-top of whatever was showing at the time), indicating to the Operator which card has a problem. The operation of the console is synchronous, in that after a request is sent to the CU, the console waits for a response (up to some timeout).

Index	Type	Purpose	Effect
1	Text	Show a brief history of messages send to CU, and responses back from CU	None
2	Text Field	Allow Operator to enter in EX of card to be queried	None
3	Text Field	Allow Operator to enter in Shelf of card to be queried	None
4	Text Field	Allow Operator to enter in Slot of card to be queried	None
5	Button	Close dialogue and return to “Maintenance” dialogue	Close dialogue, and return control to “Maintenance” dialogue
6	Button	Send query to CU using data from (2), (3), and (4)	Send appropriate query-DB message to CU
7	Button	Close “Test Failure Warning” dialogue	Close “Test Failure Warning” dialogue

Table 36: Debug Console and CU Failure Dialogue Window Description

Index	Sequence of associated system calls
6	1) Card::queryDB({2}EX, {3}Shelf, {4}Slot)

Table 37: Debug Console and CU Failure Dialogue Window Widget Mappings

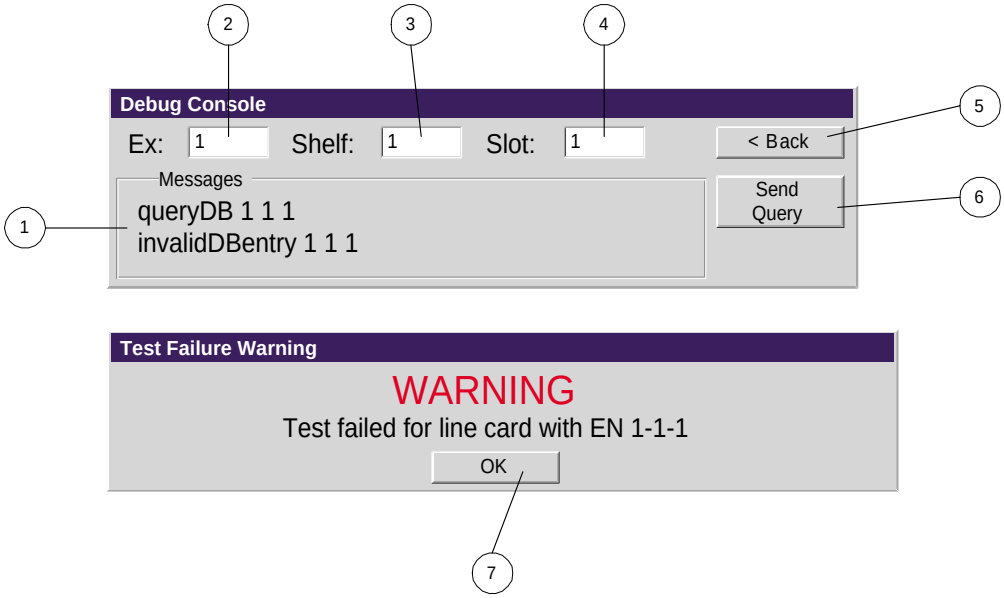

Figure 17: Debug Console and CU Failure Dialogue

To edit a subscription's billing information, this form is initially used. At first, the form is blank. The Operator then enters data into some of the fields, and presses on the "Find" button. If a unique customer account is found to match the entered data, the other fields are filled in with this account data, including the account's subscriptions. On a successful find, the "Contact Information" and "Account Subscriptions" areas are filled in, indicating that the account was successfully retrieved. If the "Find" fails, a message appears, indicating the reason for failure, and possible suggestions to increase the likelihood of a match.

Once a match is found, the Operator selects the subscription to view/edit, then clicks on the "View/Edit Selected Subscription Billing Info" button. This brings up a new dialogue box which allows modification of the subscription's billing information.

Index	Type	Purpose	Effect
1	Text Fields	Allows operator to enter in	None
2	List Boxes	Subscriptions associated with	Only a single subscription may be selected at
3	Button	Begin viewing / editing billing information pertaining to selected subscription	Open "View / Edit Subscription Billing Information" dialogue
4	Button	Close dialogue, and return to "SX4 OAM Main" dialogue	Close dialogue, and return control to "SX4 OAM Main" dialogue
5	Button	From the given information, find	From the partial information the Operator enters

Table 38: Customer Billing Operations Window Description

Index	Sequence of associated system calls
5	1a) Customer::find({1}cInfo) 1b) Customer::find({1}cID)

Table 39: Customer Billing Operations Window Widget Mappings

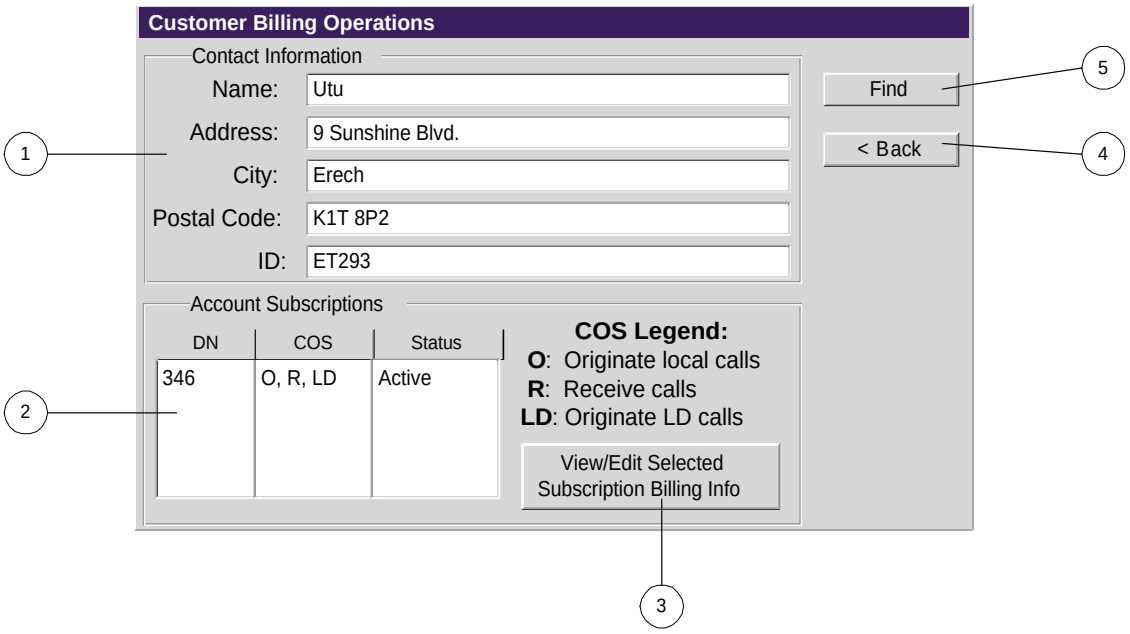

Figure 18: Customer Billing Operations

bold View / Edit Subscription Billing Info and Credit Subscription

When this dialogue appears, the current bill for the indicated (from the previous dialogue) subscription is displayed. The overall subscription balance is displayed, along with the number of local calls, and total charges for LD calls, for the selected billing period. By pressing the “Prev Bill” and “Next Bill” buttons, the Operator may scroll through the subscription’s bill history. [8, 9] If for any given bill, a reprint is desired, this may be requested. If the Operator wishes to add a credit to the customer’s current bill, this may be done by pressing the “Credit Subscription” button. [19,20] By pressing this button, another dialogue appears, allowing the Operator to add a credit to the account. The subscription’s current LD plan is displayed, and may be changed by the Operator.

Index	Type	Purpose	Effect
1	List Boxes	All LD calls for the selected billing period are listed here. The listings include the D-N called, the date, the start time, duration, rate, and charge.	None
2	Text	The number of local calls made in the selected billing period is listed here	None
3	Text	Display the current account balance for the subscription (regardless of what billing period is being looked at)	None
4	Text	The sum of the total LD charges for the selected billing period	None
5	Button	Begin changing the LD plan associated with the subscription	Open “Change LD Plan” dialogue

Index	Type	Purpose	Effect
6	Text	The current LD plan associated with the subscription	None
7	Button	Begin adding a credit to the subscription	Open “Credit Subscription” sub-dialogue
8	Button	Request a reprint of the bill for the selected billing period	Ask Printer to print bill for the given billing period [9]
9	Text	Display the currently selected billing period	None
10	Button	Look at the next billing period	Update (1), (2), (4), and (9), with the billing information for the next billing period (relative to (9), if one exists)

Index	Type	Purpose	Effect
11	Button	Look at the previous billing period	Update (1), (2), (4), and (9) with the billing information for the previous billing period (relative to (9), if one exists) [8]
12	Button	Close dialogue, and return to "Customer Billing Operations" dialogue	Close dialogue, and return control to "Customer Billing Operations" dialogue
13	Button	Add credit to account	Add credit to account [19, 20]
14	Button	Abort crediting subscription	Without modifying the subscription, close dialogue, and return control to "View / Edit Subscription Billing Information" dialogue
15	Text Field	The Operator may enter in the value of the credit to be added	This value may be positive (such as with a bill payment) or negative (such as with a special charge that must be applied)

Table 40: View / Edit Subscription Billing Info and Credit Subscription Window Description

Index	Sequence of associated system calls
8	1) Bill::send()
10	1) Bill::findBill({9}period)
11	1) Bill::findBill({9}period)
13	1) Subscription::credit({15}amt,prd)

Table 41: View / Edit Subscription Billing Info and Credit Subscription Window Widget Mappings

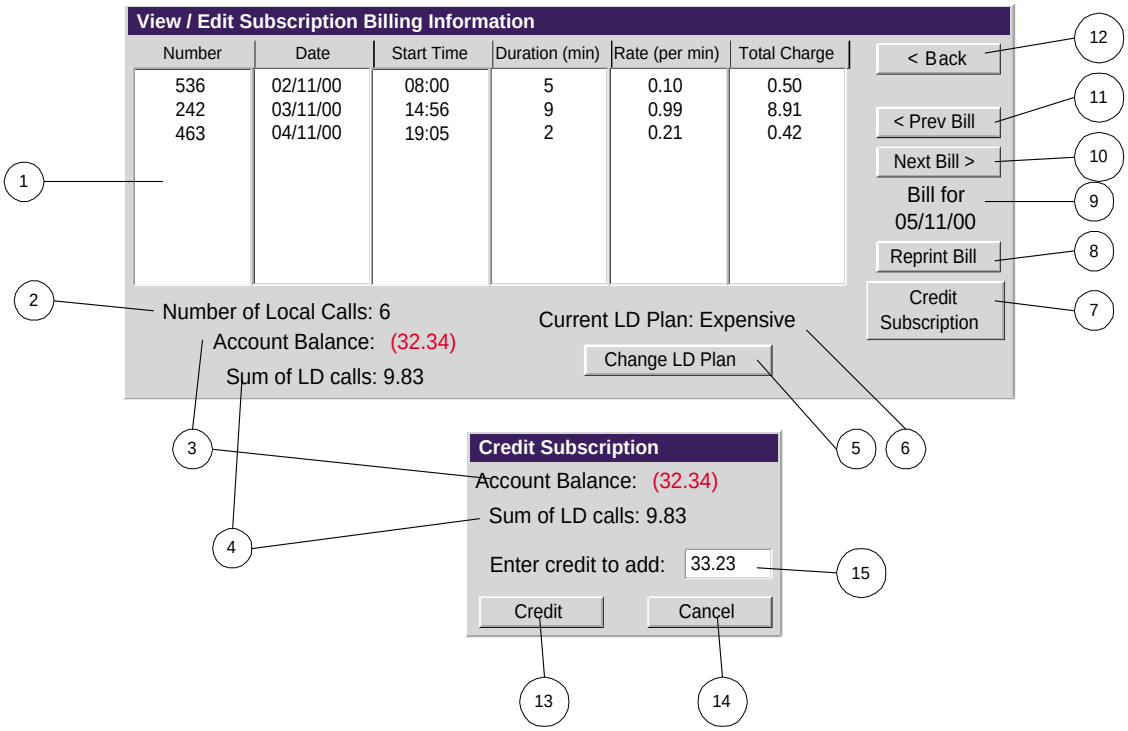

Figure 19: Edit Subscription Billing Info and Credit Subscription

To change a subscription's LD plan, this dialogue is used. When opened, it lists all the available LD plans, the time they are active, and their respective discounts. The notation used for the "Days Active" field are as follows: M=Monday, T=Tuesday, W=Wednesday, R=Thursday, F=Friday, S=Saturday, N=Sunday. Once a plan is selected, the Operator can apply the new plan to the subscription by pressing the "Change to Selected Plan" button. If no special plan is desired, the Operator may select the "Change to Default Plan" button. [11]

Index	Type	Purpose	Effect
1	List Boxes	All LD plans are listed here, with their corresponding information	Only one plan can be selected at a time
2	Button	Associate selected LD plan with subscription	Associate selected LD plan with subscription [11]
3	Button	Associate no LD plan with subscription (use only the default plan)	Associate no LD plan with subscription (use only the default plan)
4	Button	Abort changing the associated LD plan	Without modifying the subscription, close dialogue and return to the "View / Edit Subscription Billing Information" dialogue

Table 42: Change Subscription LD Plan Window Description

Index	Sequence of associated system calls
2	1) Subscription::setLDPlan({1}plnID)
3	1) Create new LD plan with 0 discount and associate it wit subscription

Table 43: Change Subscription LD Plan Window Widget Mappings

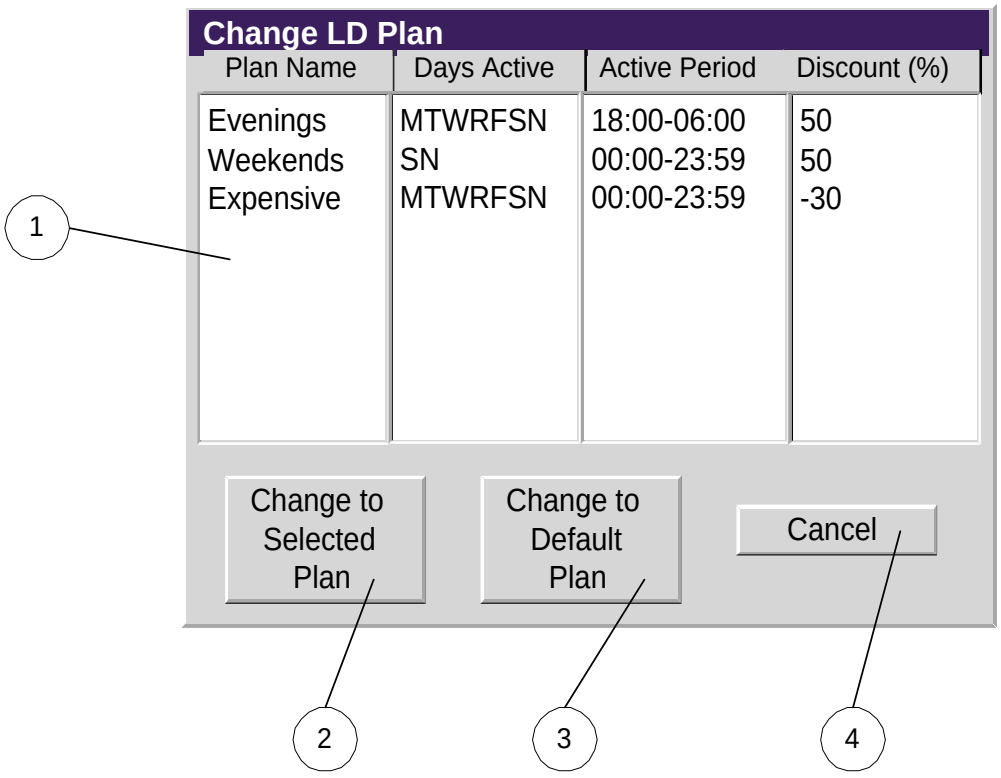

Figure 20: Change Subscription LD Plan

If the Operator wishes to modify any of the system-wide billing information, this dialogue is used. When opened, all available LD plans are listed. If the Operator wishes to add or edit a LD plan, the “Add LD Plan” and “Edit LD Plan” buttons are used, respectively. To remove a LD plan, the “Delete LD Plan” button is used.

For each exchange, the Operator may modify the local monthly billing rate, the billing day, and the LD charges to other exchanges. [25, 26, 34] After changes are made, the “Update Changes” button is pressed to commit those changes.

Whenever any LD plans are added, or modified, or exchange billing info is modified, a message confirming the changes is displayed in the “Status Messages” area. [27, 28, 29]

Index	Type	Purpose	Effect

1	List Boxes	List all the LD plans, with their corresponding attributes	Only one can be selected at a time
2	Text	This area displays information messages, regarding the results of operations on the list of LD plans (add, edit, delete), or results of an update to an exchanges billing information (7)	None
3	Scroller	Allows the Operator to select the exchange to view and edit the billing information for	Takes on values between 1 and 9.
4	Text Field	The local monthly rate for the exchange	The Operator may change this to any positive value
5	Text Field	The day of the month the exchange's customers are billed on	The Operator may change this to any day between 1-31.
6	Text Fields	The LD rates from the exchange in (3) to the other 8 exchanges	These may be any positive value. The field corresponding to the exchange selected in (3) is blank, and any changes to it are ignored.
7	Button	Update the changes to the selected exchange's billing information	Update the changes to the selected exchange's billing information [25, 26, 34]
8	Button	Delete the selected LD plan	Remove the selected LD plan from the system [29]
9	Button	Begin adding a new LD plan	Open "Add / Edit LD Plan" dialogue with all input fields empty [27, 28]
10	Button	Begin editing a new LD plan	Open "Add / Edit LD Plan" dialogue with all input fields populated from data from (1) (selected plan).

11	Button	Close dialogue, and return to “SX4 OAM Main” dialogue	Close dialogue, and return control to “SX4 OAM Main” dialogue

Table 45: Billing Plan Operations Window Description

Index	Sequence of associated system calls
3	1) Exchange::findExchange({3}ex)
7	1) Exchange::setLocalRate({4}rate) 2) BillingDaemon.billingDay = {5} 3) For each exchange rates{6}, use Exchange::setLDRate(ex, {6}rate)
8	1) LDPlan::discount=0

Table 46: Billing Plan Operations Window Widget Mappings

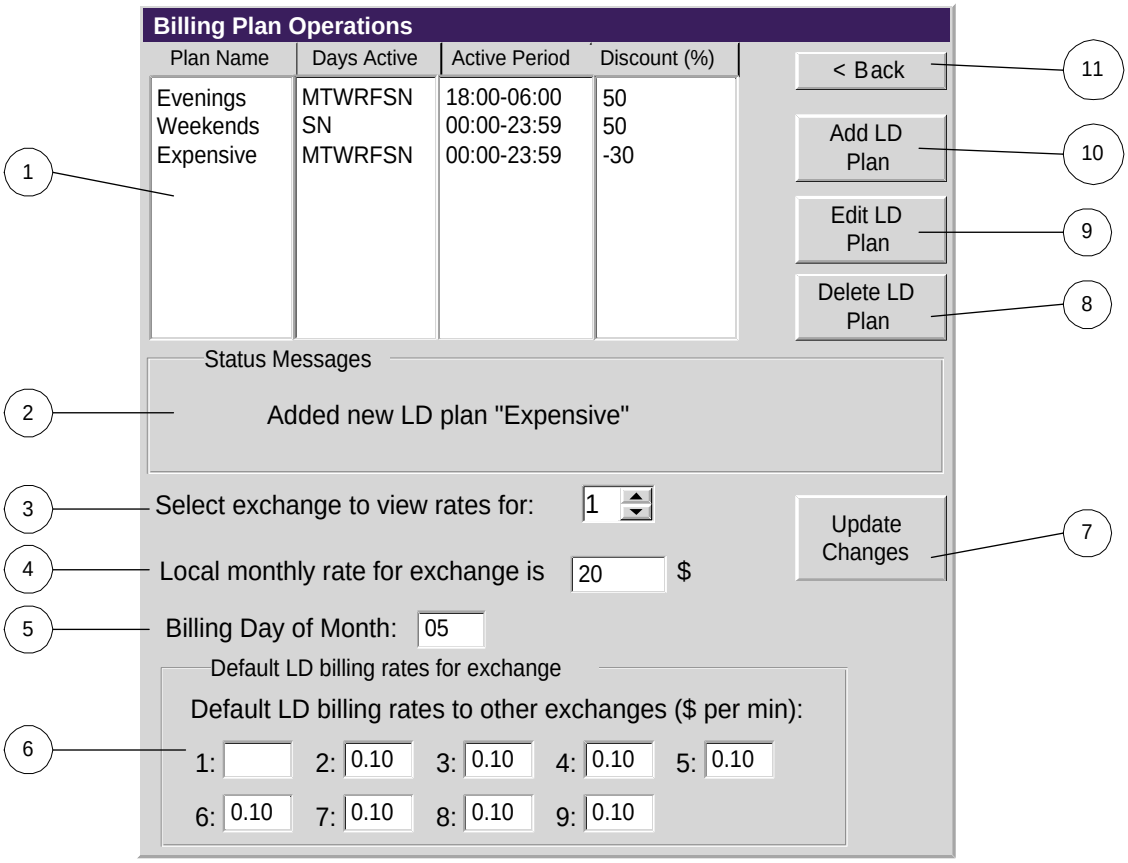

Figure 21: Billing Plan Operations

To add or edit a LD plan, this dialogue is used. If editing a plan, the selected plan is displayed in the “Plan Name” field, otherwise it is initially blank. All the plans available are listed in the drop-down box. The Operator may change the name in the “Plan Name” entry field. If the entered

name matches an existing LD plan, then that plan is updated, otherwise a new plan is created. The Operator may select the attributes of the plan (days active, activation period, and discount) here. To accept the changes, the Operator uses the “Commit” button. [27, 28]

Index	Type	Purpose	Effect
1	Editable Drop-Down List	This specifies the name of the LD plan to modify/add. The Operator may directly enter in a name here.	The drop-down list contains the names of all the current LD plans.
2	Check Boxes	Used by Operator to designate which days the LD plan is active on.	None
3	Text Fields	Used by Operator to specify the time period the LD plan is active	None
4	Text Field	Used by Operator to specify the discount for this LD plan	This can be a positive, or negative value. A negative value could be used, for example, when a customer’s connection involves a special expense (like an isolated and distant line)
5	Button	Apply the changes made to the list of LD plans	If a LD plan with the same name as (1) exists, then it is updated with the attributes from (2), (3), and (4). Otherwise, a new LD plan is created.
6	Button	Abort adding/editing the LD plan	Without changing the list of LD plans, close dialogue, and return control to “Billing Plan Operations” dialogue

Table 47: Add / Edit LD Plan Window Description

Index	Sequence of associated system calls
5	1a) LDPlan::discount = {4}dcnt 2a) LDPlan::period = {2,3}prd 1b) Send create LD plan event with {4}discount and {2,3}period

Table 48: Add / Edit LD Plan Window Widget Mappings

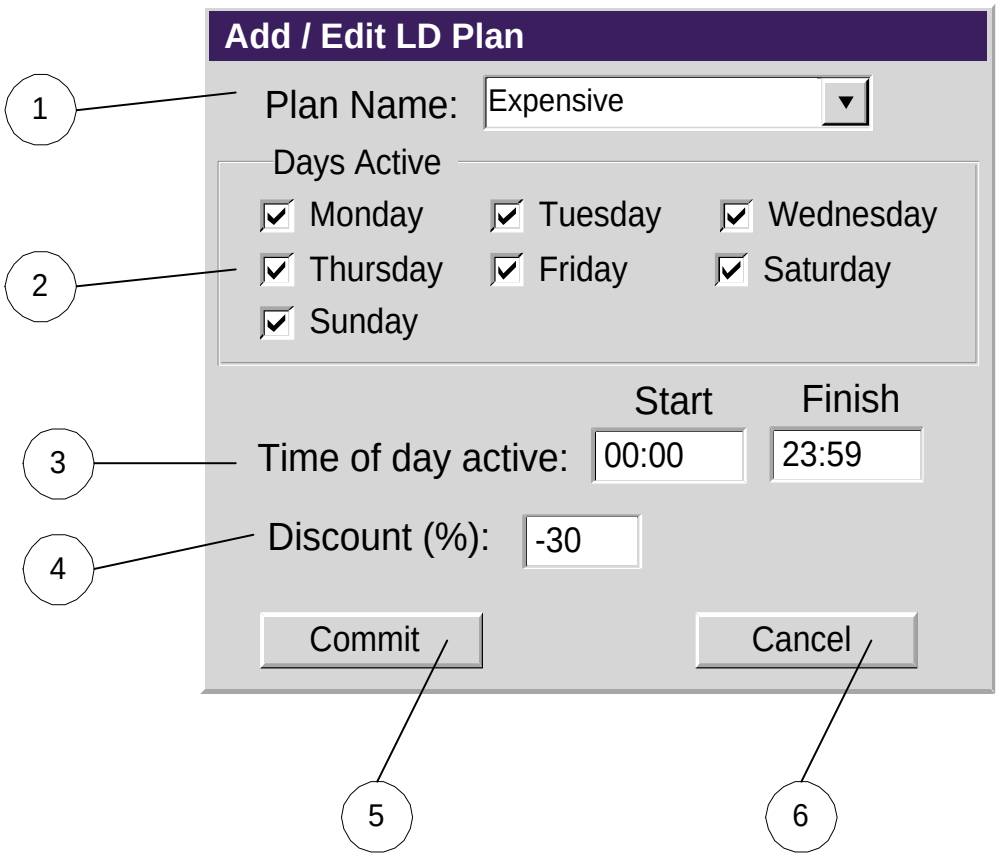

Figure 22: Add / Edit LD Plan

3.1.2 Hardware Interface

The following table shows the communication between the CU and OAM. The parameters that are used are specified in the table, List of Parameters and Values.

Message	Parameters	To	From	Meaning	Response to Event
init	EX	O&A	CU	A new CU has come online and is ready for use.	The O&A acknowledges the request
updateDB	EX, STAT, Shelf, Slot, COS	O&A	CU	A CU's cached exchange database has changed and the master exchange database should be updated.	The O&A updates the master exchange database based on the arguments received with the message.
updateDB	EX, DN, Shelf, Slot, STAT	CU	O&A	The O&A's master exchange database has changed and the corresponding CU's cached exchange database should be updated.	The CU updates the cached exchange database based on the arguments received with the message.
billCall	CallerEX, CallerShelf, CallSlot, dialedPhoneNumber, TimeOfCall, duration	B	CU	A call has been completed and must be billed and associated with the proper subscription.	The Billing object updates the corresponding customer's current bill by adding the call to the bill.

testEN	EX, Shelf, Slot	CU	M	A request has been made to test the specific hardware equipment at the location specified	The CU tests the corresponding device by running diagnostic tests
setStatus	EX, Shelf, Slot, ChangeSTAT	CU	M	A request has been entered by the operator that requires a change in the status of the hardware device	The CU sets the status bits on the requested hardware device
resetDevice	EX, Shelf, Slot	CU	M	A request has been entered that requires the hardware device be reset. This changes the hardware device to idle state, if it was in any other state.	The CU changes the corresponding card to be in idle state
queryDB	EX, Shelf, Slot	CU	OAM	A request has been entered in the operator's debug console that requests the CU's hardware device record for the device be outputed	The CU returns what its database holds for the device requested
contentsDB	EX, Shelf, Slot, DN, COS, STAT	OAM	CU	Results from a queryDB operation. The CU has a device that corresponds to the one requested	The OAM displays the results to the operator's debug console

invalidDBEntry	EX, Shelf, Slot	OAM	CU	Results from a queryDB operation. The CU does not have a device that corresponds to the one requested	The OAM displays the results to the operator's debug console

Table 50: Hardware Interface Table

3.1.3 Software Interface

None.

3.1.4 Communications Interface

There are ten types of messages sent between the CU and the OAM. The message protocol is defined as follows: the first argument of a message is the message keyword which is followed by zero or more integral arguments. The message keyword dictates the number of integral arguments contained in the message. The representation of the parameters (with parameters listed in order of sending) is shown in the table "List of parameters and values" on page50.

Message	To	From	Keyword	Parameters	# Param-s	Typical Example
Init	OAM	CU	“init”	EX	1	msg(“init”,1)
UpdateDB	OAM	CU	“updateDB”	EX, Shelf, Slot, STAT	4	msg(“updateDB”, 1, 2, 1, 020)
UpdateDB	CU	OAM	“updateDB”	EX, Shelf, Slot, DN, COS	5	msg(“updateDB”, 1, 23, 2 , 1 ,2)
BillCall	OAM	CU	“billCall”	CallerEX, CallerShelf, CallerSlot, dialedPho-neNumber, TimeOfCall, duration	6	msg(“billCall”, 1, 23, 2, 543, 2000100511205, 32)
TestEN	CU	OAM	“testEN”	EX, Shelf, Slot	3	msg(“testEN”, 3, 2, 1)
SetStatus	CU	OAM	“setStatus”	EX, Shelf, Slot, ChangeSTAT	4	msg(“setStatus”, 3, 2, 1, 020)
ResetDevice	CU	OAM	“resetDevice”	EX, Shelf, Slot	3	msg(“resetDevice”, 3, 2, 1)
QueryDB	CU	OAM	“queryDB”	EX, Shelf, Slot	3	msg(“queryDB”, 3, 2, 1)
ContentsDB	OAM	CU	“contentsDB”	EX, Shelf, Slot, DN, COS, STAT	6	msg(“contentsDB”, 3, 2, 1, 23, 0, 020)
InvalidDB-Entry	OAM	CU	“invalidDB-entry”	EX, Shelf, Slot	3	msg(“invalid-DBEntry, 3, 2, 1)

Table 51: Communications Interface Table

Parameter	Represents	Values
EX	Exchange number	0: reserved 19: exchange number 10255 : unused
Shelf	Shelf where line card is located	0-1: line interface shelf 2: trunk interface shelf 3-8: reserved 9: space and time switches, test and ring generator 10-255: unused
Slot	Slot where line card is located	0-31: valid slot 32-255: unused
DN	Dialed Number	0: unassigned trunk card 1-9: remote exchange 10-69: partial phone number 70-98: reserved for future use 99: designates an available line card 100-255: unused
COS	Class of services	bit 0: receive calls bit 1: can originate calls ? bit 2: can originate long distance calls ? bit 3-7: unused
STAT	Status of line card	bit 0: empty skit bit 1: offline failure bit 2: offline maintenance bit 3-5: test results bit 6-7: unused

Table 52: List of Parameters and Values

3.2 Behavior Requirements

The following tables and figures illustrate the various use cases of the system.

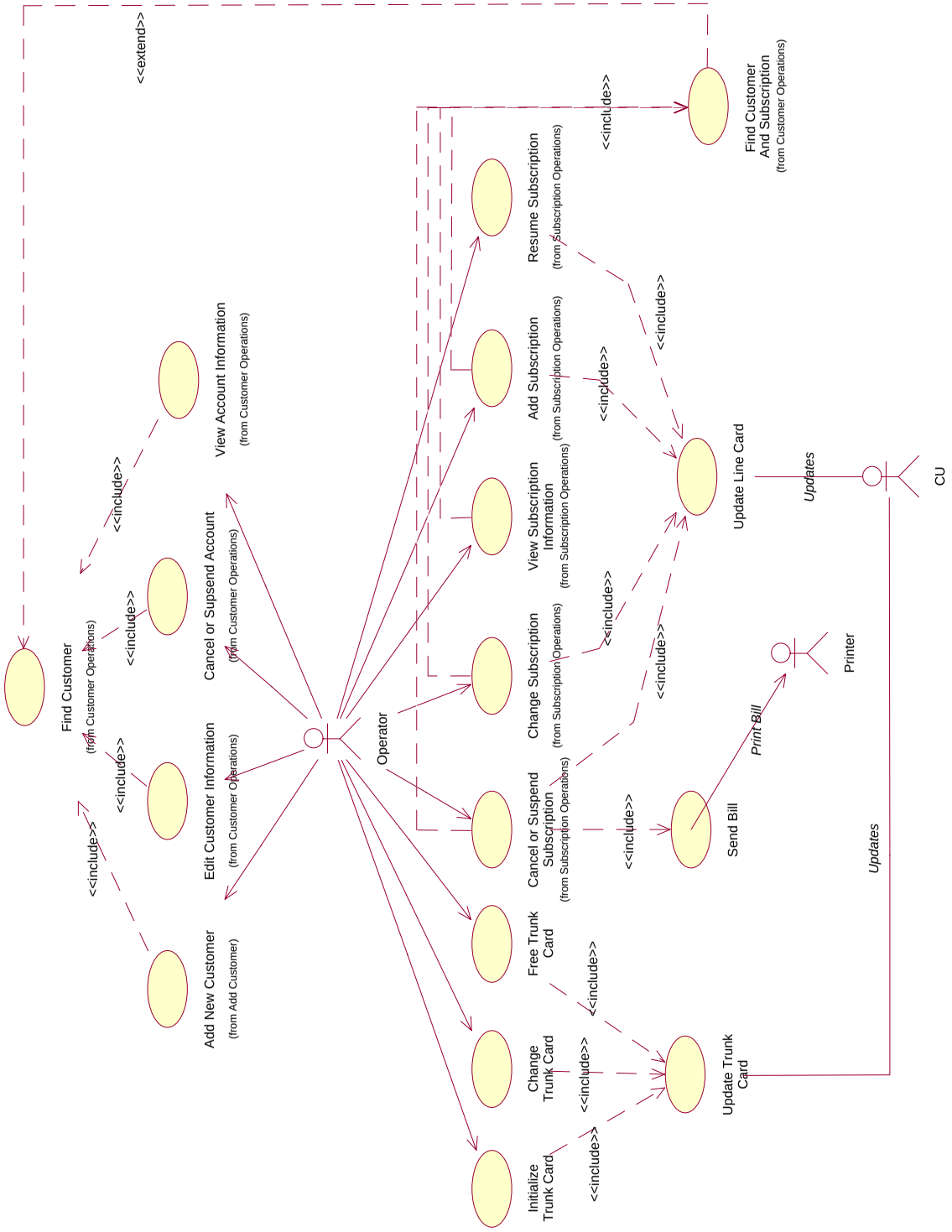

Figure 23: Use Case 1 - Operation and Administration

Use Case #	1. Operations and Administration
Goal in Context	Operator issues commands from the console for operations and administration tasks. Tasks include adding, modifying, customer accounts and subscriptions. Software takes appropriate action and returns results to the operator console.
Scope & Level	This is a high level summary. The scope is the operator's actions.
Preconditions	Operator has been authenticated.
Success End Condition	1. If an Add Customer action is requested, a new customer is created and the information recorded. [1] 2. If the operation is related to an existing customer account then the corresponding account is found and one of the following cases applies. [36] 2a. If an Edit Customer action is requested, the recorded customer information is updated. [2] 2b. If a Suspend account action is requested, the customer's account and all subscriptions are suspended. [35] 2c. If a Cancel account action is requested, the customer's account and all subscriptions are canceled and bills are sent. [3] 2d. If a View account action is requested, the customer's account and all records are visible to the operator. [4] 2e. If an Add Subscription action is requested, a new subscription is added to the customer's account. [21] 3. If the operation is related to an existing subscription, then the subscription and customer account are found, then one of the following cases applies. [38] 3a. If Cancel Subscription action is requested, the customer's subscription is no longer active and the phone number and line card are released. The bill for the current subscription is printed to the Printer. [10,16] 3b. If Suspend Subscription action is requested, the customer's subscription is set to not active, the phone number is still allocated, but the line card is released. [17] 3c. If Change Subscription action is requested, the customer's subscription is updated with the new information and possibly the line card updated. [22] 3d. If View Subscription action is requested, the customer's subscription information is returned. [23] 3e. If Resume Subscription action is requested, the customer's subscription is activated and line card updated. [18] 4. If Initialize Trunk Card is requested, the trunk card is initialized to virgin status. [6]

	5. If Change Trunk Card is requested, the trunk card is released and a new Trunk Card is allocated. [6] 6. If Free Trunk Card is requested, the trunk card is released. [6]
Failed End Condition	If the customer account cannot be found. (if not an add account) Then the desired action is not taken. If the customer subscription cannot be found (if not an add subscription) Then the desired action is not taken If the operation is a card op, and it fails, then the operation is not taken.
Primary, Secondary Actors	Operator, CU, Printer
Trigger	Request from operator is received.
DESCRIPTION	1. Operator requests action, either Add, Edit, Cancel/ Suspend, Resume, View Account or Add, Edit, Cancel/ Suspend, Resume, View Subscription or Initialize, Change or Free Trunk Card 2. The customer's account (and subscription) record is found if applicable. 3. The requested action is then taken.
EXTENSIONS	none
SUB-VARIATIONS	none

Table 54: Use Case 1 - Operations and Administration

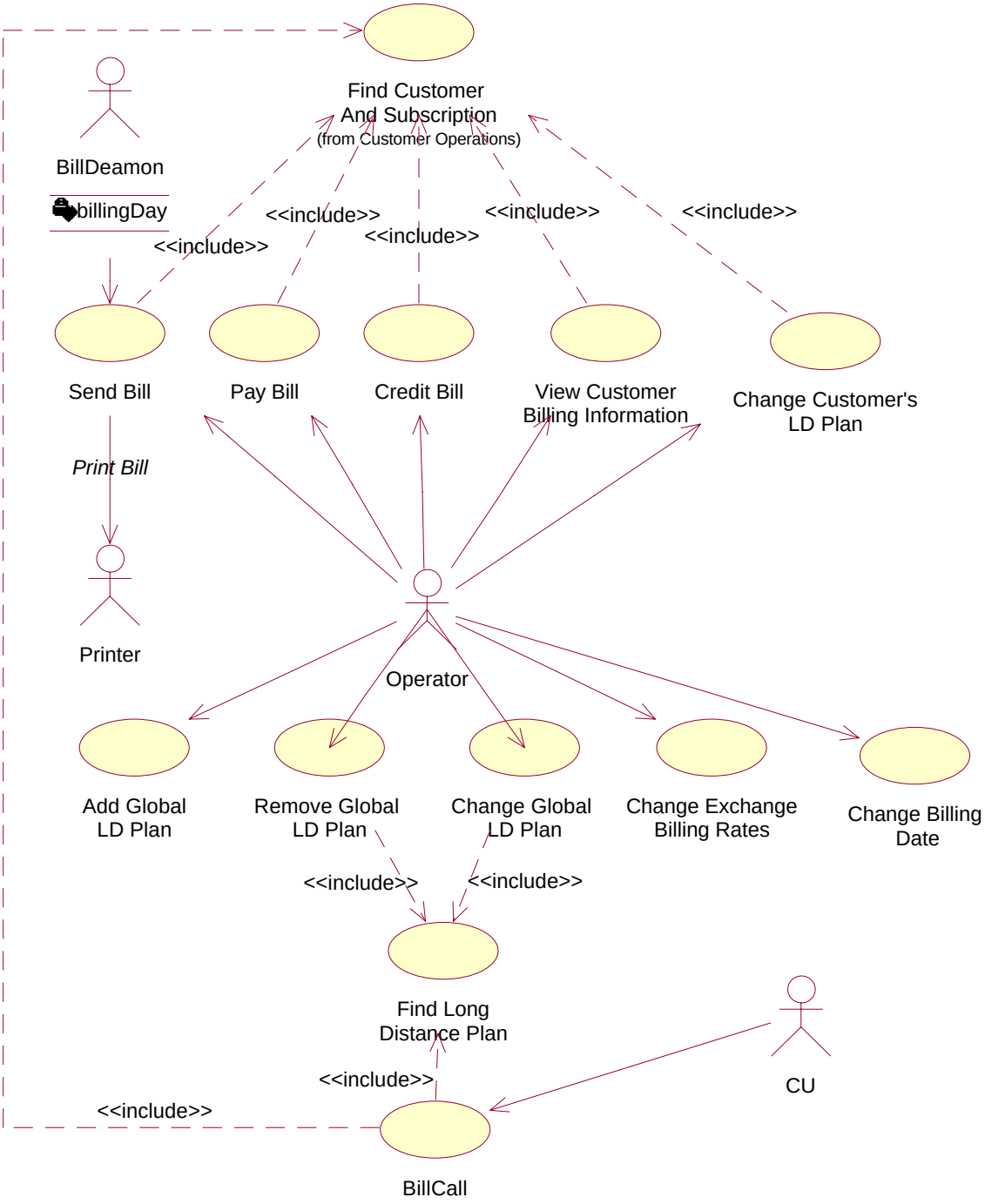

Figure 24: Use Case 2 - Billing

Use Case #	2. Billing
Goal in Context	Operator can issue billing related operations and the CU can bill calls that have been completed
Scope & Level	The level of this use case is high. The scope is from an operator and CU's view.

Preconditions	If an operator command, the operator is logged in and authenticated.
Success End Condition	1. If the operation is related to an existing customer subscription, then the subscription and customer account are found and one of the cases applies. [36, 38] 1a. If the operator requests a bill to be Sent (Send Bill) , then the bill is prepared and sent to the Printer. [9] 1b. If the operator files a Payment for a bill (Pay Bill), then the customer's account is credited with the amount paid. [19] 1c. If the operator wishes to view the customer's Billing history (View Customer Billing Information), then the information requested is returned. [8] 1d. If the operator requests to file an additional charge/credit on the subscription, then the appropriate monetary value charge/credit is applied to the subscription's current bill. [20] 1e. If the operator requests to change the subscription's LD Plan, then the new LD Plan of the subscription takes effect immediately. [11] 2. If the operator requests to change the billing rates of an exchange, then the LD and local rates are updated as the operator specified. [25, 26, 30] 3. If the operator requests to add a global LD plan, then the plan is added to the available LD plans. [27] 4. If the operator requests a change or remove of an existing LD plan, the LD plan is found and then one of the following cases applies. 4a. If the operator requests a removal of a global LD plan, the LD Plan's discount is then set to zero. [29] 4b. If the operator requests a change of a global LD plan, then the LD plan is updated immediately, and subscriptions associated with that plan start using the new rates immediately. [28] 5. If the CU sends a call completed message, then the call is billed to the associated subscription. [33] 6. If the Operator requests to change the date at which bills are sent out to customers, then the billing date is changed on the daemon which requests bills to be printed. [34]
Failed End Condition	1. If the operation is related to an existing subscription and it cannot be found then an error is returned and the action is not taken.

	2. If the change to the exchange's billing rates fails, then the action is not taken. 3. If the addition of a new global LD plan fails, then the operator is notified of the error. 4. If the change or remove of a global LD fails, then no changes occur to any subscriptions or rates. 5. If the call is completed and the update to the subscription's bill fails, then the operator is notified of the error.
Primary, Secondary Actors	Operator, CU, Printer
Trigger	Operator requests billing related operation, or the CU bills a . call
DESCRIPTION	1. The operator requests a bill, viewing billing information, records bill payment, and updates billing information. 2. The CU sends call completed messages and the calls are recorded in a subscription's current bill.
EXTENSIONS	none
SUB-VARIATIONS	none.

Table 56: Use Case 2 - Billing

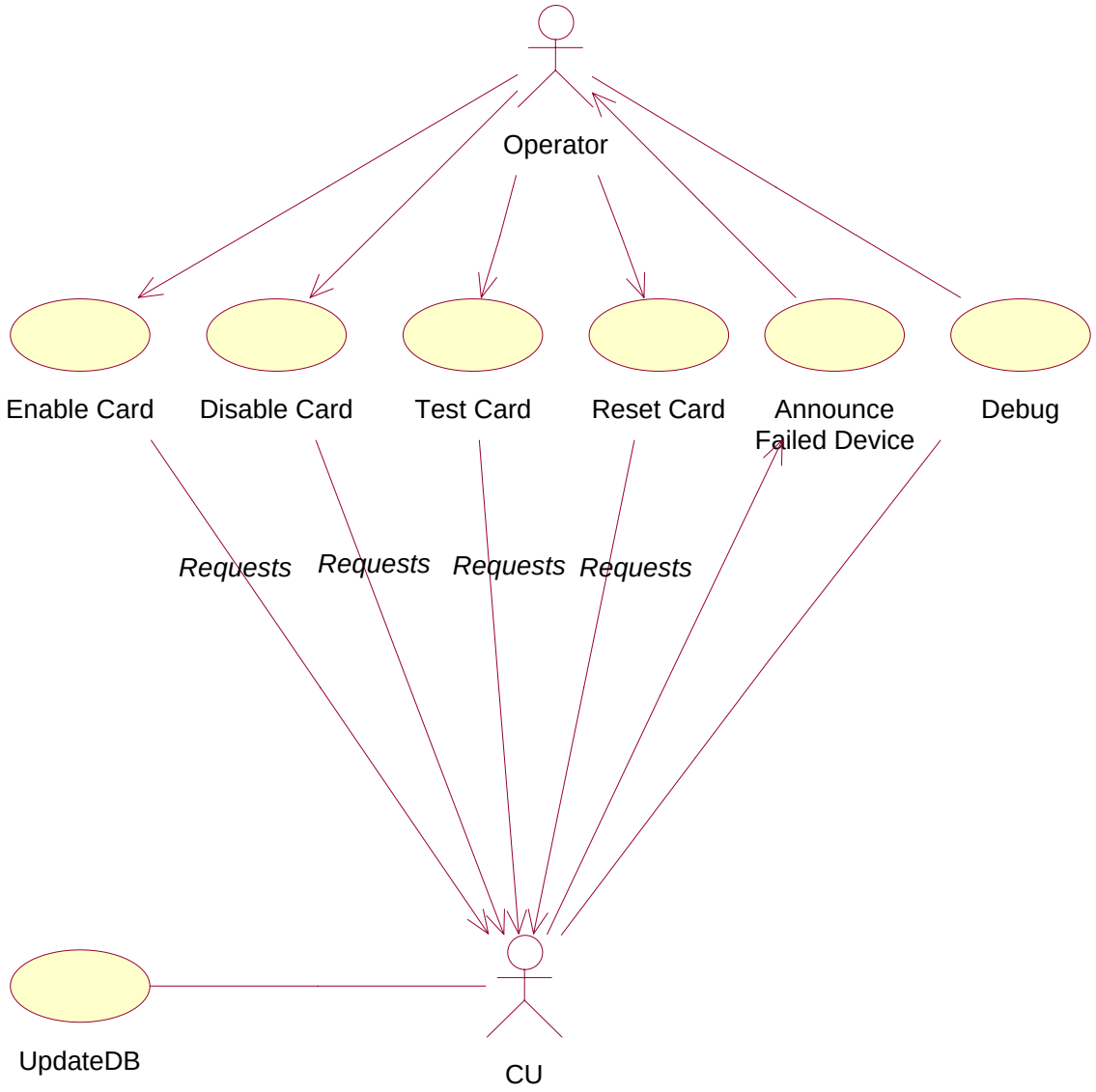

Figure 25: Use Case 3 - Maintenance

Use Case #	3. Maintenance
Goal in Context	The operator requests testing and maintenance of hardware . devices. The CU sends update messages to the OAM.
Scope & Level	The level of this use case is high. The scope is from an operator's and CU's view.
Preconditions	For operation commands the operator is authenicated.
Success End Condition	1. The operator requests a line card operation, then one of the following cases applies. And it is asserted that the line card is to be valid. 1a. The operator requests an enable card operation, then the card is correctly enabled by setting the status bits. [39]

	1b. The operator requests a disable card operation, then the card is correctly disabled by setting the status bits. [40] 1c. The operator requests that a card be tested, then the CU performs hardware tests on that device. If an error is found it will be reported by a CU -> UpdateDB message [41] 1d. The operator requests a card to be reset. The associated card is then reset by the CU. [42] 2. The operator requests some debug messages be sent to the CU, they are displayed to the operator's debug console with the results from the CU. [37] 3. The CU announces that a hardware device has failed, then the operator is notified of the failed device. [5]
Failed End Condition	1. The operator requests a card that does not exist, an error is written to the console. If the operation fails on the card, then the card must be defective and the CU will return an error via a UpdateDB message with a hardware failure status. 2. No failed end condition for this case, errors are written to the console. 3. No failed end condition for this case, the failure message is held in a queue until the operator logs in if logged out so no failure messages will be missed.
Primary, Secondary Actors	Operator, CU
Trigger	Operator requests a card test/command. CU announces that a hardware device has failed.
DESCRIPTION	1. The system handles card requests and passes the message along to the CU, updating the DB when appropriate. 2. The system receives failure messages from the CU and displays them to the operator.
EXTENSIONS	none
SUB-VARIATIONS	none.

Table 58: Use Case 3 - Maintenance

3.2.1 Object Structures/Process Structure Requirements

The following diagram represents the class view of the OAM system.

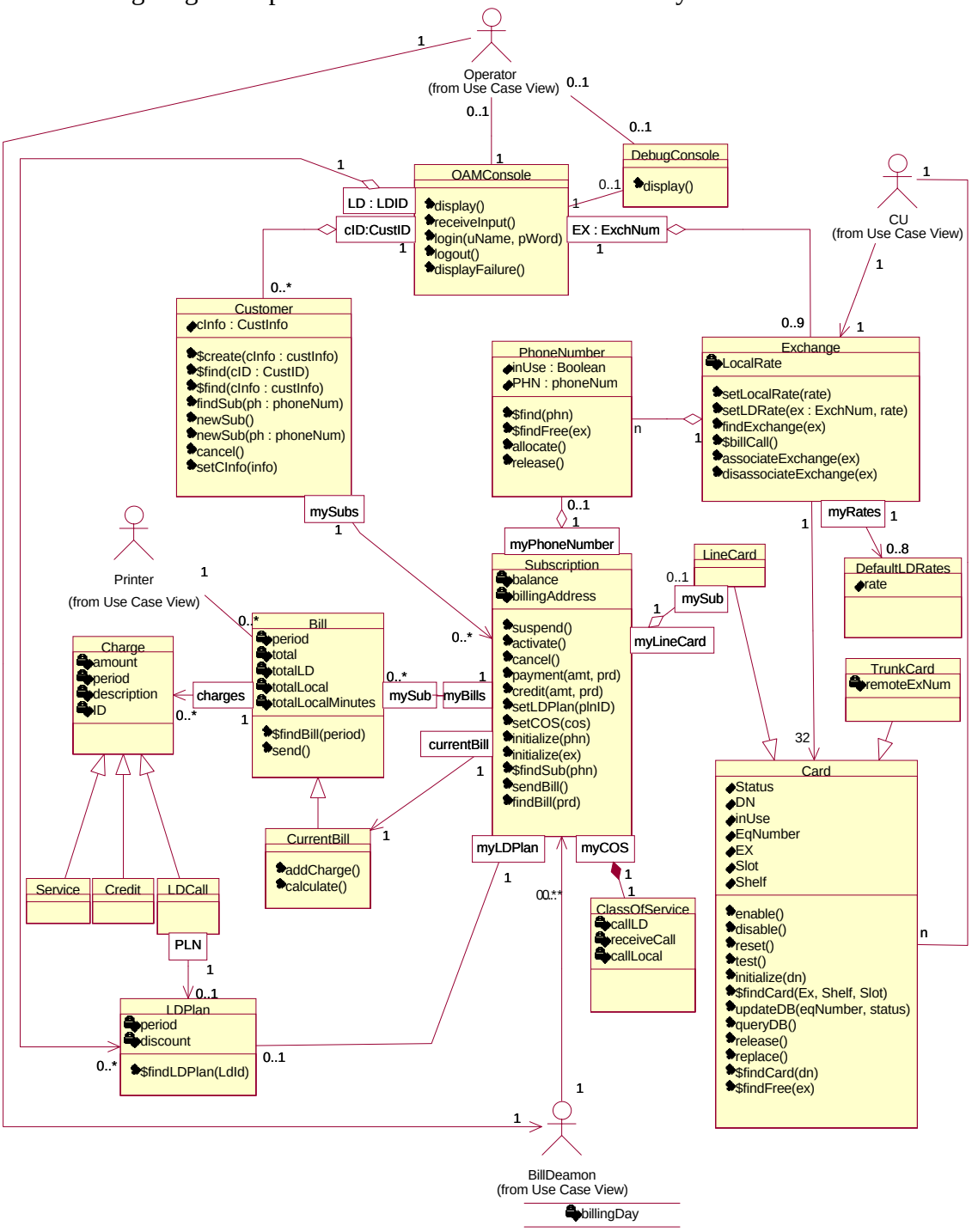

Figure 26: UML Class View

The class model is described in the following paragraphs. Further technical specification is provided in the Data Dictionary.

The class model uses a Model, View, Controller (MVC) structure. The OAM Console which represents the view in the model, allows the Operator to request and receive commands from the system. This console displays data, failure messages, and receives input from the operator. An auxiliary debugging console (DebugConsole) is provided for the abstraction of the Debugging Console from the normal operations Console. For authentication purposes the Operator is not allowed to use the OAM Console until a valid username and password have been entered. [14] The OAM Console has relationships to Customer records, Exchanges and Long Distance Plans. All of the operations of an operator can be issued via these three "channels." The requests that the operator issues are handled by the controllers, objects such as Subscription, Customer, and Exchange.

The Customer class can have a series of subscriptions that can be active, canceled or suspended. A requirement is that all data is maintained in the system and that old subscriptions are not to be destroyed even though they may no longer be active. [15] The operations provided through the Customer class are ones to create and maintain subscriptions and customers. The ability to add customers is provided on the model. [1] Updating of customer information can be done by calling the setCInfo method with the new information. Hence, the requirement for updating customer information can be executed through this model. [2] Subscriptions can be also found from a customer, that is a search can be done on the subscriptions that a customer has to find a specific one. [36] New subscriptions can also be added by either requesting a subscription in a particular exchange or providing a phone number that you request (this also specifies the exchange by the first digit). [21] A corresponding Subscription will be created.

The Subscription class maintains a LDPlan, PhoneNumber, LineCard, COS, current bill and a series of old bills. The Subscription represents the service provided on a telephone line to a customer. A subscription can be cancelled, suspended, activated, initialized and its properties edited. [10, 11, 17, 18, 21] When a suspend occurs the line card that is associated with the class is deallocated. When a cancel occurs, the account is suspended and the phone number is freed. An activate finds a free line card in the exchange and allocates it. An initialize finds a free phone number and allocates it. The initialize is done on the creation of the object so that it can acquire a phone number. The class interacts with the BillDeamon in that the BillDeamon on a scheduled interval requests sendBill for all the subscriptions in the system. [24] The Operator can adjust the billing period by calling a setBillDate on the BillDeamon. [34] Note that this does not require any explicit class or methods on that class, it is Operator to BillDeamon communication only. When the BillDeamon signals that the period has ended, then the currentBill is printed and placed in the list of previous bills. A new bill is created and then set as the current one. [24] The Operator can record payment and credits; these are recorded against the current bill, which is the bill where all charges are recorded for the current billing period. If a customer wishes to change a subscription's phone number within an exchange, the Operator must cancel the card subscription, and start a new one. In this case, the customer is guaranteed to get a phone number and line card. If the customer is changing a number to one outside the current exchange, then the customer is not guaranteed to get a line card or phone number since the remote exchange may be full.

The Bill class contains a series of Charges. There is a extension of bill which is CurrentBill that contains operations to add charges to a bill and calculate the total of the bill. These operations are not permitted on past bills since they are read-only. The series of charges have three extensions: Credit, Service, LDCall. The Credit is used to credit the account when an operator either gives

the account a credit or a payment. The Service is used to represent any service charges, including monthly local rate service. The LDCall is used to charge LD calls to a bill. [33] The Charge class has four attributes which the subclasses use. Each Charge has a unique id so that it can be clearly referenced on a bill. A description, amount and period are used in different ways among the extensions. A LDCall is also associated with a LDPlan, so that it is clear what calls belong to which LD Plan if there were to be a change in the subscription's LDPlan during the month.

The LDPlan represents the Long Distance Calling plan; it has a discount and a period. The discount is a percentage discount off of the normal LD calling rates. The period is a TimePeriod for which the LDPlan is in effect. [31] For example, this field will store between 7pm-7am if that was the time for the discount to be effective. There is a find, create and remove on the class. These operations were the ones that are required from the specification. A Operator can create new LD Plans or remove existing ones. [27, 28, 29]

The Exchange Class is the class that has knowledge of phone numbers and cards associated with that exchange. It has a series of n cards, and m phone numbers. Note that m > n. The Exchange class also stores the long distance calling rates to other exchanges through the class DefaultLDRates, this is to ensure that the Operator can assign a long distance rate from each exchange to every other one. [30] A local rate is an attribute of the class which is the monthly service charge of clients who have subscriptions. The Exchange allows the operator to set these rates. The Exchange also interacts with the CU through billing. The CU will signal that a call has been completed, the Exchange will then bill this call to the appropriate subscription. It does this by finding the line card associated with the device that made the call and finds the associated subscription. [33] The operator can also enable, disable, test, and reset line cards through the Exchange since it maintains the list of cards in its exchange. The operator can associate and disassociate an exchange to another exchange, that is having a trunk card between two exchanges or not. It can do this by using the associateExchange and disassociateExchange methods on the class. These methods allocate and initialize a trunk card, and conversely, deallocate a trunk card.

The Card class has all methods of a hardware card. The device can be enabled, disable, reset, tested. [39, 40, 41, 42] The device also supports initialization, replacing and releasing. The initialize sets the dialed number for the card. Cards have two generalizations there is a Trunk Card and Line Card which generalize to a Card. A Line Card is associated with a subscription. A Trunk Card has an exchange that it is associated with, note that this is the same as the DN. The CU can connect to these devices directly to perform updates and return results. The operator can perform queries on these devices through the Debug Console.

The PhoneNumber class represents phone numbers that are in the system and related to an exchange. The PhoneNumber has an inUse flag and an integer storing its associated phone number as well as a series of operations to find free ones and allocate/deallocate them. A PhoneNumber is related to an Exchange an Subscription.

The above paragraphs describe how the relationships interacted amongst the other objects. The Data Dictionary contains further documentation such as pre-conditions and post-conditions on the methods.

3.2.2 Dynamic Requirements

The following state and sequence diagrams illustrate the flow of control though the OAM system. The state chart can be followed for all acceptable requests, thus providing a map of the the potential control/data flow. Whereas, the sequence diagrams provided an illustration of the actual control/data flow for a specific scenario.

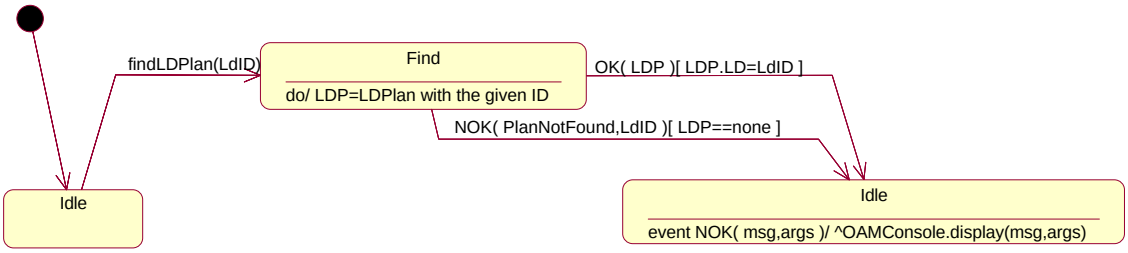

Figure 27:LD Plan State Diagram

The LD Plan state diagram has one main activity, finding a LD Plan. It shows the operator of finding a Long Distance Plan. If the LD Plan is found then it is returned in a OK return result. Otherwise, NOK is returned with a LD Plan not found. The LDPlan is called from the OAMConsole when an Operator does LD billing operations.

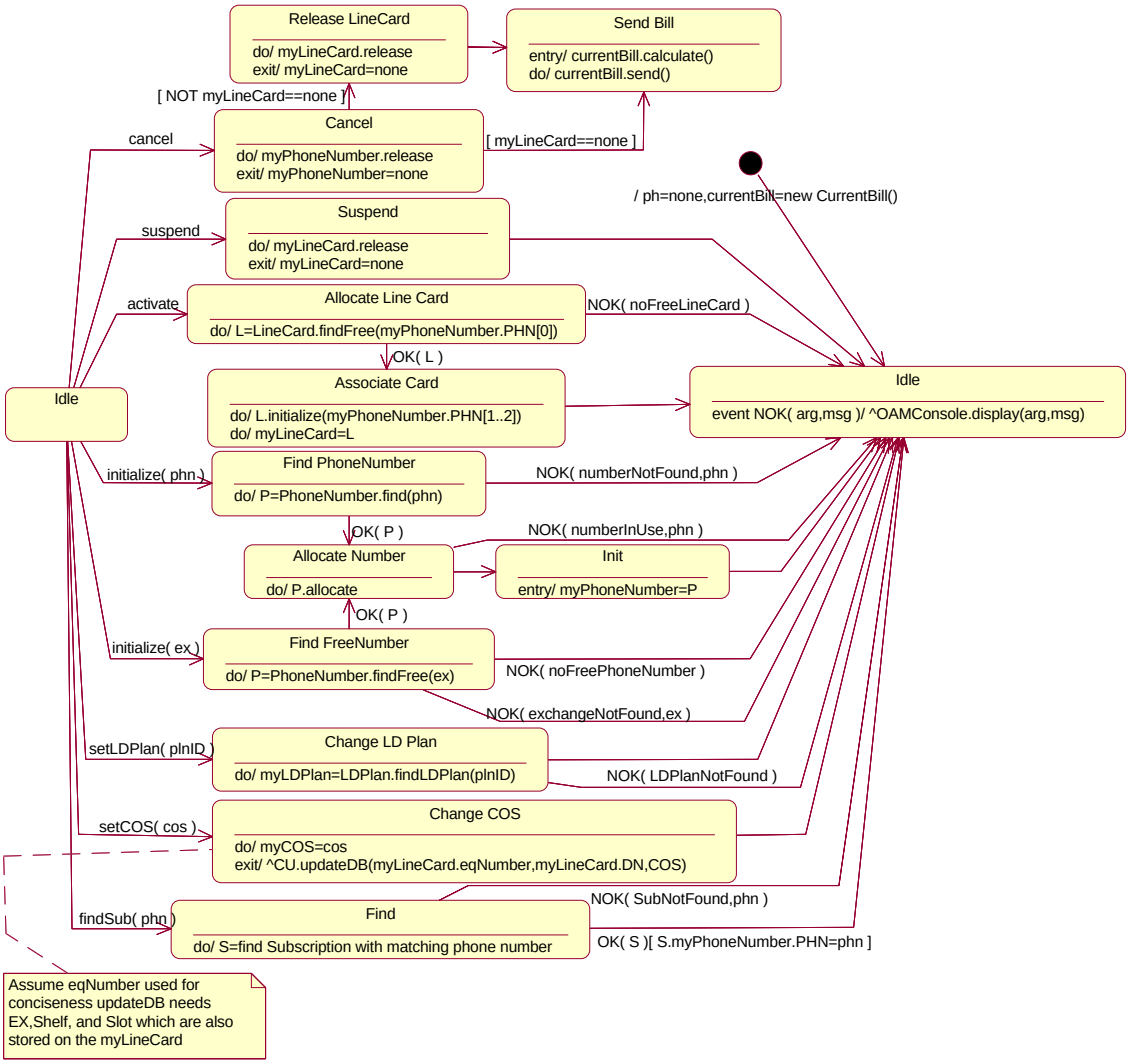

Figure 28:Subscription Admin State Diagram

The Subscription Admin State Diagram shows the activities in the Subscription class that are Administration related. Below there is a State Diagram for Subscription that has billing related activities. The diagram shows the following events: suspend, cancel, activate, initialize, setLD-Plan, setCOS and findSub. When a suspend occurs, the line card is released and it returns to the idle state. A cancel event will cause the phone number and line card to be released and a bill is printed/sent. An activate event will transition to a state where a line card is acquired, if none are available an error is returned. An initialize event will transition to either find a free phone number (if just an exchange was provided) or to check if the number requested is available. If either fails, then an error is returned. Particularly, the FindFreeNumber state will return NOK if there is no room in the exchange or there are no phone numbers left. The Allocate Number state will return NOK if the phone number is already in use. After that a line card is allocated and initialized, if one is available. SetLDPlan and setCOS events transition to states where the associated member is set. The SetLDPlan can return a NOK if the LDPlan specified is not found as a valid LDPlan, otherwise it returns OK. The setCOS requires that the CU be updated, so it sends an updateDB message.

FindSub will find a subscription given a phoneNum, it will return either NOK if a subscription is not found, or OK with the subscription that was found.

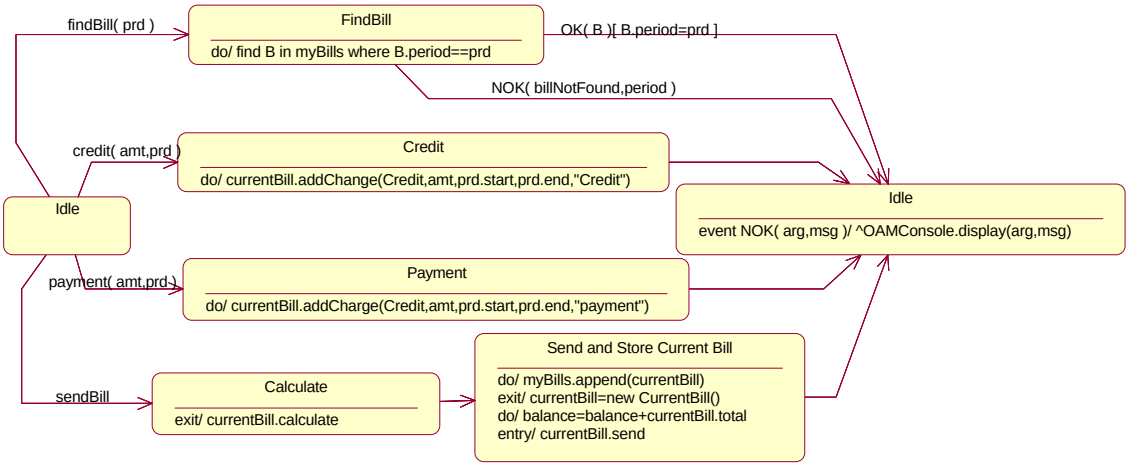

Figure 29:Subscription Billing State Diagram

The Subscription Billing State Diagram shows four events; findBill, payment, credit and sendBill. There four main activities corresponding to input event. When the subscription receives a findBill event and a period, a search is done to find a bill that corresponds to the period provided. If a bill is not found a NOK is returned to the Operator, otherwise an OK with the Bill is returned. If the Subscription receives a credit event, a charge with type credit is applied to the Subscription's current bill. If the subscription receives a payment event, a charge with type payment will be applied to the current bill. The BillDeamon can send events to the Subscriptions signaling the end of the billing period. When this occurs, the Subscription receives a sendBill message. The transition into ChargeForService is taken; in this state the Current Bill is found and calculated. Once completed, the transition into the Send And Store Current Bill is taken. This will cause the current bill to be added to the list of previous bills and a copy of the bill to be sent to the printer.

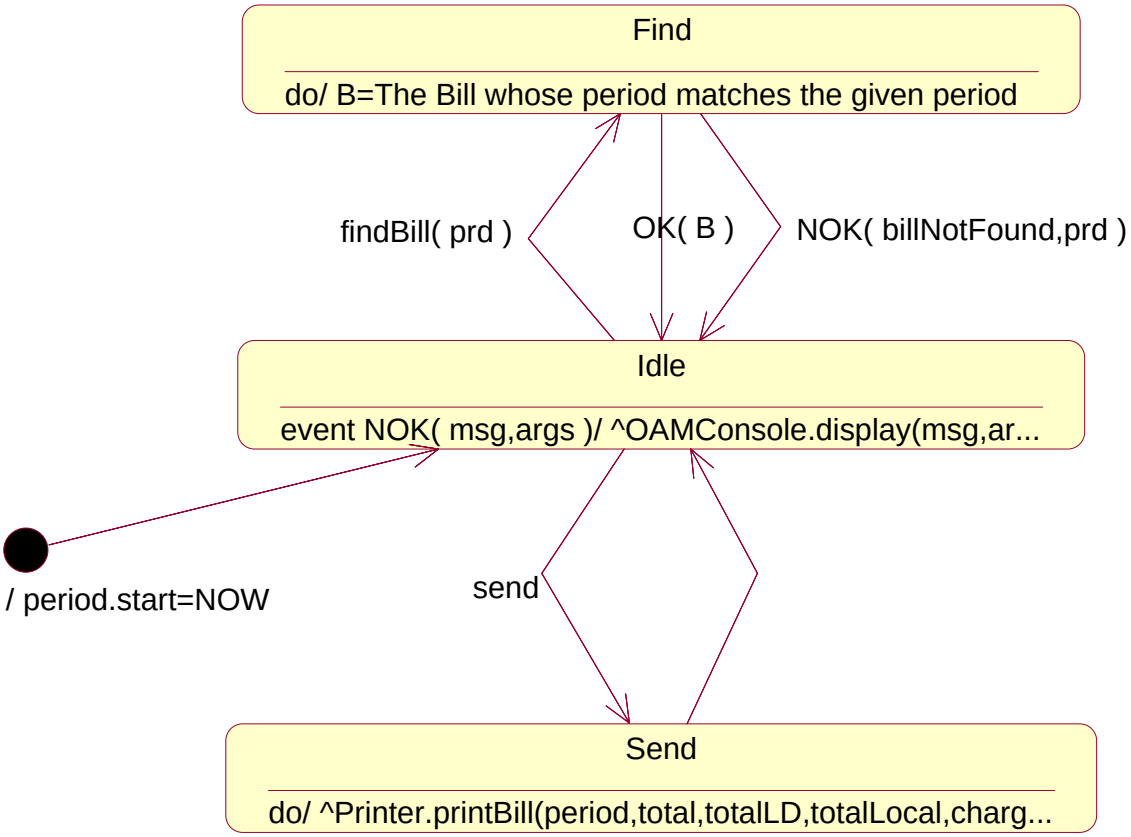

Figure 30:Bill State Diagram

The Bill State Diagram shows two events and activities. These events and associated activities are finding a bill and sending a bill. The Send Bill activity sends the bill to the Printer, and returns to the idle state. The Find Bill event transitions into FindBill which finds a bill given a time period. The Find can either return OK with the bill returned, or NOK with a message that the bill was not found.

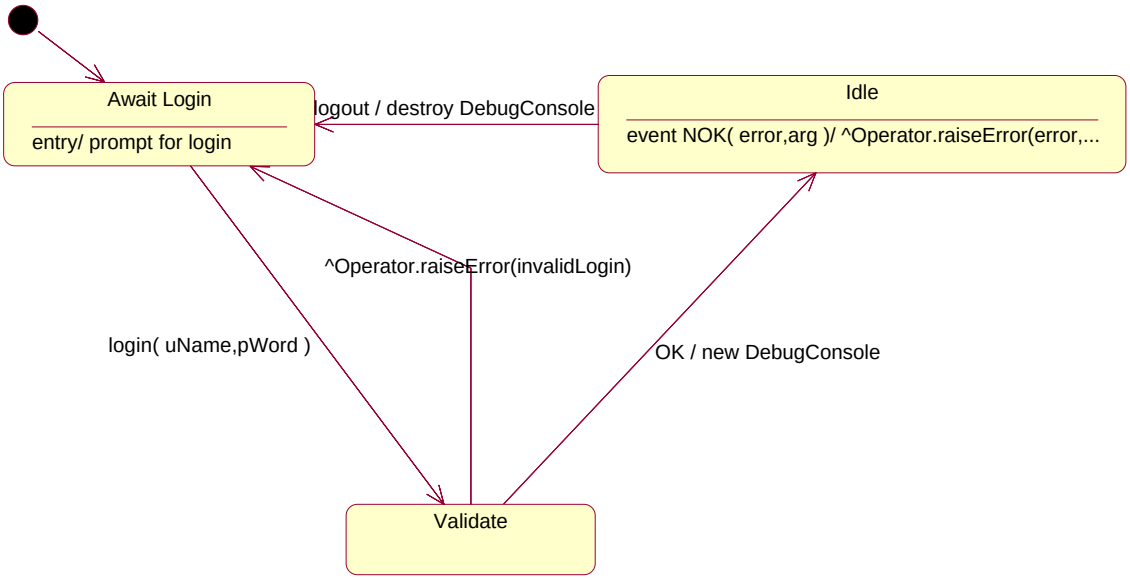

Figure 31: OAMConsole State Diagram

The OAMConsole State Diagram shows only the Authentication in the class. Displaying and ReceivingInput are trivial and thus not shown. There is a first transition that is required for the OAMConsole to accept input events. The OAMConsole begins in the AwaitLogin state, this state will transition out of it when a login(userName, password) event is received into the Validate state. The validate state checks to see if the login is valid and if so, it transitions to the Idle state where it can begin receiving Operator commands. If the validate fails then an error is displayed to the Operator stating that the userName, password was invalid. Upon the logout event, the OAMConsole will transition from the Idle state to the AwaitLogin state.

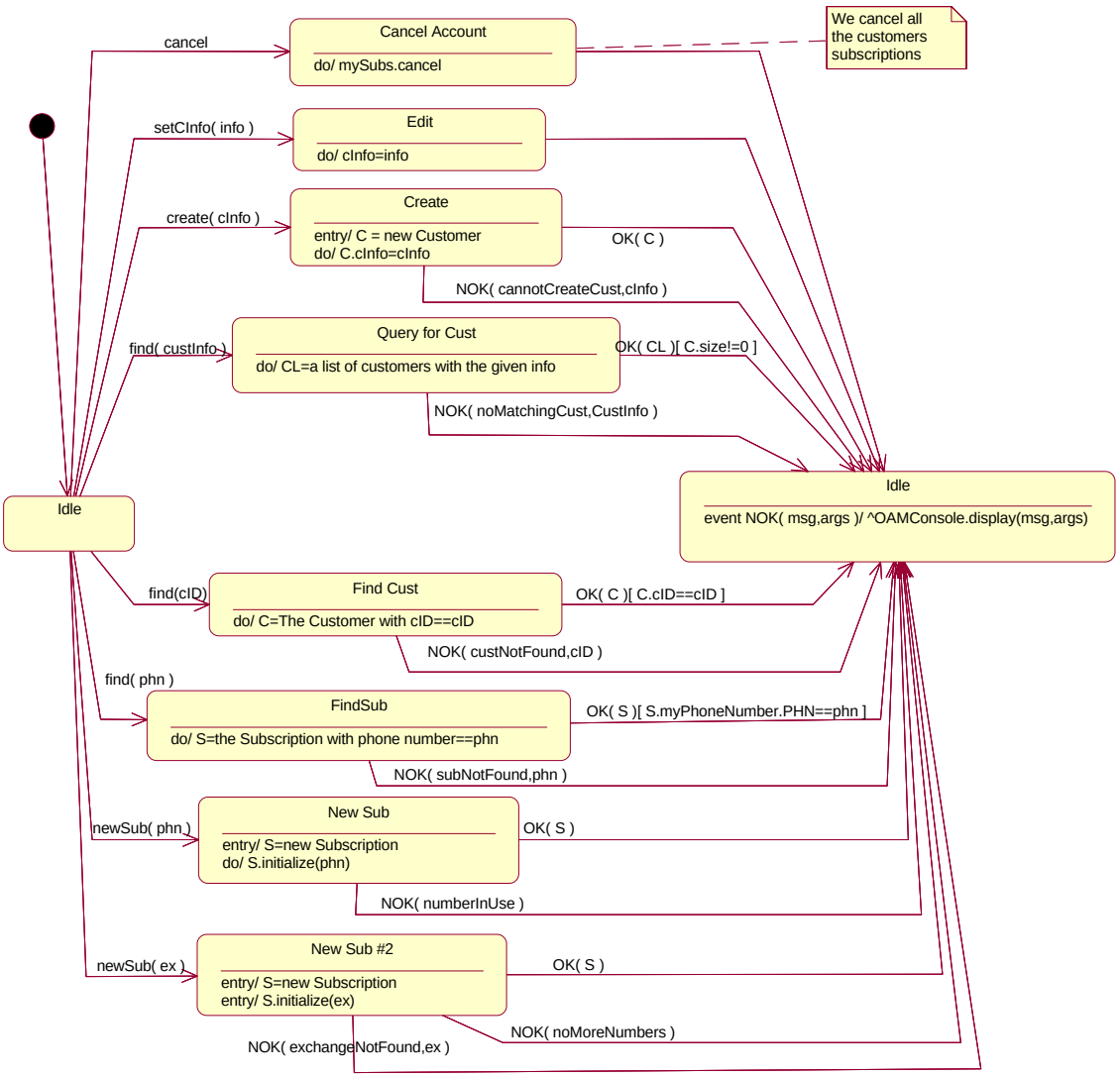

Figure 32: Customer State Diagram

The Customer State Diagram shows the Customer events of create, find, setCInfo, cancel, newSub. When the class receives a create event a transition is performed from the idle state to the Create state. This state then generates a Customer and assigns the customer information the cInfo provided. Either an OK with the Customer is returned or a NOK is returned as an error. The setCInfo event transitions into the Edit state in which the Customer information is updated to the value passed in. When a cancel event is received all of the related subscriptions are cancelled in the Cancel state. Note that the Customer information is not removed, nor are the subscriptions. All information is retained in the system. On a find event, one of three states may be transitioned to depending on the parameters. If a CId is passed in, then a find for the customer with the CId must be performed, causing a transition into the FindCust state. If a cInfo is passed in, then a find for customers that match the data provided is performed, causing a transition into the QueryForCust state. Finally, if the parameter passed in is of type phoneNum, then it is a find for a Subscription causing a transition into the FindSub state.

In the Query For Cust state a list of customers that match the given CustInfo is built and

returned to the Operator. The Find Cust state and activity finds the customer that is associated with the CustId, an OK can be returned if a customer is found with the corresponding cId, otherwise an error is returned in the form of a NOK. The Find Sub state and activity accept a phoneNum as a parameter and call the Subscription::Find to find the associated subscription. The result from finding the subscription will be returned if and only if the phone number matches one of the phone numbers owned by this customer. The newSub event can transition into two states depending on parameter. If an exchange is specified then the current state becomes New Sub #2 which allocates a phone number in an exchange. If there are no more phone numbers or th exchange is invalid, a NOK is returned, otherwise a OK is returned with the newly created Subscription. If a phone number is specified then the current state becomes New Sub which allocates a specific phone number in an exchange. If the phoneNumber requested is already in use, a NOK is returned to the Operator, otherwise the Subscription is created.

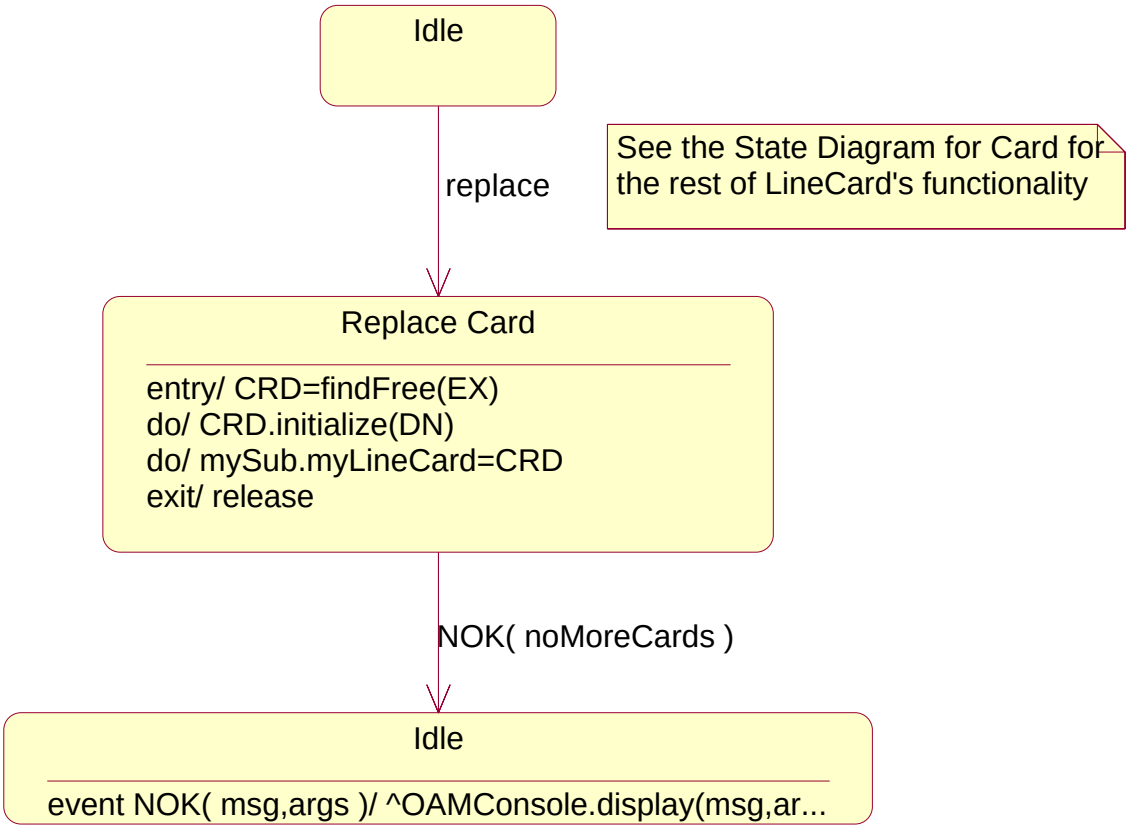

Figure 33: Line Card State Diagram

The Line Card State Diagram shows the functionality of the replace method on the class. Other functionality for this class is shown in the Card class which this class extends from. When a replace event occurs, another line card is allocated and it is initialized to the current DN of the card being replaced. Then the subscription which is associated with the card to be replaced is reassociated to the new line card that has been allocated. The card to be replaced is then released.

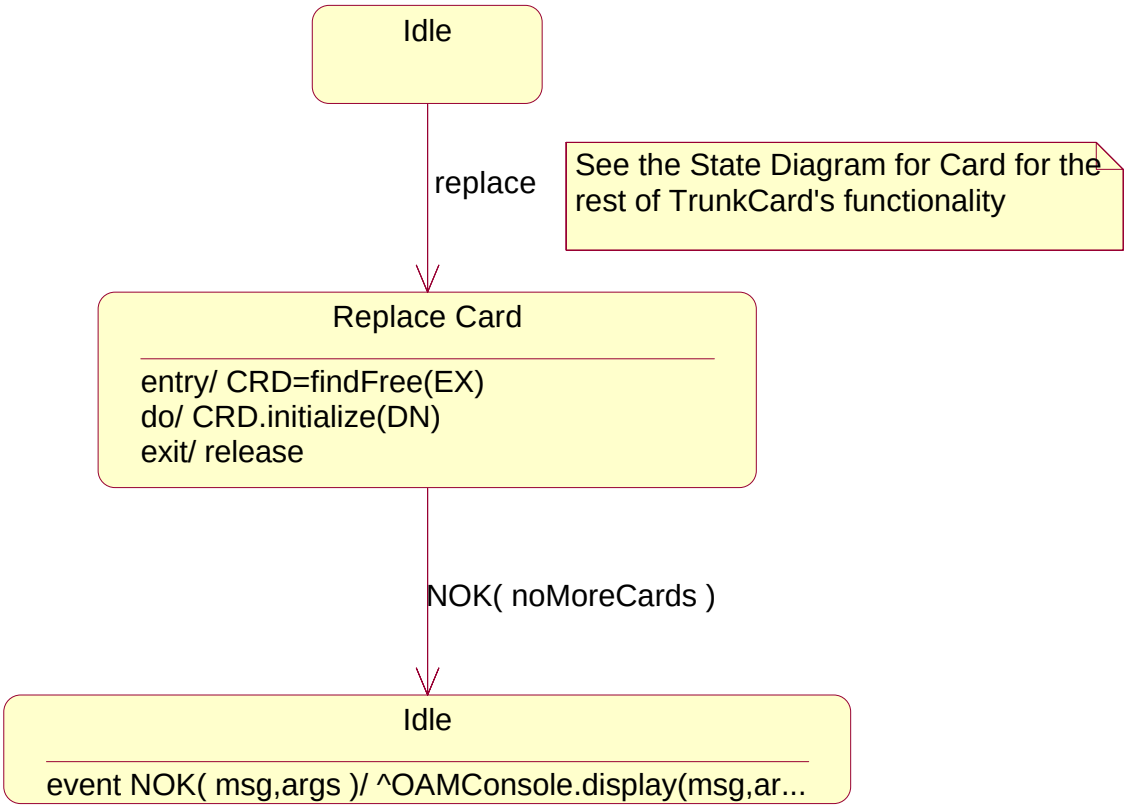

Figure 34:Trunk Card State Diagram

The Trunk Card State Diagram shows the functionality of the replace method on the class. Other functionality for this class is shown in the Card class which this class extends from. When a replace event occurs, another trunk card is allocated and it is initialized to the current DN of the card being replaced. Then trunk card to be replaced and is released.

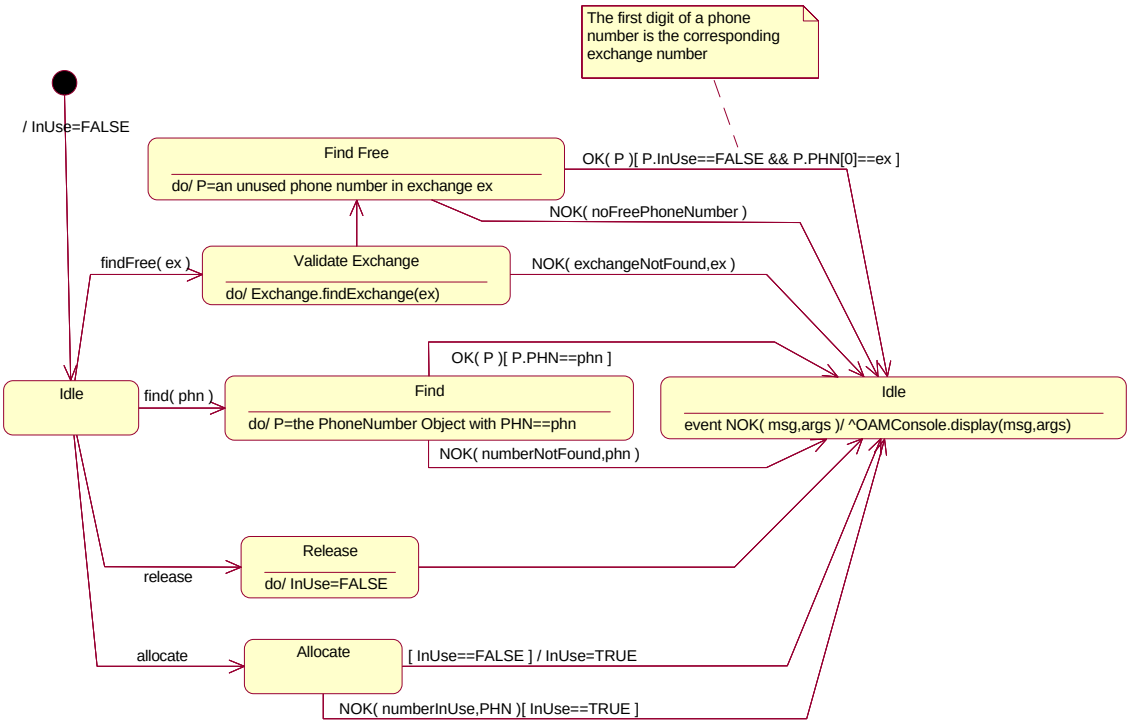

Figure 35:Phone Number State Diagram

The Phone Number State Diagram shows four events: allocate, deallocate, findFree, and find. The diagram begins in the idle state and transition out of the state on any of the four events. The allocate event causes a transition into the Allocate State which will set the inUse attribute to true if it is not in use, otherwise it will return a NOK, stating that the phone number is already in use. It will then return to the idle state. The deallocate transition simply sets the inUse attribute to false and transitions into the idle state. When a find event is received, the current state becomes Find. A PhoneNumber is returned if and only if the phoneNum of the PhoneNumber found matches the phoneNum specified, otherwise NOK is returned. The findFree event causes a transition into Validate Exchange, in which the ex: ExchNum is check to see if it is valid. If not an error NOK is returned stating that the exchange is invalid, otherwise a transition into Find Free occurs. In this state a free phone number is found if one exists in the exchange specified, if not an error is returned stating that there are no free phone numbers in the exchange. Control is then returned to the idle state.

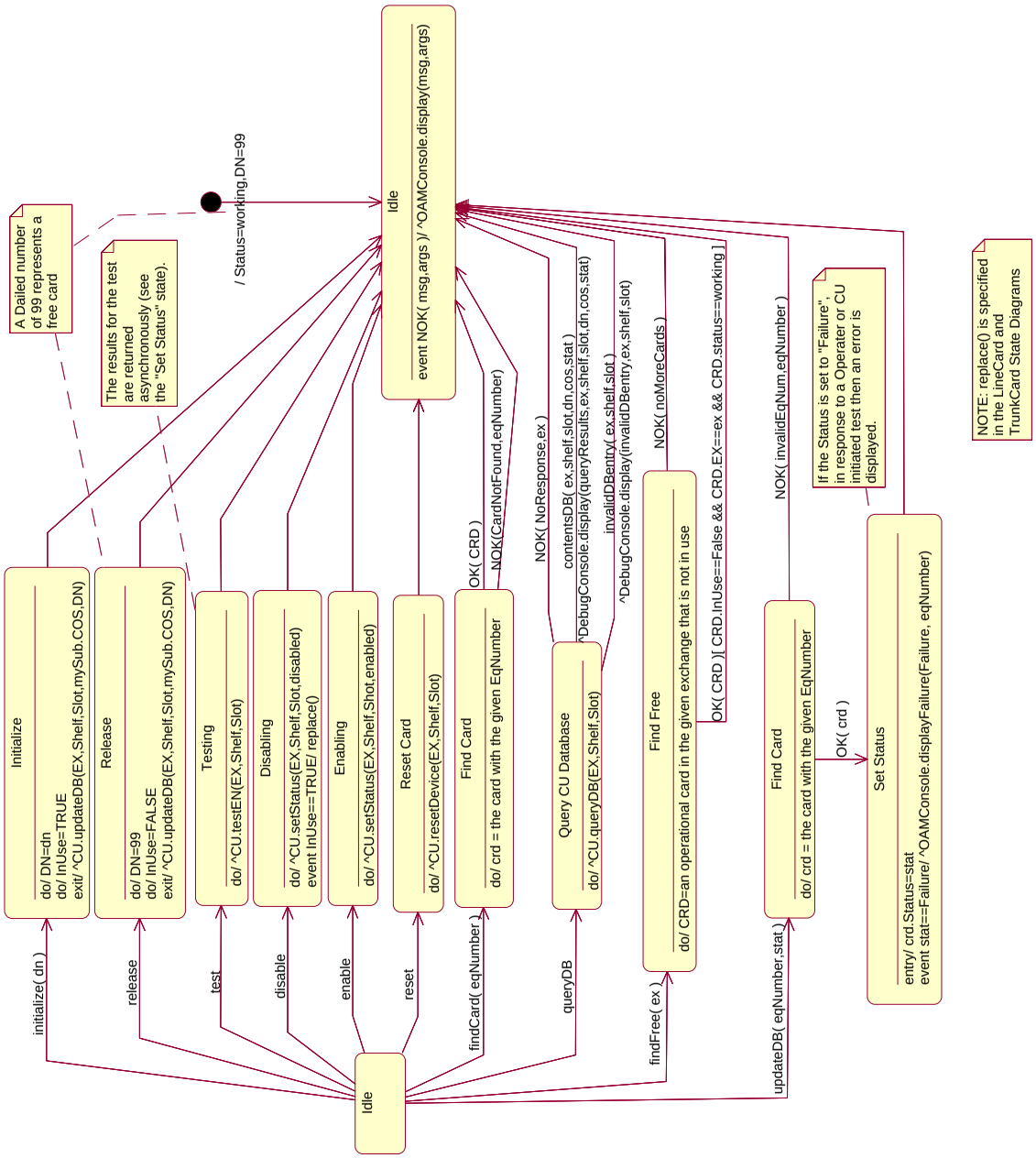

Figure 36:Card State Diagram

The Card State Diagram has 10 input events. When an initialize event is received, the dialed number is set to the DN provided and the CU is sent an UpdateDB message, then the current state goes back to the idle state. [13] When a release event occurs a transition to Release occurs. In this state the dialed number is set to 99 which represents that the line card has not been allocated, and the CU is told by an updateDB message. [13] When a test event occurs a transition occurs into the Testing state which sends a testEN message to the CU, the state then goes back to the idle state. If a hardware device fails the test, it will send a updateDB message to this object which it will process. [47, 12] If a disable event occurs, the current state is transitioned into the Disabling state, in which a setStatus (sends an updateDB message) message is sent to the CU setting the card as disabled.

[7] If an enable event occurs, a transition into Enabling State is taken. In this state, the card's status is updated on the CU. This state is much like the disable, but it sets the status to true. [7] The reset event causes a transition to be taken into Resetting. The Resetting state will cause a message to be sent to the CU to reset the specified device. If a find event is received then a transition is taken into Find. The Find activity finds a card given a specific equipment number. It can either return NOK stating that the card could not be found or OK, returning the Card that matches the EquipNum. When the operator wishes to do debugging, the QueryDB messages can be sent to the card, these events are synchronous events to the CU. [43, 46] When the QueryDB event is received a transition into Query CU Database occurs. Upon entry to this state, a message is sent to the CU requesting debugging of the card. It remains in this state until either a ContentsDB, InvalidDBEntry or a timeout occurs. [44, 45] The invalidDBEntry and ContentsDB write the output received to the DebugConsole. If a timeout occurs an error is returned if the form of a NOK. The findFree event causes a transition to be taken to the Find Free State. This state finds a free card in the exchange and returns OK if a card is found, otherwise it returns NOK. The UpdateDB event which is sent by the CU causes a transition into the FindCard activity. The associated card is located, then the status on the card is updated as the CU requested. If the status bits on the update indicate a hardware failure then the Operator will be notified of the error.

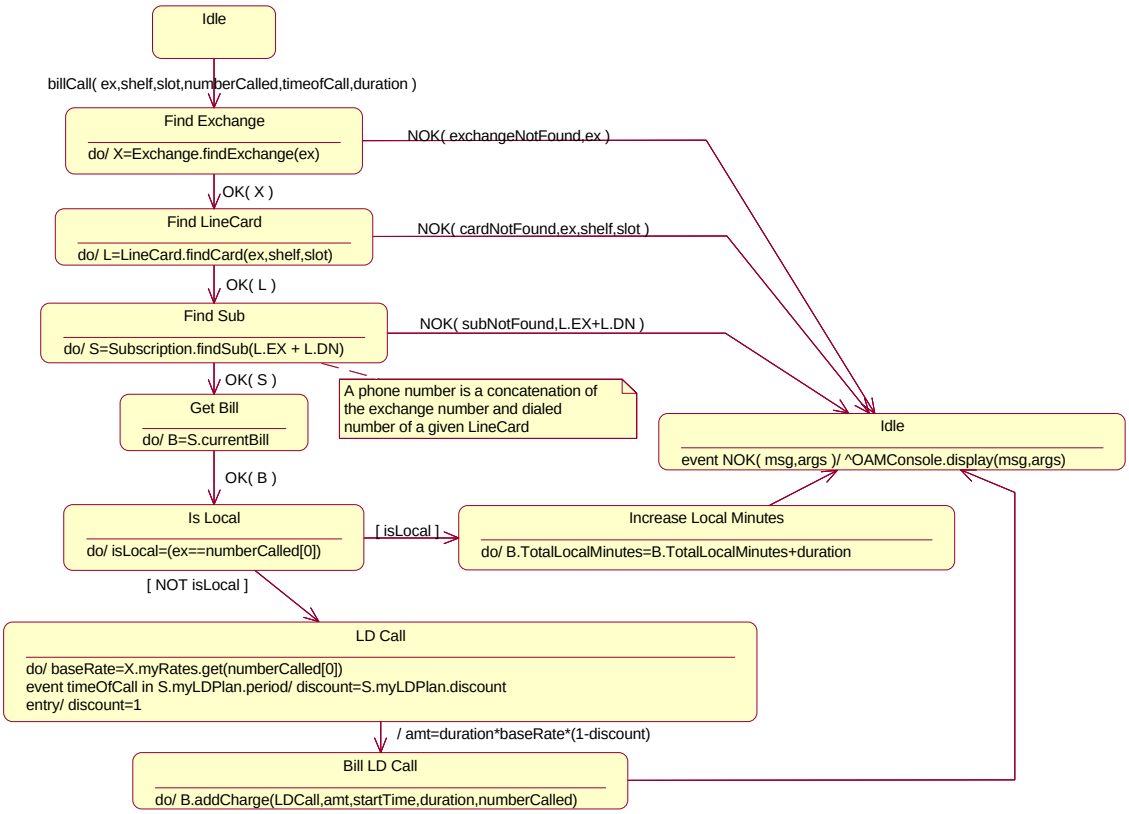

Figure 37:Exchange -Bill Call State Diagram

The Exchange - Bill Call State Diagram is one in a sequence of three state diagrams for Exchange. The Bill Call follows the following pattern. The bill call event is received, and the first transition is into Find Exchange, where the exchange is found. If it is not found then a NOK is returned (and back to the idle state) otherwise it transitions into Find LineCard state. In this state the line card is either found or not. If it is not a NOK is returned (and back to the idle state), otherwise the transition occurs to Find Subscription. The subscription is located, if it is not found a NOK is returned, otherwise it transitions into GetBill. The GetBill state retrieves the current bill from the subscription, then transition to the state IsLocal is taken. The phone number called is determined if it is in the local exchange. If it is in the local exchanged then the transition to IncrementLocalMinutes is taken, otherwise it is a LD Call and the transition to LDCall is taken. The IncrementLocalMinutes state takes the current bill in increments the number of total minutes by the duration of the call. It then returns to the idle state. The LDCall state finds the current LD discount plan and finds the billing amounts for the exchange called. Once these items are found, the Bill LD Call creates a charge on the current bill of the amount calculated; control then returns to the idle state.

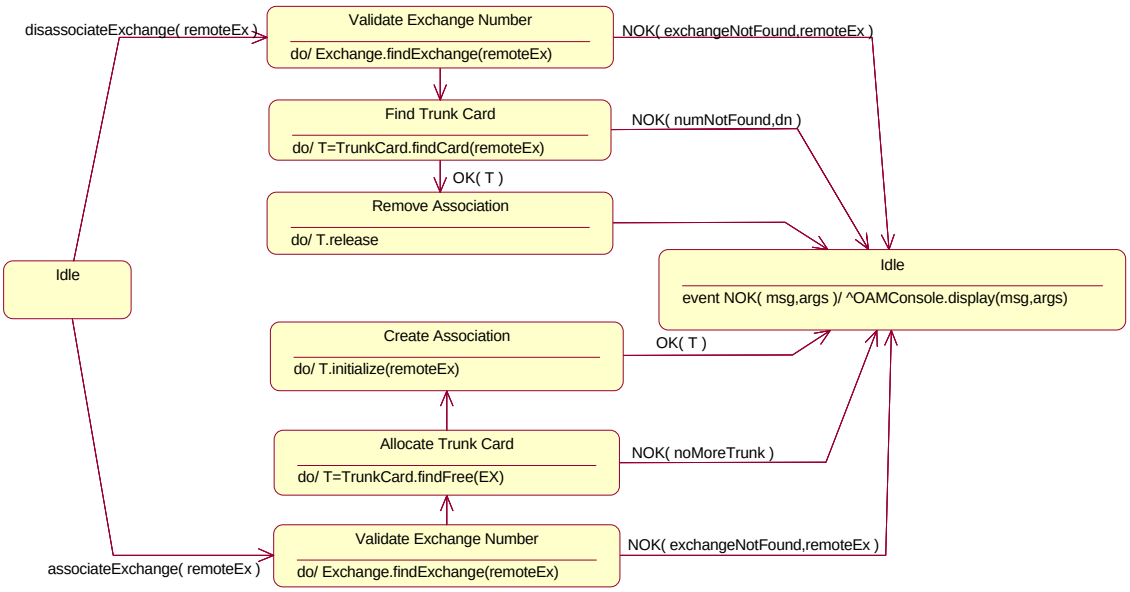

Figure 38:Exchange -Exchange Associations State Diagram

The Exchange - Exchange Associations State Diagram shows the process of associating one exchange with by the use of a trunk card. There are two input events to this state diagram; associateExchange, and disassociateExchange. The associateExchange event causes a transition to be taken from the idle state into the Validate Exchange state. The remote exchange is validated, if it is not a valid exchange a NOK is returned to the caller, otherwise a transition to AllocateTrunkCard is taken. In AllocateTrunkCard, an attempt to acquire a trunk card is done, if unsuccessful the state returns NOK and proceeds back to the idle state. If the attempt was successful, a transition to Create Association is taken. In Create Association, the trunk card's DN is set to the number of the remote exchange, then control is returned back to the idle state. The disassociateExchange event causes a transition to be taken from the idle state into the Validate Exchange state. the remote exchange is validated, if it is not a valid exchange a NOK is returned to the caller, otherwise a transition to FindTrunkCard is taken. In FindTrunkCard, a trunk card is searched for that has a DN matching the remote exchange number. If one is found, a transition into RemoveAssociation is taken, otherwise a NOK is returned to the caller. In RemoveAssociation the trunk card is released and control is returned to the idle state.

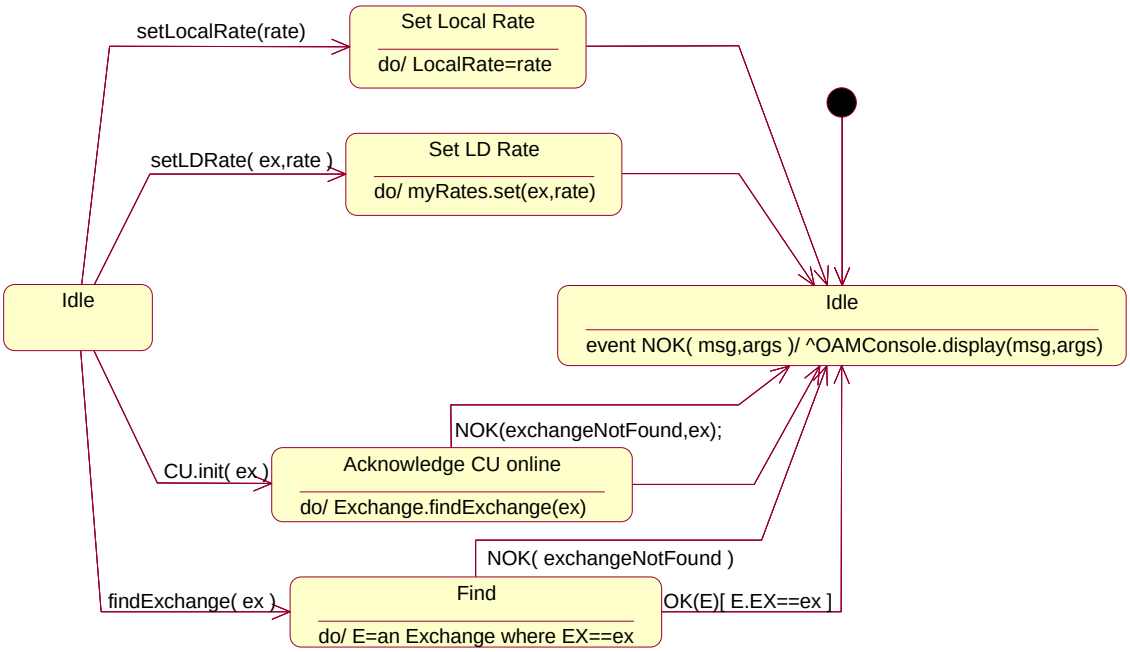

Figure 39:Exchange -Misc. State Diagram

The Exchange - Misc. State Diagram shows the four input events; setLocalRate, setLDRate, CU.Init and findExchange. Upon receiving a setLocalRate event the control transitions from the idle state to the Set Local Rate state. In the Set Local Rate state the local rate attribute is set to the value specified, then control is returned to the idle state. If the setLDRate event is received, the transition to the Set LD Rate is taken. In the Set LD Rate state the rate corresponding to the exchange provided is updated with the new rate specified. Once the new rate has been set, a transition back to the idle state occurs. If a CU.Init event is received a transition goes to the Acknowledge CU online state. In this state the exchange number provided is validated by trying to find the exchange in the available exchanges. If the exchange is not found then a NOK is returned and outputed to the OAMConsole. If a findExchange event is received a transition into the Find state occurs. In Find an exchange is searched for that matches the exchange number specified. If one is not found the NOK event is returned, otherwise OK is returned with the Exchange that was found.

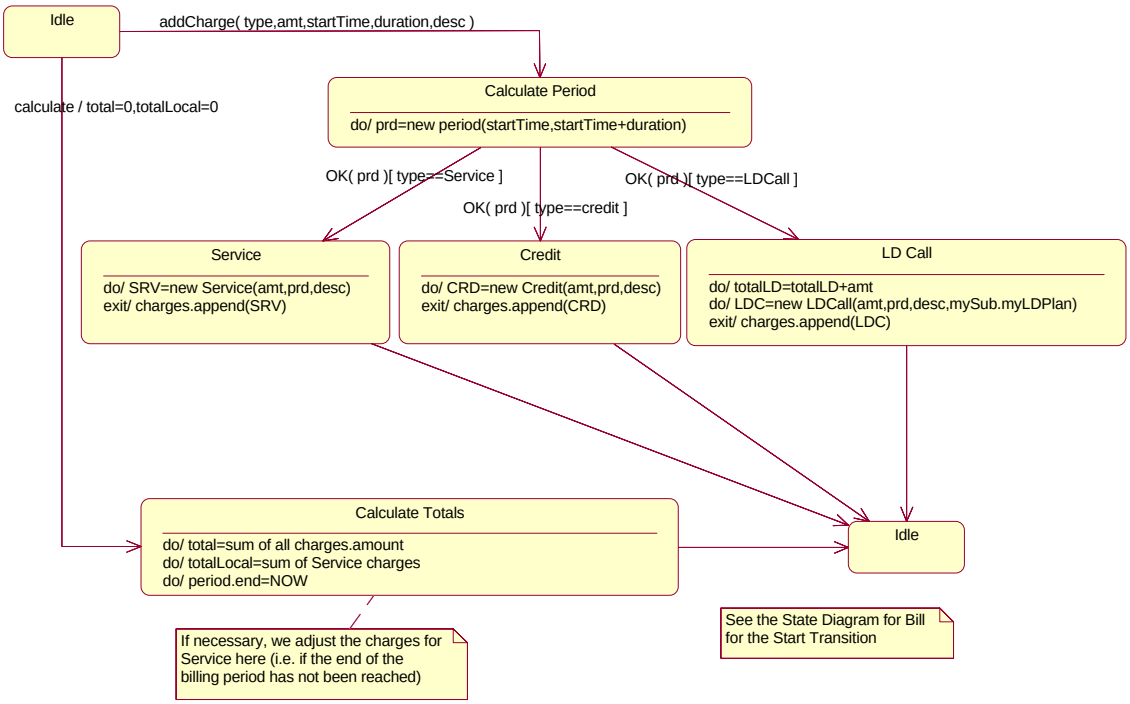

Figure 40:Current Bill State Diagram

The Current Bill State Diagram has two input events, addCharge and calculate. If the event received is a calculate event, the totals and a transition to Calculate Totals is taken. In Calculate Totals, all charges corresponding to the bill are summed. The attributes corresponding to the totals are also set, and then control is returned to the idle state. If the event received is an addCharge event, then a transition into Calculate Period is taken. In Calculate Period, a new TimePeriod is generated based on the input specified, once this is done, the type of charge determines the next transition. If a Service was to be added, the transition to Service is taken. In Service, a new Service is created and its parameters set, then the charge is appended to the charges on the bill. If a Credit was to be added, the transition to Credit is taken. In Credit, a new Credit is created and its parameters set, then the credit is appended to the charges on the bill. If a LDCall was to be added, the transition to LDCall is taken. In LDCall, the total number of LD is incremented, a LDCall is created and assigned its values by the parameters, then the charge is appended to the charges on the bill.

The following diagrams depict various scenarios that were determined to be important and require explicit graphical representation.

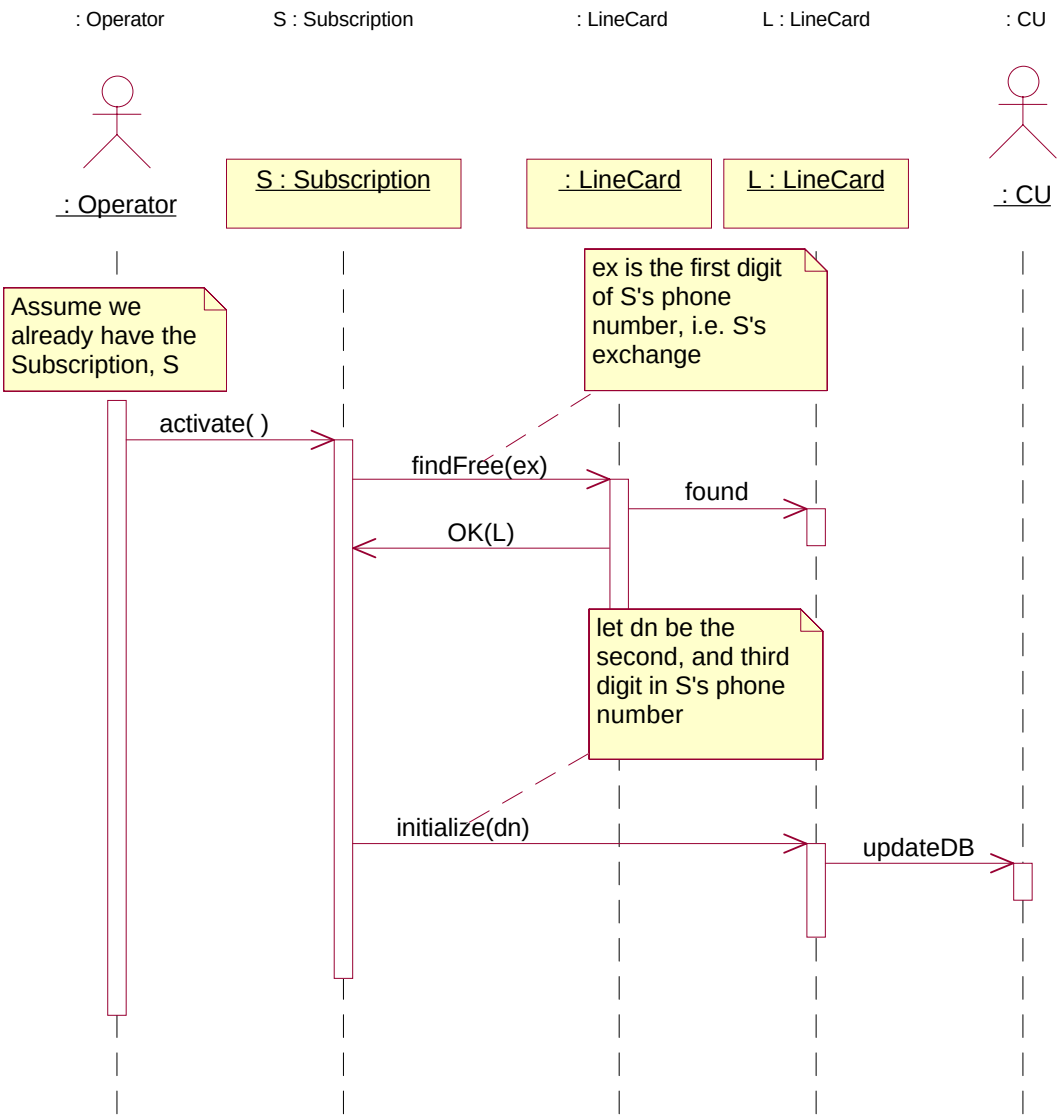

Figure 41:Sequence Diagram - Activating a Subscription

The Activating a Subscription Sequence Diagram traces the successful activation of a subscription to an existing subscription. This is accomplished by, finding a free line card in the associated exchange, initializing the line card with the dialed number. To begin, the Operator requests from an activate on an existing subscription. The subscription then tries to find a free line card in the exchange that the subscription is associated with. Note that the subscription should already have a PhoneNumber acquired in the exchange, so the subscription is associated with an exchange. A line card, L is found and returned to the subscription S, where S then initializes L to the DN (last 2 digits of associated PhoneNumber). Once this initialization of the line card takes place, an UpdateDB message must be sent to the CU to signal that the physical hardware line card must be updated.

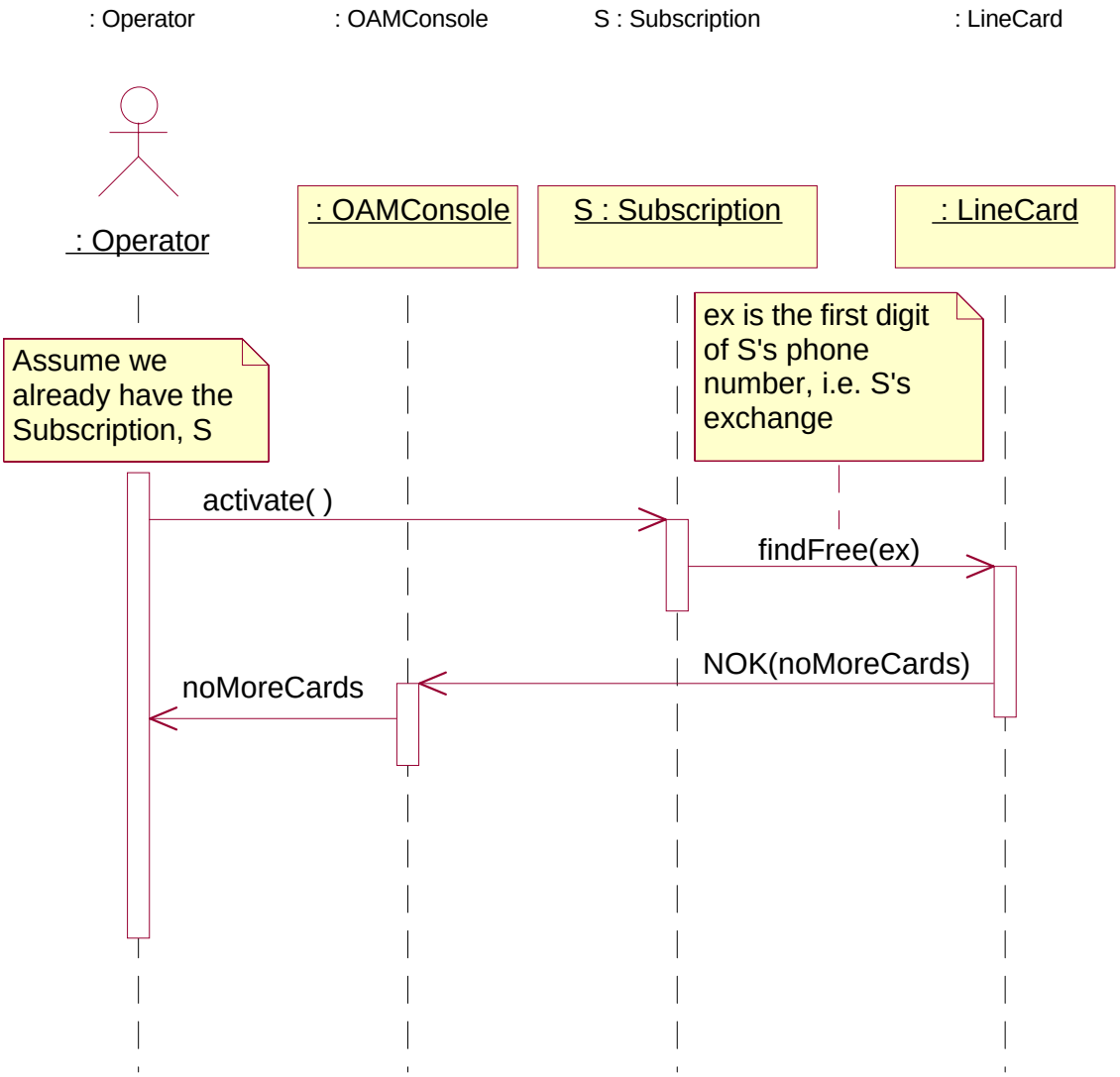

Figure 42: Sequence Diagram - Activating a Subscription Error

The Activating Subscription Error Sequence Diagram depicts the situation in which an error is generated when trying to activate a subscription. In this case the error is generated by the fact that there are no more cards in the exchange. The sequence is as follows: the Operator initially requests activate on an existing subscription S. S then queries for a free line card in the exchange specified by the PhoneNumber of the subscription S. The LineCard static method returns a NOK, stating that there are no free line cards and one cannot be acquired. A message is displayed on the OAM-Console stating that there are no free cards and the operation of activating a subscription has failed.

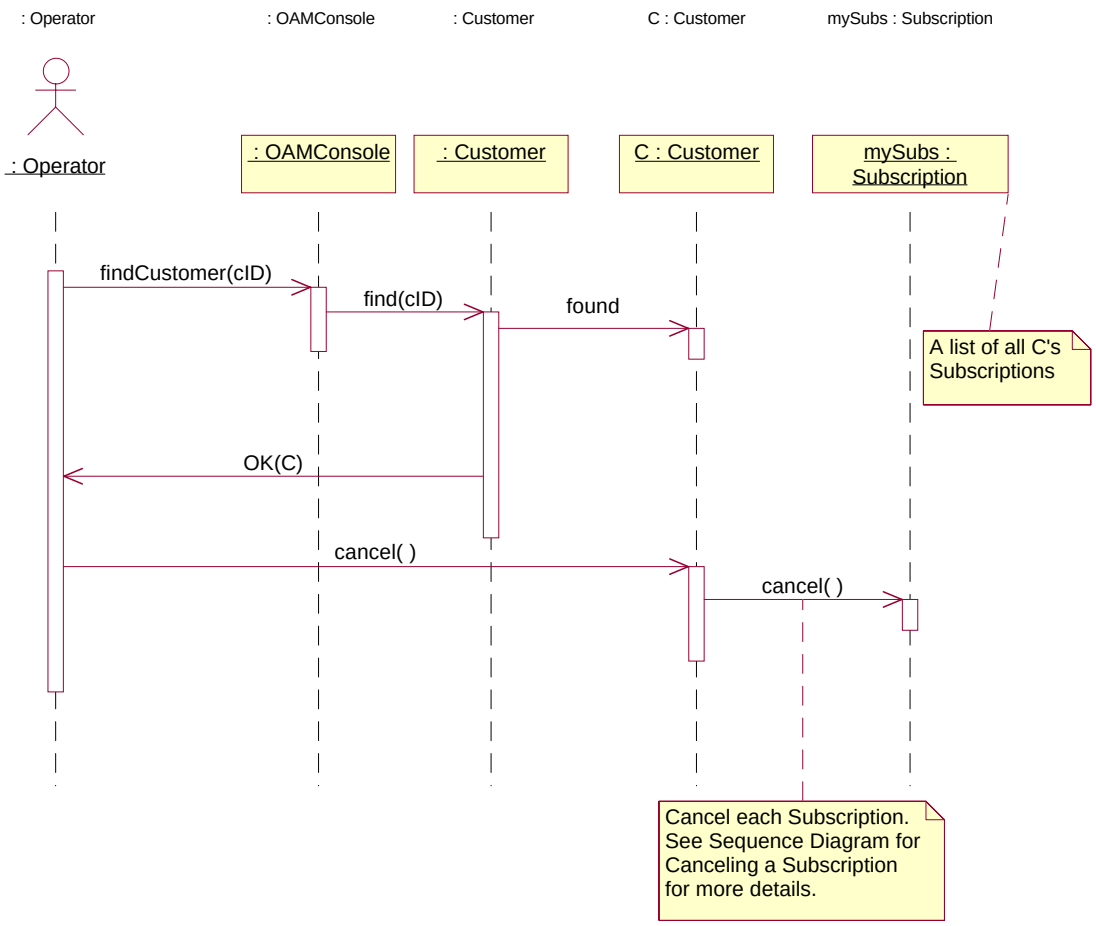

Figure 43: Sequence Diagram - Cancel Customer

The Cancel Customer Sequence Diagram traces the events of cancelling an existing customer's account. The cancel of an account will cause all subscriptions associated with this customer to be cancelled aswell. The sequence diagram depicts the action of finding the customer account using a customer identifier. The Operator sends a message to the OAMConsole to find a Customer given an identifier. The OAMConsole then passes this message onto the Customer class which then searches for such a Customer. The Customer is found and returned successfully back to the Operator. The Operator then requests a cancel on the Customer which then causes a cancel to occur on each of the subscriptions that the Customer is associated with. See the Cancel Subscription for information on cancelling individual subscriptions.

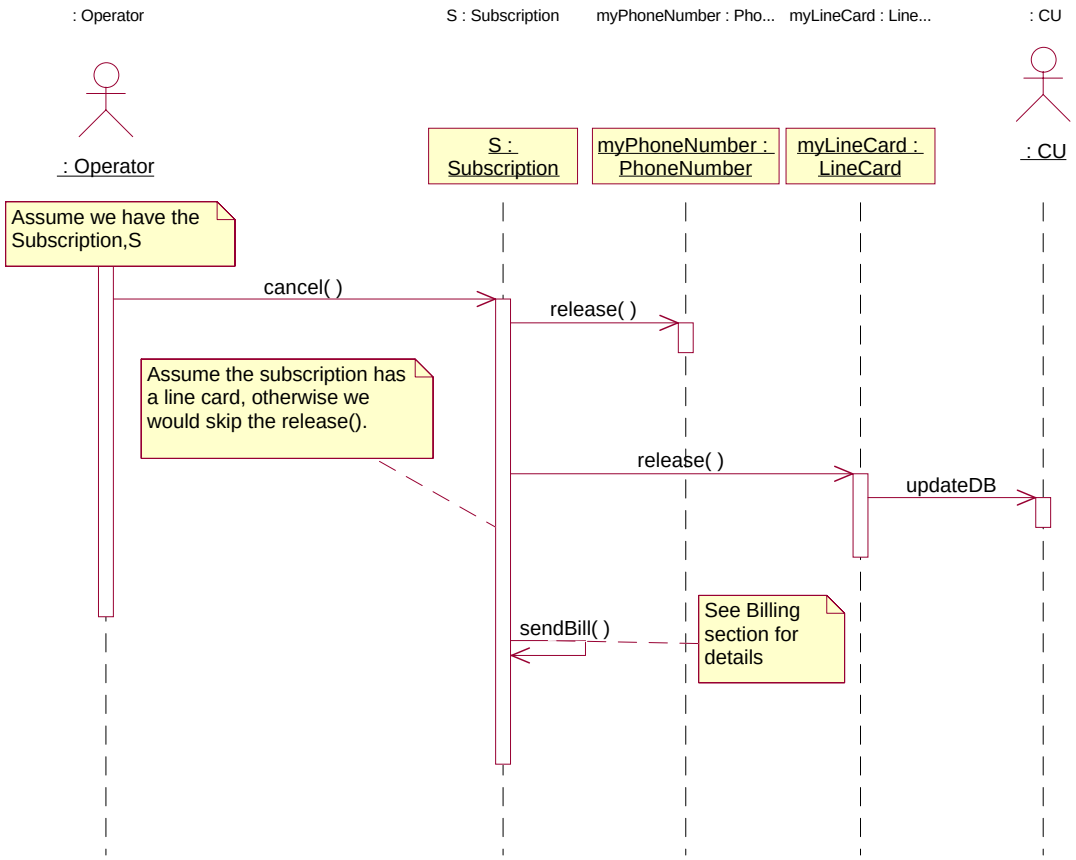

Figure 44: Sequence Diagram - Cancel Subscription

The Cancel Subscription Sequence Diagram illustrates the cancel on a subscription. There are various modifications of this sequence diagram since this one depicts the Operator cancelling the subscription directly. However, as seen in the Cancel Customer Sequence Diagram a cancel subscription can originate from a Customer rather than an Operator, but the functionality is essentially the same. The sequence is as follows: the cancel event is sent by the Operator to the existing Subscription. The subscription then releases the associated PhoneNumber and LineCard. The release on the LineCard causes an UpdateDB message to be sent to the CU stating that the card is now not allocated. The subscription then requests itself to send a bill. (The send bill operation on a subscription will be depicted later)

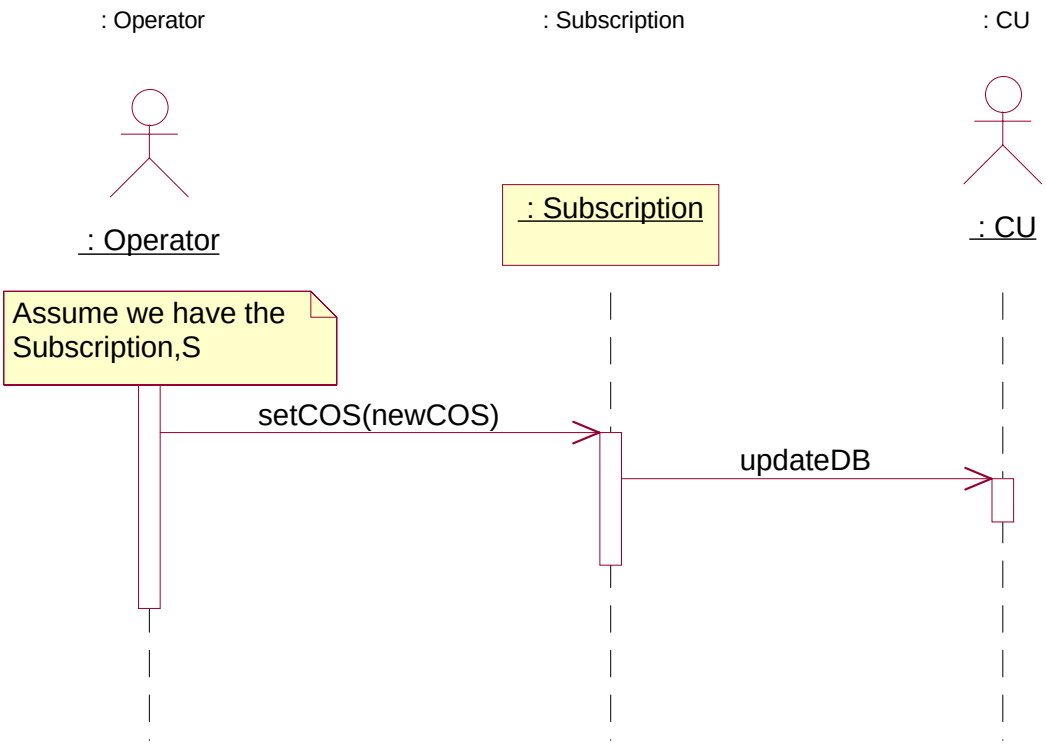

Figure 45: Sequence Diagram - Change Class of Service

The Change Class of Service Sequence Diagram traces the sequences involved in setting the class of service on a subscription. The Operator requests of an existing subscription to set the COS. The set of the COS causes an UpdateDB message to be sent to the CU, stating that the properties for the line card associated with the subscription have changed.

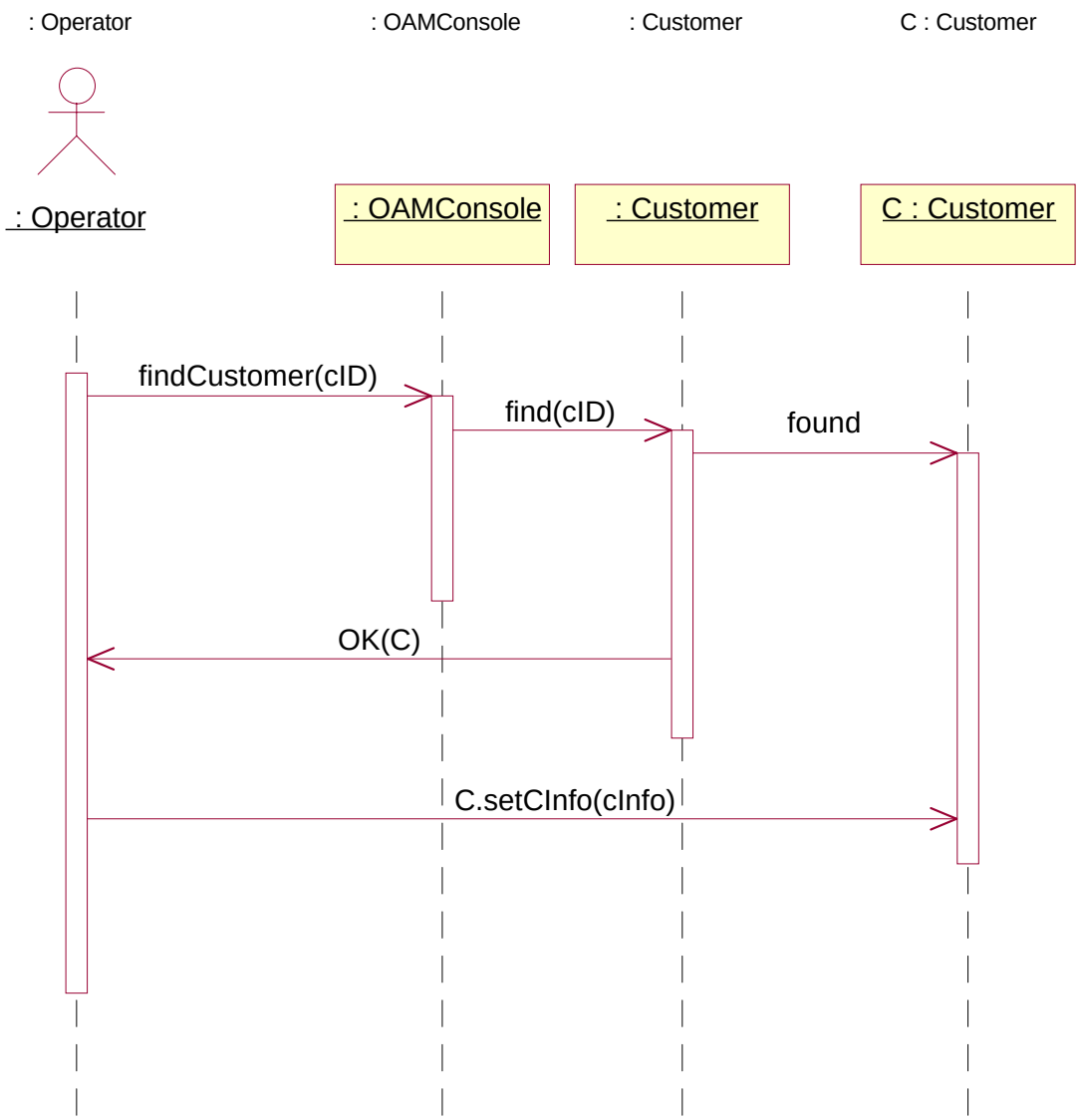

Figure 46: Sequence Diagram - Change Customer Info

The Change Customer Info Sequence Diagram illustrates the sequence of events for changing a customer's information. The sequence begins by the Operator requesting a find of a customer by identifier, this is sent to the OAMConsole. The OAMConsole then sends the message on to the Customer class which finds the Customer C and returns it to the Operator. The Operator then sends the event setInfo on the Customer C that was found. This results in the Customer's information being updated by the Operator.

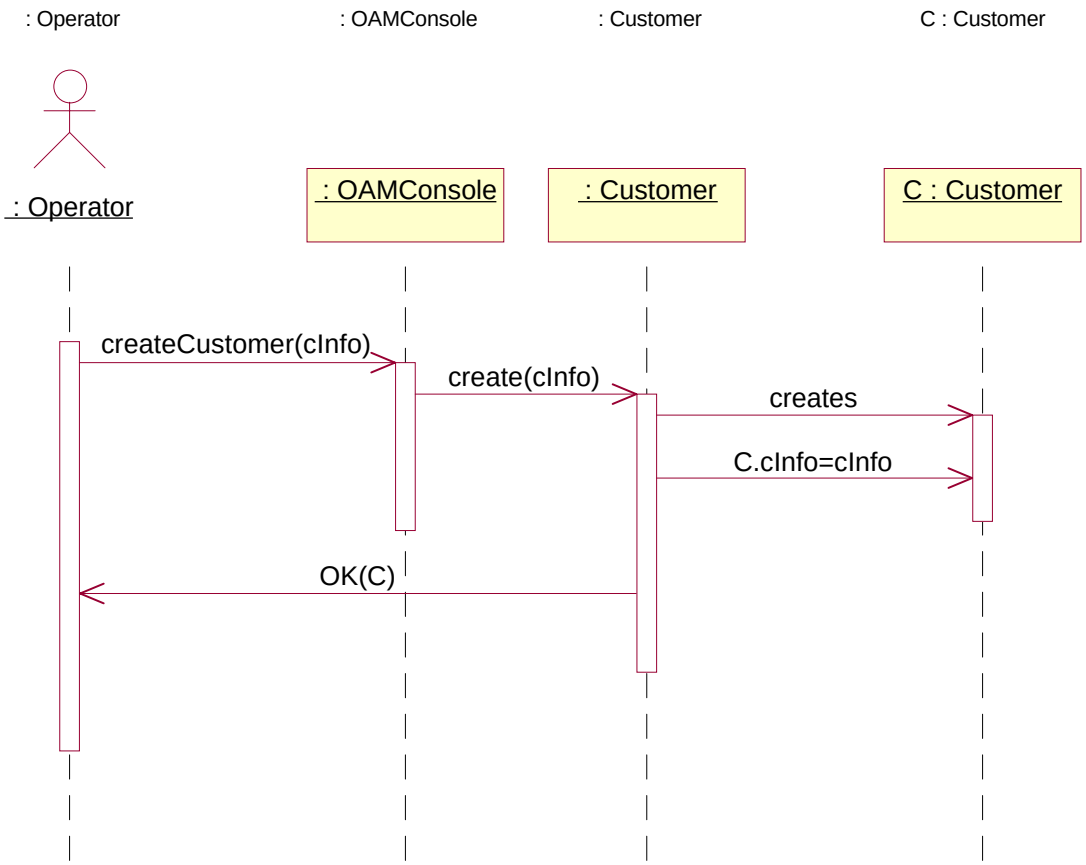

Figure 47: Sequence Diagram - Create Customer

The Create Customer Sequence Diagram traces the steps required to complete a successful creation of a customer. First, the Operator sends a createCustomer event to the OAMConsole with the customer information as a parameter. The OAMConsole then sends a create event to the Customer class. This customer class then creates a new Customer C, and sets the customer information on C with the data that was specified. The Customer class then returns this Customer back to the Operator, successfully completing the sequence.

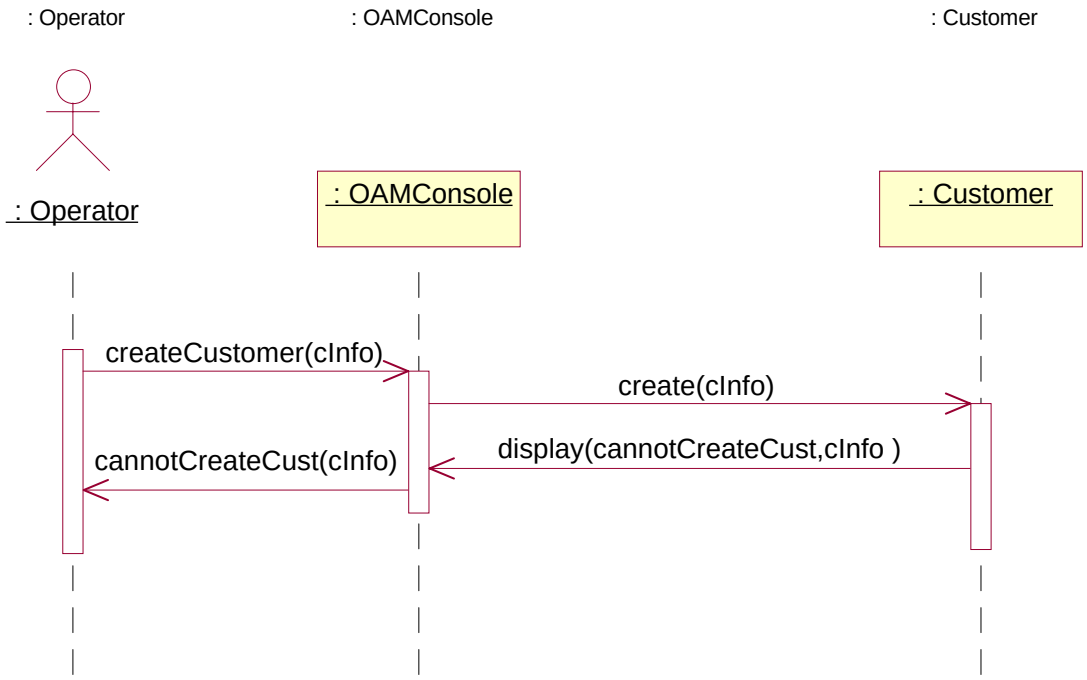

Figure 48: Sequence Diagram - Create Customer Error

The Create Customer Error Sequence Diagram depicts the erroneous situation in which the Customer cannot be created. The failure could be for a number of reasons including that the customer information supplied was invalid or incomplete. The sequence is relatively simple: the Operator sends a createCustomer event to the OAMConsole. The OAMConsole then redirects this message to the Customer Class. This Customer Class then determines that it cannot create the Customer requested. The error is returned to the OAMConsole stating that the Customer could not be created. The OAMConsole then informs the Operator that the Customer could not be created.

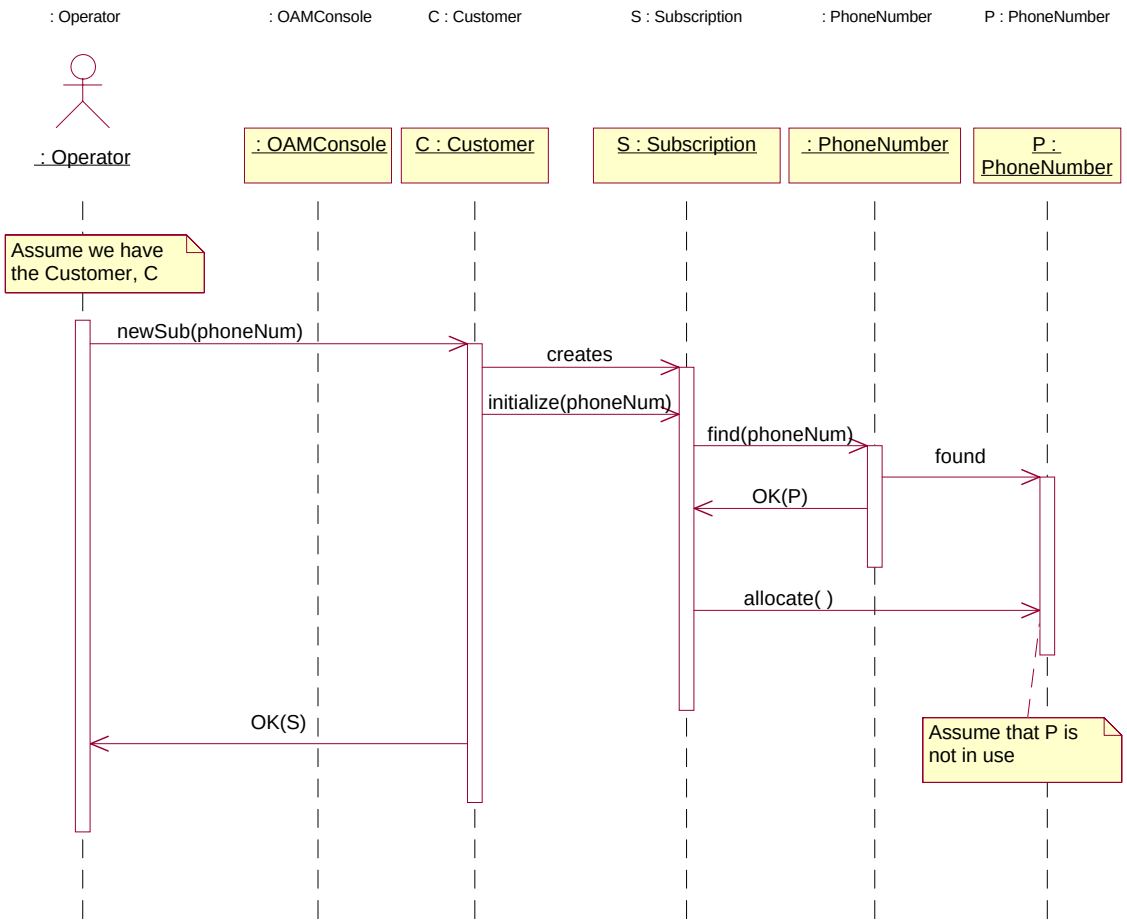

Figure 49: Sequence Diagram - Create Sub with Specific Number

The Create Subscription with Specific Number Sequence Diagram shows the creation of subscription with a phone number that the customer requests. The Operator sends a newSub event with the phone number requested specified to the existing customer. The customer would have been previously found by one of the find customer events. (See sequence diagrams for more information) The customer, C then creates a new subscription and initializes it with the phone number specified. This initialization causes the newly created subscription to find the phone number specified and attempt to allocate it. It does so successfully in the case depicted here, the PhoneNumber returns the PhoneNumber found and the allocate on P is successful. The Customer class then returns to the Operator that the creation of the subscription was successful.

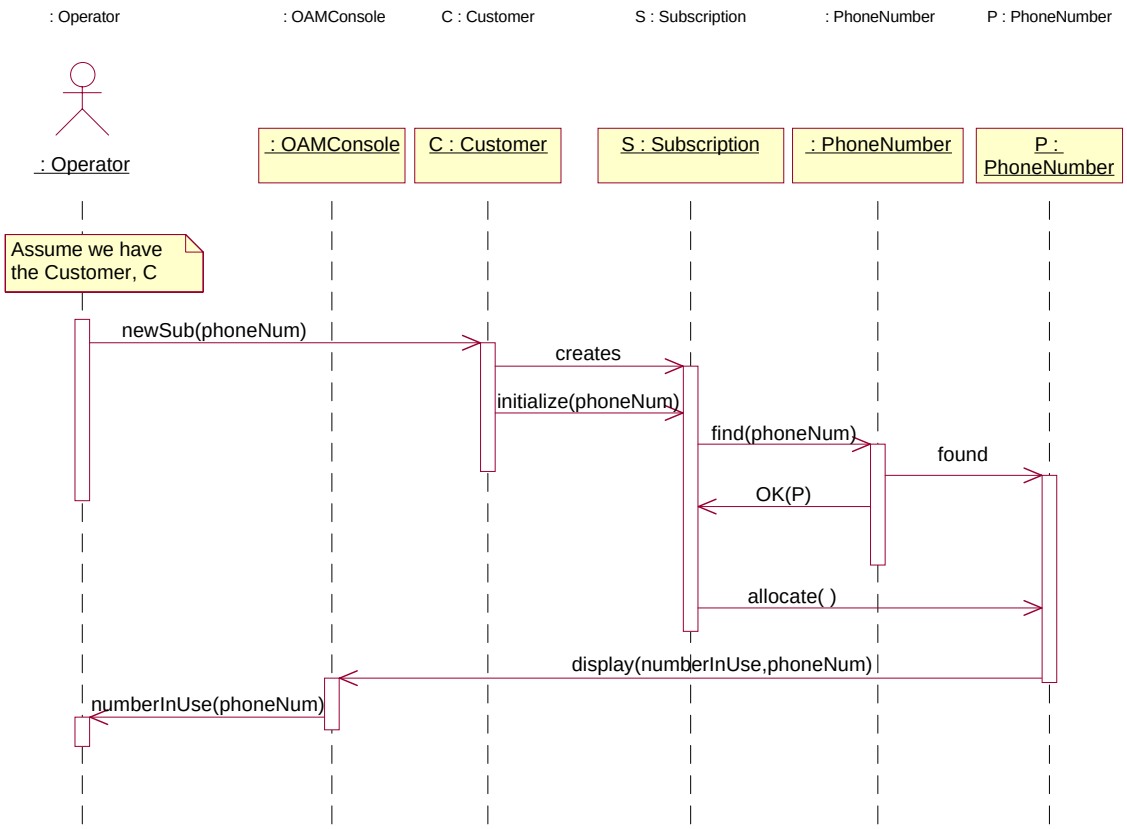

Figure 50: Sequence Diagram - Create Sub with Number error

The Create Subscription with Number error Sequence Diagram shows the erroneous case of the Create Subscription with Number. The error is raised because the phone number requested is already in use. Essentially the same as Create Sub with Number, the Operator requests a newSub specifying the phone number. The existing customer then creates a new subscription and initializes it with the phone number. The initialize on the subscription causes the subscription to attempt to find and allocate the PhoneNumber. However, this attempt fails, the PhoneNumber is found but the number is in use. The PhoneNumber then returns an error to the OAMConsole stating that the PhoneNumber is already in use. The OAMConsole displays this error, returning to the Operator that the phone number is already allocated.

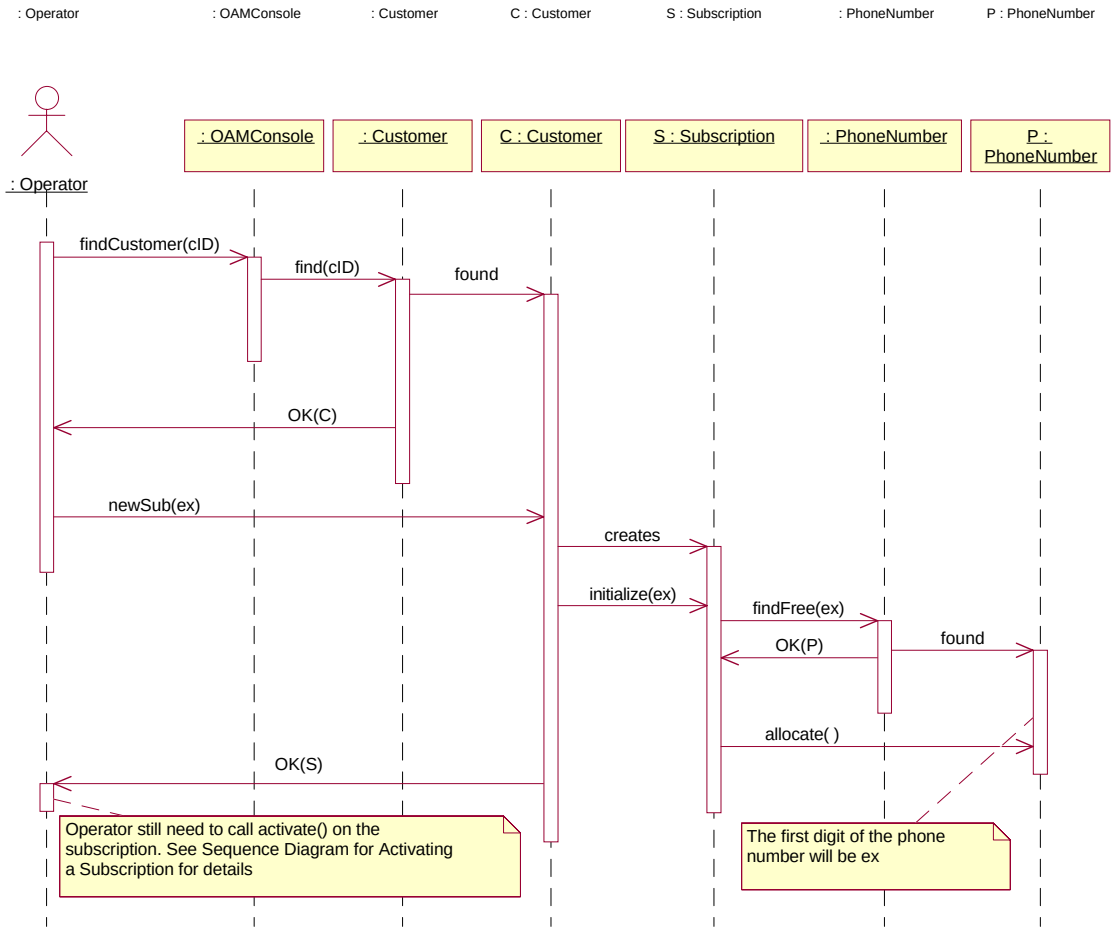

Figure 51: Sequence Diagram - Create Subscription

The Create Subscription Sequence Diagram depicts the creation of a subscription with an unspecified phone number, only an exchange is specified. The sequence begins with finding the customer given a customer identifier; this was shown in previous diagrams and is very similar to those. The Customer is found and returned to the Operator. The Operator then issues a newSub event on the Customer specifying the exchange in which the new subscription is to be allocated in. The Customer then creates a subscription, S and initializes it with the exchange specified. The initialize event on the subscription causes the subscription to find a free phone number in the exchange. A free PhoneNumber, P is found and returned to the subscription. The subscription then attempts to allocate this PhoneNumber and is successful. The Subscription then returns an OK to the Operator stating that the Subscription was created with success.

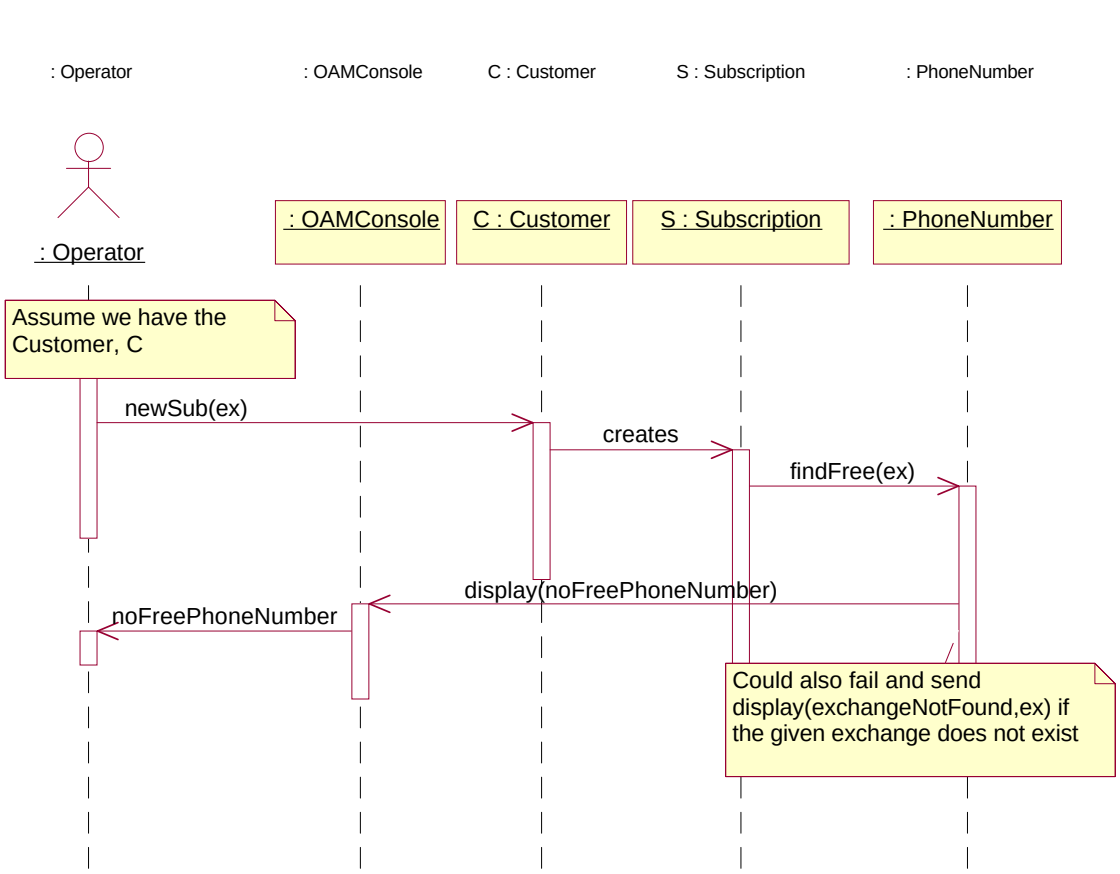

Figure 52: Sequence Diagram - Create Subscription Error

The Create Subscription Error Sequence Diagram illustrates the an error condition that are generated when an Operator requests a phone number in an exchange. The Operator sends a newSub event to an existing Customer. This Customer class then creates a subscription and attempts to initialize it with the exchange provided. The subscription can return an error to the OAMConsole stating that the exchange is invalid, but in this case it does not. (ie. the exchange is valid). The subscription continues by trying to find a free phone number in the exchange. The PhoneNumber returns an error to the OAMConsole stating that there are no free numbers in the exchange and the request has failed. The error is then displayed to the Operator by sending the event noNumberFree.

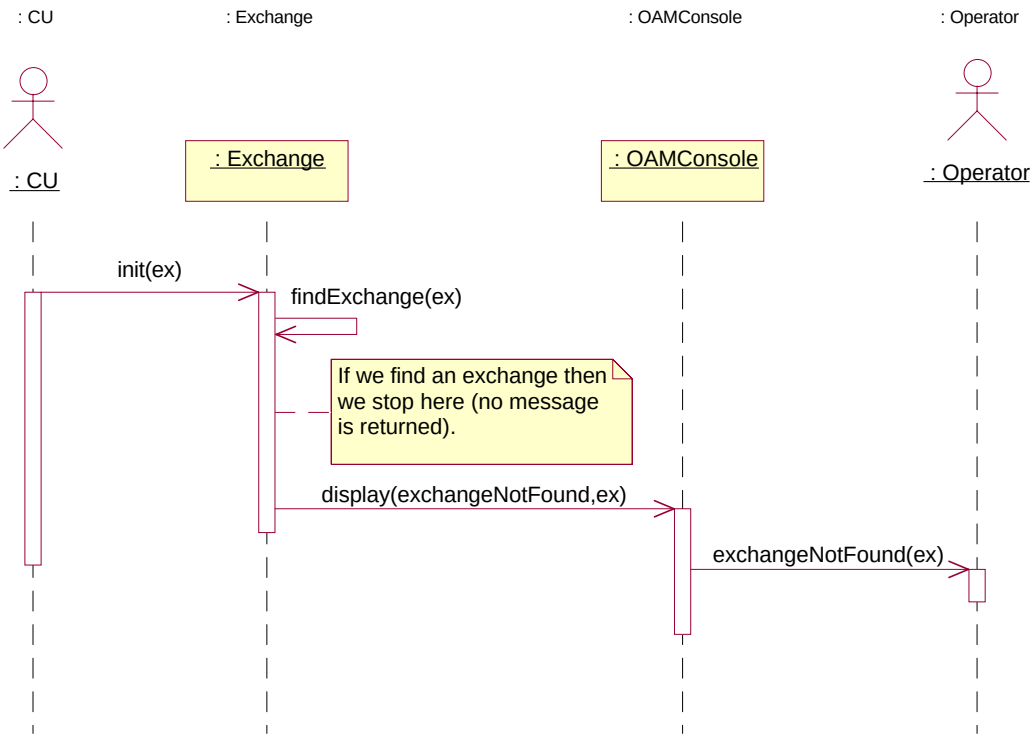

Figure 53: Sequence Diagram - CU Initialization Error

The CU Initialization Error Sequence Diagram traces the path when a CU comes online. In this sequence diagram we depict an error occurring that the exchange is not a valid exchange that the CU reported when it started up. The case in which there is no error, the exchange would just be checked and if no error, it would just acknowledge the request. The sequence is as follows: upon start up of the CU it broadcasts a init signal to the OAM stating that it is alive an what its exchange number is. The Exchange class handles this request and checks to see if it is a valid exchange. If it is not then an error is written to the OAMConsole stating that a CU has come online but it is not a correct exchange. This message is then displayed to the Operator stating that the exchange was not found on an init of a CU.

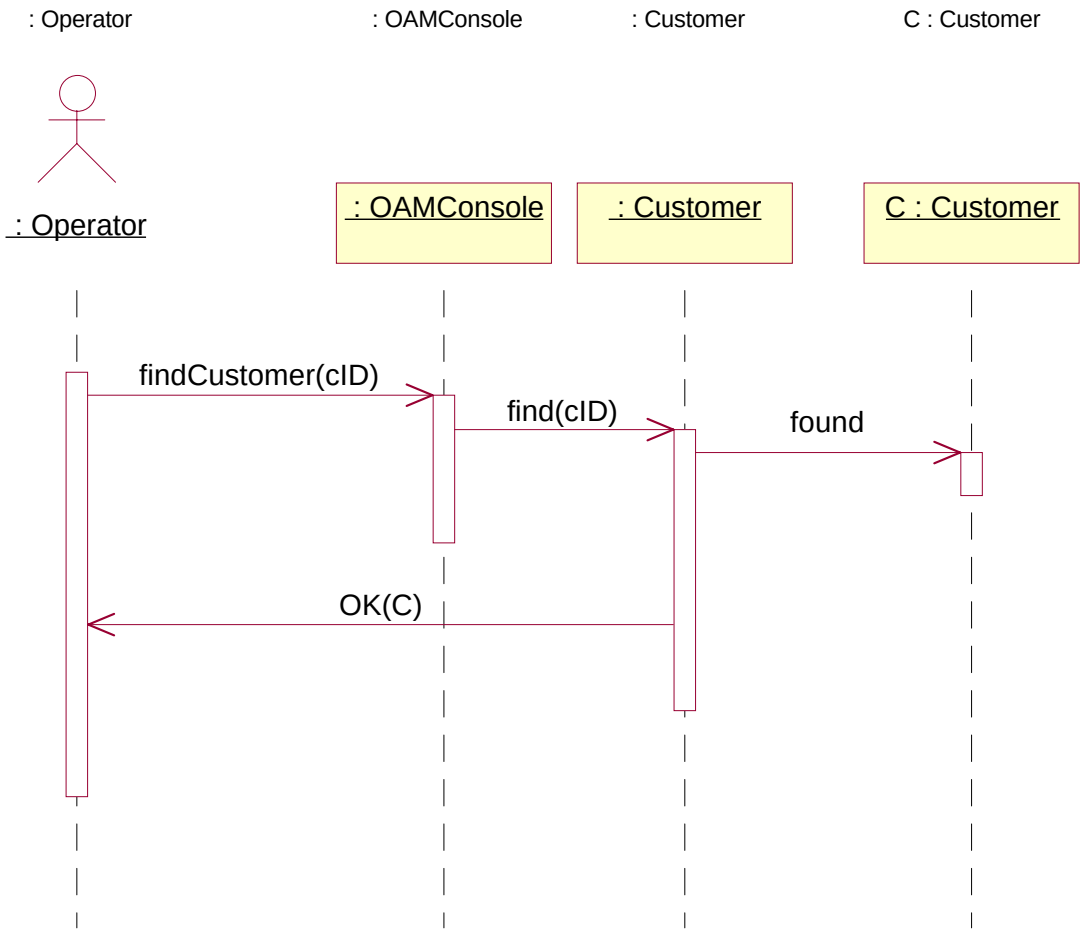

Figure 54: Sequence Diagram - Find Customer By cID

The Find Customer by cID Sequence Diagram traces the sequence of the Operator finding a customer by unique identifier. The sequence begins by the Operator sending a findCustomer event with a specified cID. The OAMConsole receives this event and passes it onto the Customer through a find event. The Customer class then searches the Customers for a Customer that matches the cID. In this case the Customer is found and a OK with the Customer found is returned to the Operator.

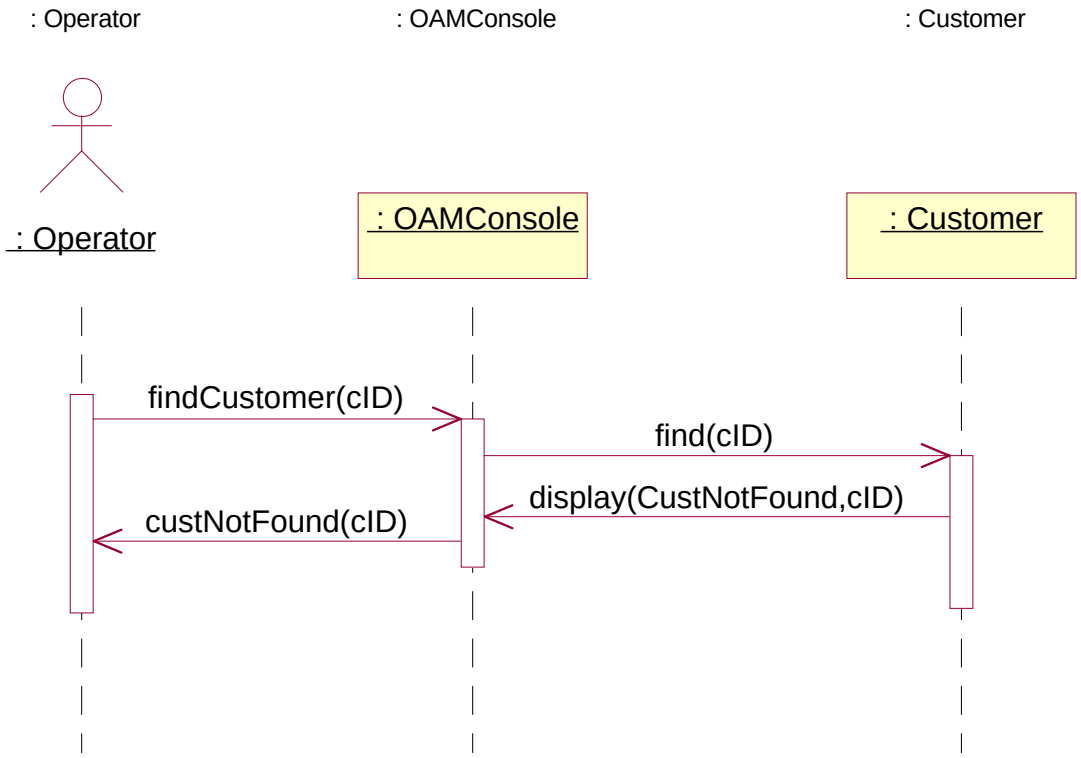

Figure 55: Sequence Diagram -Find Customer by cID error

The Find Customer by cID Sequence Diagram illustrates the sequence of events in which a customer is searched for but not found. The sequence begins with the Operator sending a findCustomer event to the OAMConsole. The OAMConsole then relays this message onto the Customer class. The Customer class then searches through the Customers and finds that no such Customer exists that matches the cID specified. The Operator then returns display no matching customer found to the OAMConsole. The OAMConsole then sends this message to the Operator to receive it.

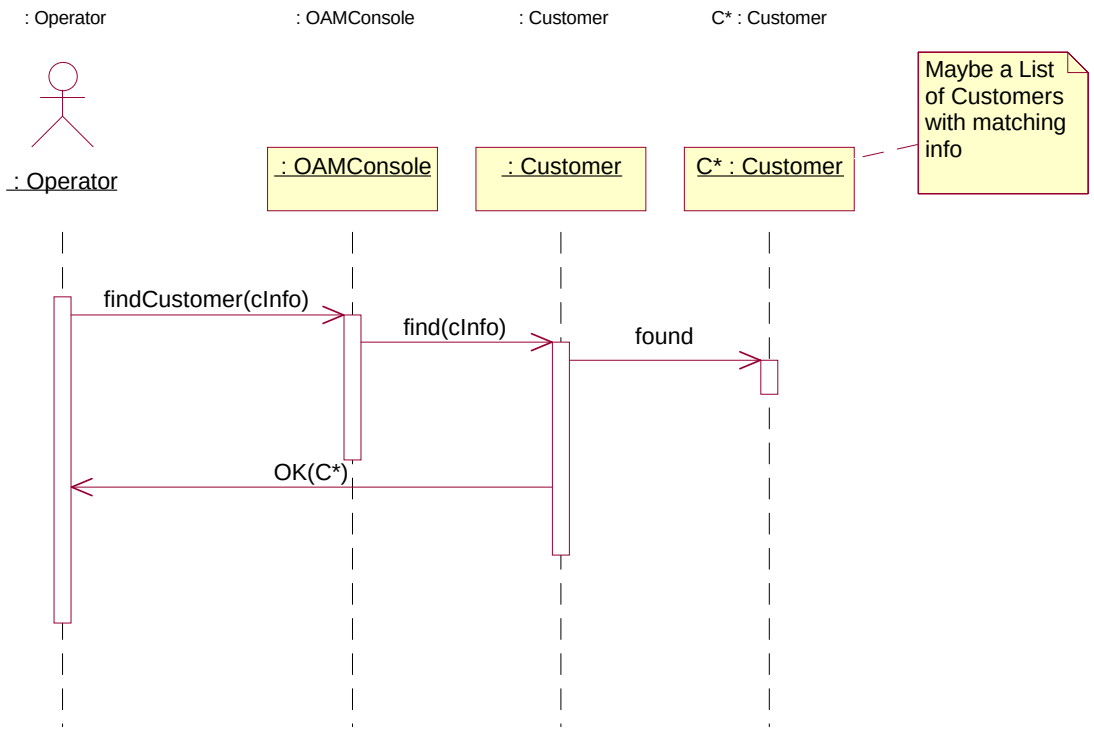

Figure 56: Sequence Diagram -Find Customer by Info

The Find Customer by Info Sequence Diagram traces the successful find of a list of customers that match the given information. The sequence begins by the Operator sending a findCustomer event to the OAMConsole given some customer information. The OAMConsole then sends this event to the Customer class which then builds up a list of Customers that match the information specified. If at least one is found then the list is returned. When none are found, that is shown in the Find Customer by Info error Sequence Diagram. The Customer class then returns this list of Customers to the Operator.

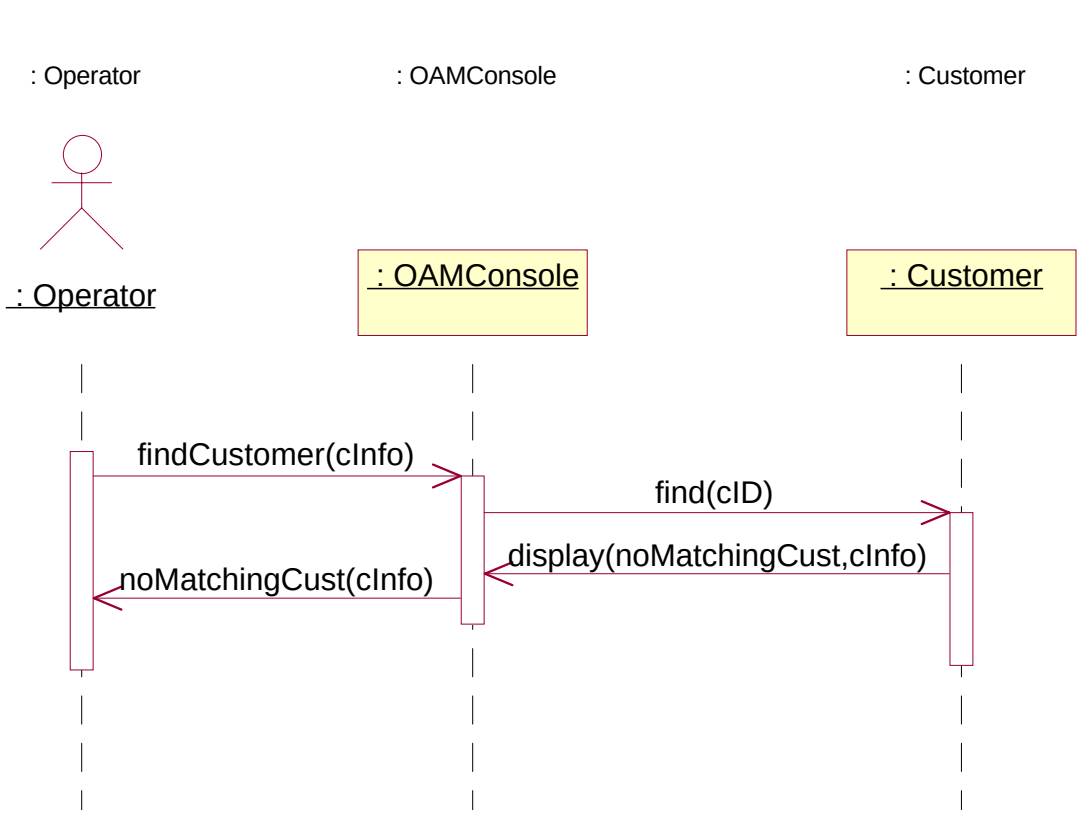

Figure 57: Sequence Diagram -Find Customer by Info error

The Find Customer by Info error Sequence Diagram shows the erroneous state which is reached when no customers are found that match the information provided by the Operator. The sequence is very much like the success condition (see Find Customer by Info Sequence Diagram) but with different results. The sequence begins with the Operator sending a findCustomer event to the OAMConsole. This event is received by the console and is then sent to the Customer class. The Customer class then attempts to find customers that match the information provided. However, in this case it is unsuccessful and returns message to the OAMConsole to display a message stating that no matching customers were found. The OAMConsole then relays this message to the Operator.

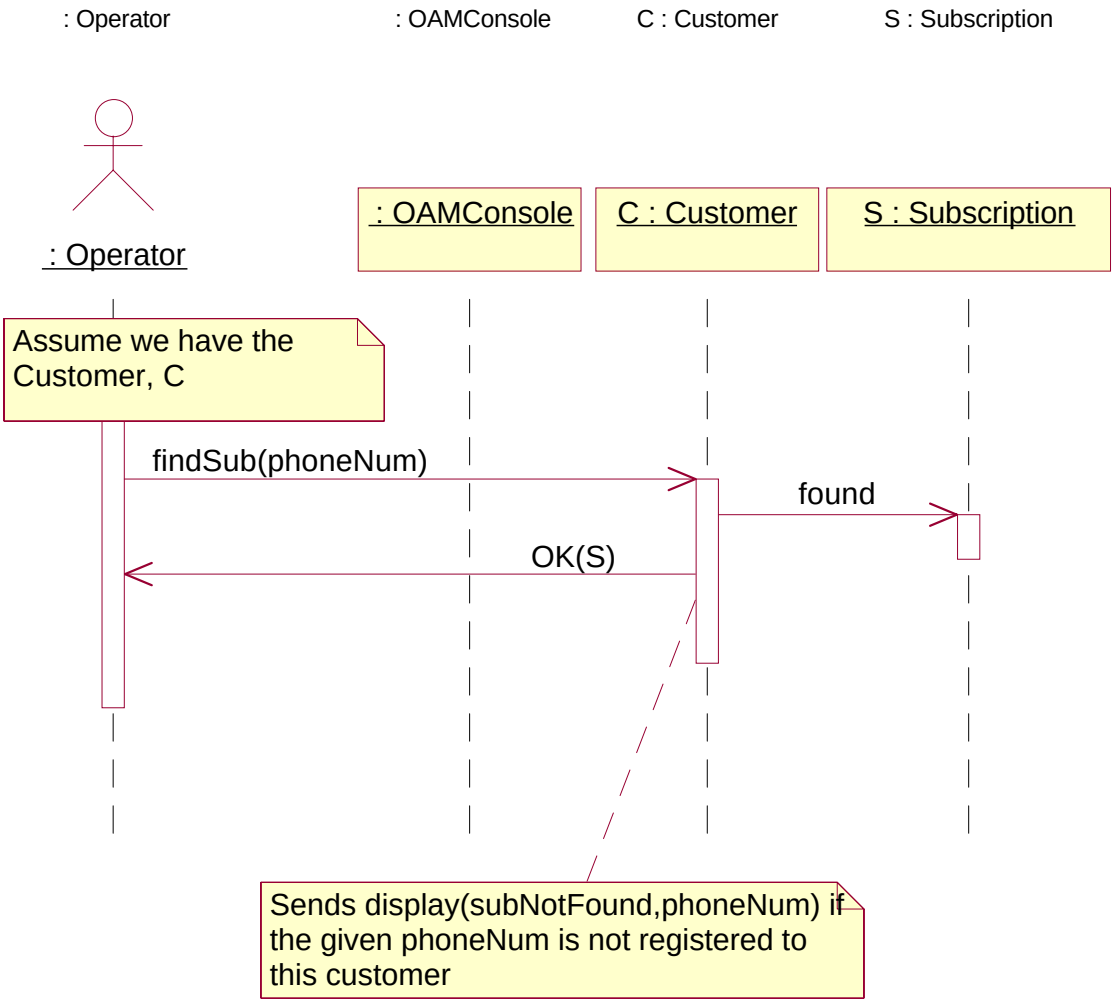

Figure 58: Sequence Diagram -Find Subscription for Customer

The Find Subscription for Customer Sequence Diagram illustrates the Operator attempting to find a subscription given a phone number that is associated with a customer. This diagram assumes that the Operator has already found a Customer. The Operator then sends a findSub event to the Customer with a specified phoneNum. The Customer then searches the subscriptions related to the Customer and finds the subscription has the phoneNum specified. The Subscription is then returned to the Operator via an OK message.

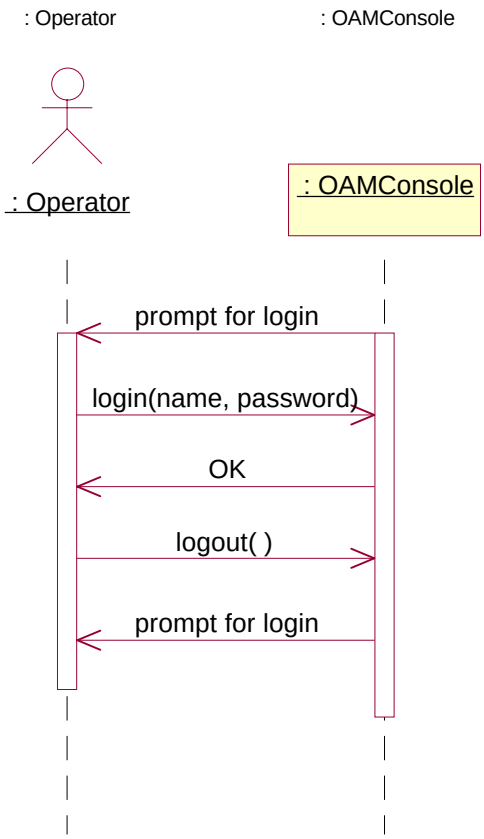

Figure 59: Sequence Diagram - Login and Logout

The Login and Logout Sequence Diagram illustrates the actions for which the Operator must do in order to be authenticated and deauthenicated on the system. The sequence begins with the OAM-Console prompting for a login. The Operator then can respond with a login event that contains the username and password. The OAMConsole then authenticates the username and password. In this case the are valid and correct. In the case they were not, the OAMConsole would return an error to the Operator stating that the username and/or password is invalid. The OAMConsole would then reprompt for login. However, in this case OK is returned. The Operator is then allowed to access the system. Proceeding at a later time the Operator sends a logout event to the OAMConsole. The OAMConsole then deauthenicates the Operator and returns with a prompt for login completing the sequence.

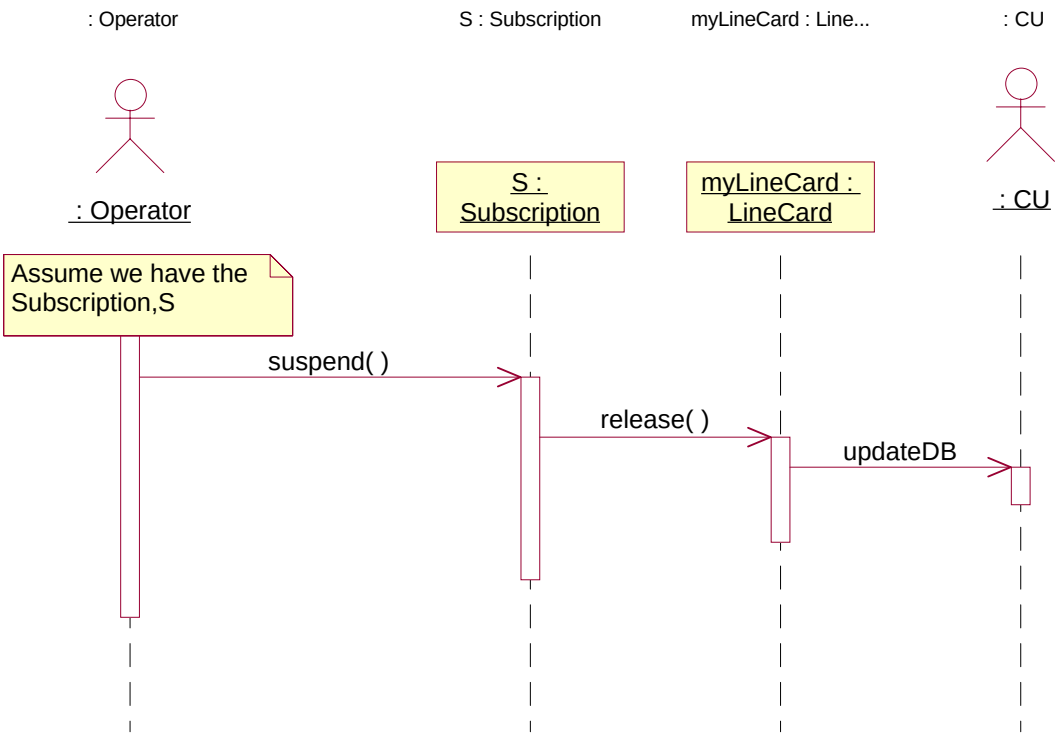

Figure 60: Sequence Diagram - Suspend Subscription

The Suspend Subscription Sequence Diagram traces the suspension of a subscription. The diagram assumes that a subscription has already been found. The Operator first sends a suspend event to the Subscription. The Subscription then releases the associate line card (if one exists). This then causes the LineCard to update the device status which sends a message to the CU to updateDB. These three sequences complete the sequence diagram.

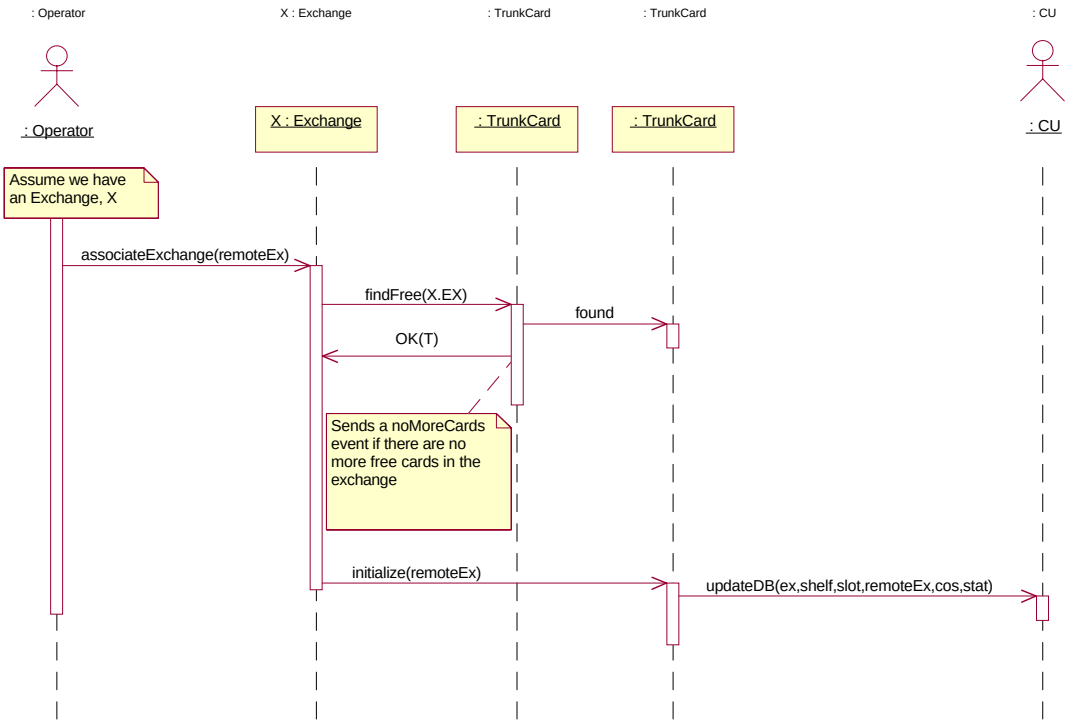

Figure 61: Sequence Diagram - Associate Exchange

The Associate Exchange Sequence Diagram illustrates the steps in which one exchange is associated with another. The sequence diagram begins with the Operator sending an associateExchange event to the Exchange. The Exchange then attempts to find a free card in the current exchange. If a card is found, as it is in this diagram, the card is returned to the exchange. The Exchange then initializes the card and sets the remote exchange number on the card. The initialization requires an updateDB message to be sent to the CU, and then the sequence is complete.

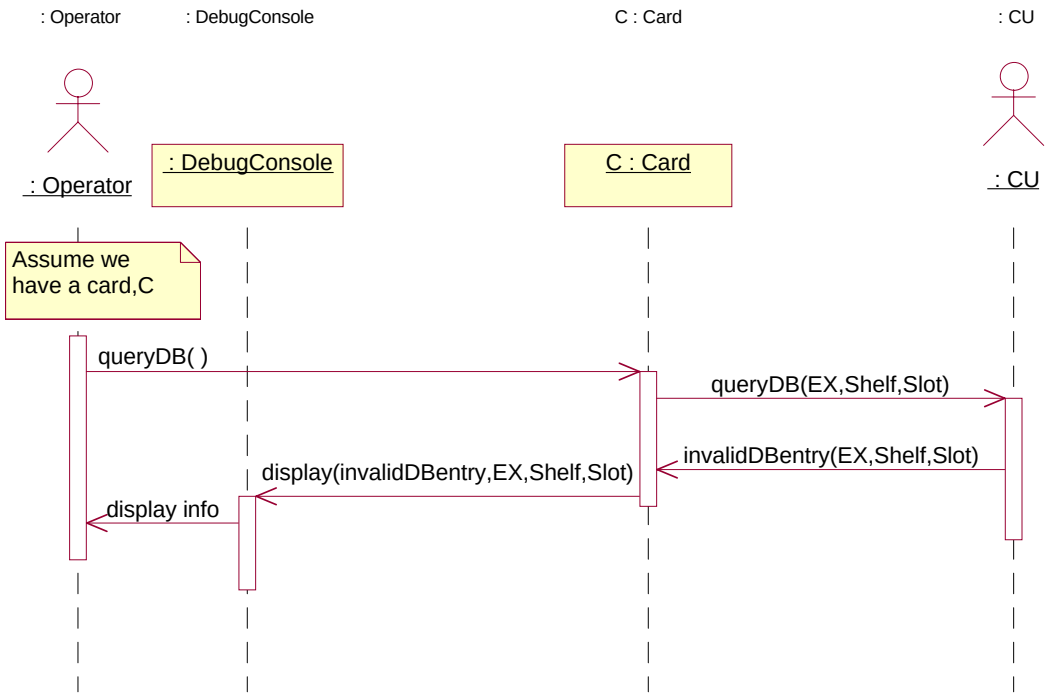

Figure 62: Sequence Diagram - Debug - InvalidDBEntry

The Debug - InvalidDBEntry Sequence Diagram shows the Operator requesting a queryDB message to be sent to the CU, and the response returned of InvalidDBEntry. The Debug - ContentsDB sequence diagram shows a successful similar case, but where the hardware device is found. The sequence begins with the Operator sending a queryDB event to the Card. The Card then relays this event to the CU, via the queryDB message. The Card waits until a response is issued from the CU. The CU responds with the InvalidDBEntry message stating that the card requested is not valid. The Card then displays a message on the DebugConsole stating that the request is invalid. The DebugConsole indicates to the Operator that an invalid query has been requested.

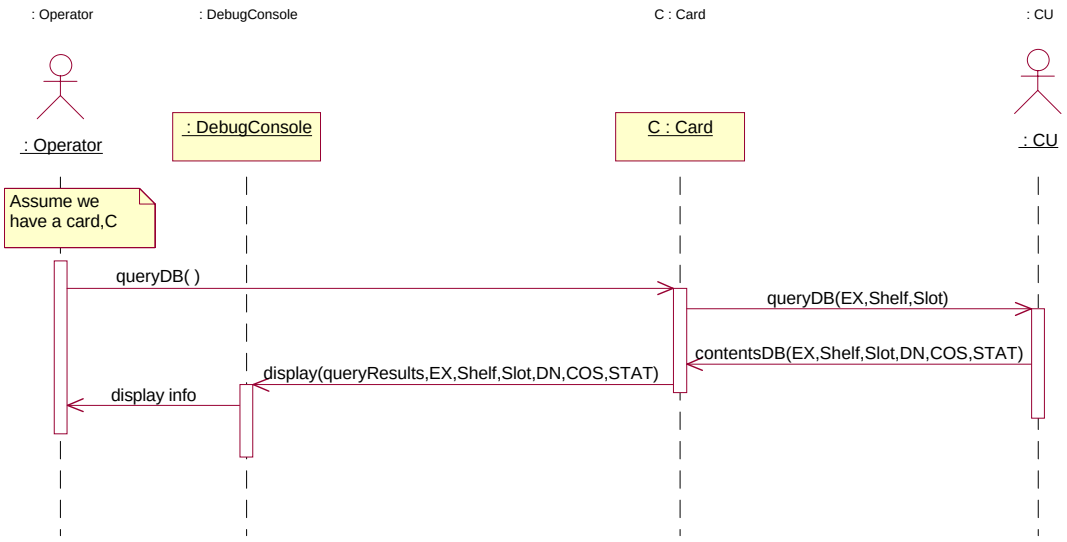

Figure 63: Sequence Diagram - Debug - Return Contents

The Debug - Return Contents Sequence Diagram illustrates a similar sequence to the Debug - InvalidDBEntry sequence diagram. In this diagram a CU is queried of its database and the contents are returned to the Operator. The Operator begins by sending a queryDB event to the Card. It is assumed that a Card has been found already. This event then propagates to the CU through a queryDB message. The Card then waits for a response from the CU. In this case, the response is a contentsDB message. The Card then displays the associated message to the DebugConsole. The DebugConsole then in turn displays the message to the Operator.

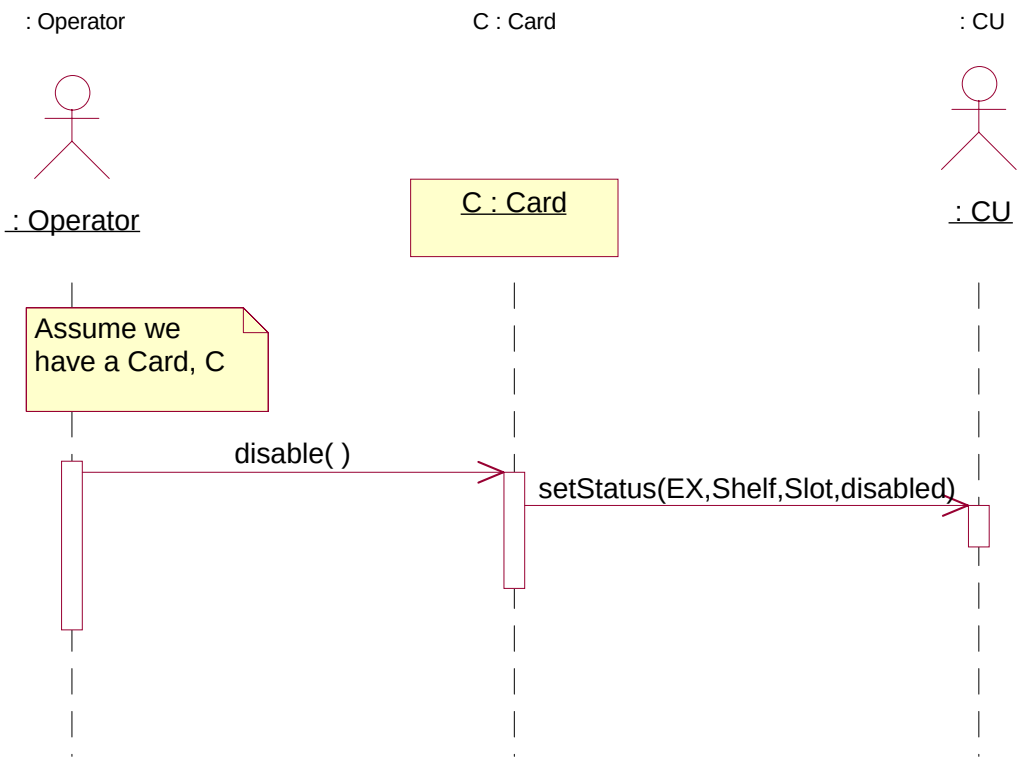

Figure 64: Sequence Diagram - Disable Card

The Disable Card Sequence Diagram traces the path of execution when the Operator wishes to disable a card. It is assumed in the diagram that the card has already been found. This can be done by finding a card by the dialed number, or finding the card by equipment number. The Operator begins by sending a disable event to the Card. The Card then sets its status to disabled which requires a setStatus message to be sent to the CU.

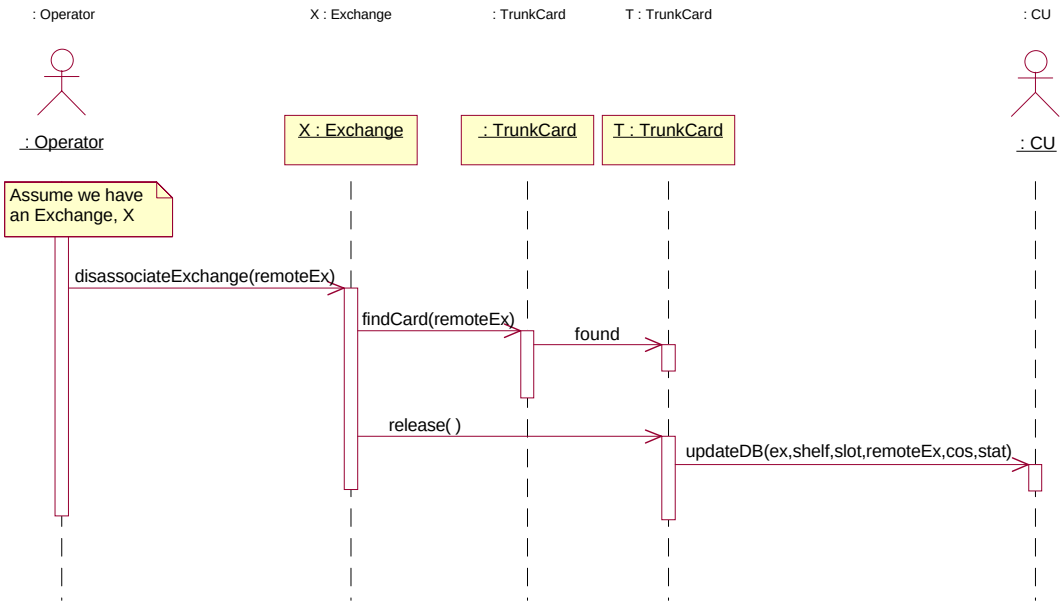

Figure 65: Sequence Diagram -Disassociate Exchange

The Disassociate Exchange Sequence Diagram illustrates the sequence in which a remote exchange is disassociated with the current exchange. The diagram assumes that the current exchange has been found (this can be done as the Find Exchange Sequence Diagram indicates) The Operator sends a disassociateExchange event to the Exchange. The Exchange then finds the trunk card that is associated with this remoteEx number. The Card is found and returned to the Exchange. The Exchange then sends a release event on the card which causes the card to send a setStatus message to the CU. Then the sequence is complete.

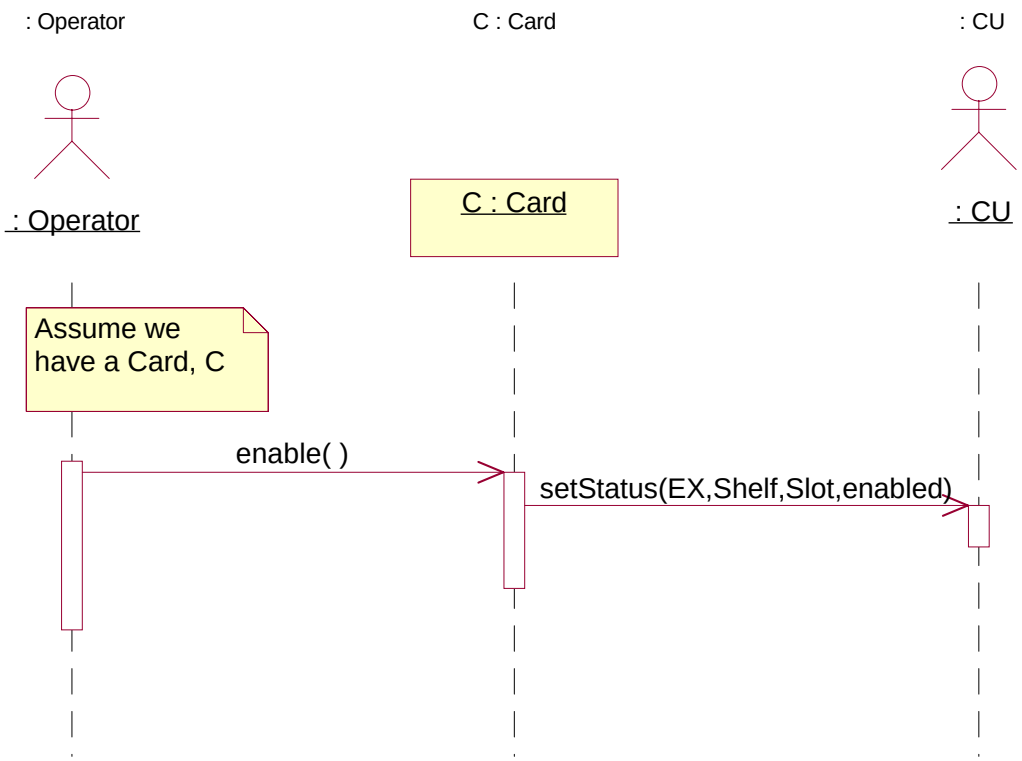

Figure 66: Sequence Diagram -Enable Card

The Enable Card Sequence Diagram illustrates the enabling of a hardware card. The diagram assumes that the card has been previously found. The Operator begins by sending an enable event to the Card. Upon receiving the event, the Card sends a setStatus message to the CU, stating that the cards status has changed completing the sequence.

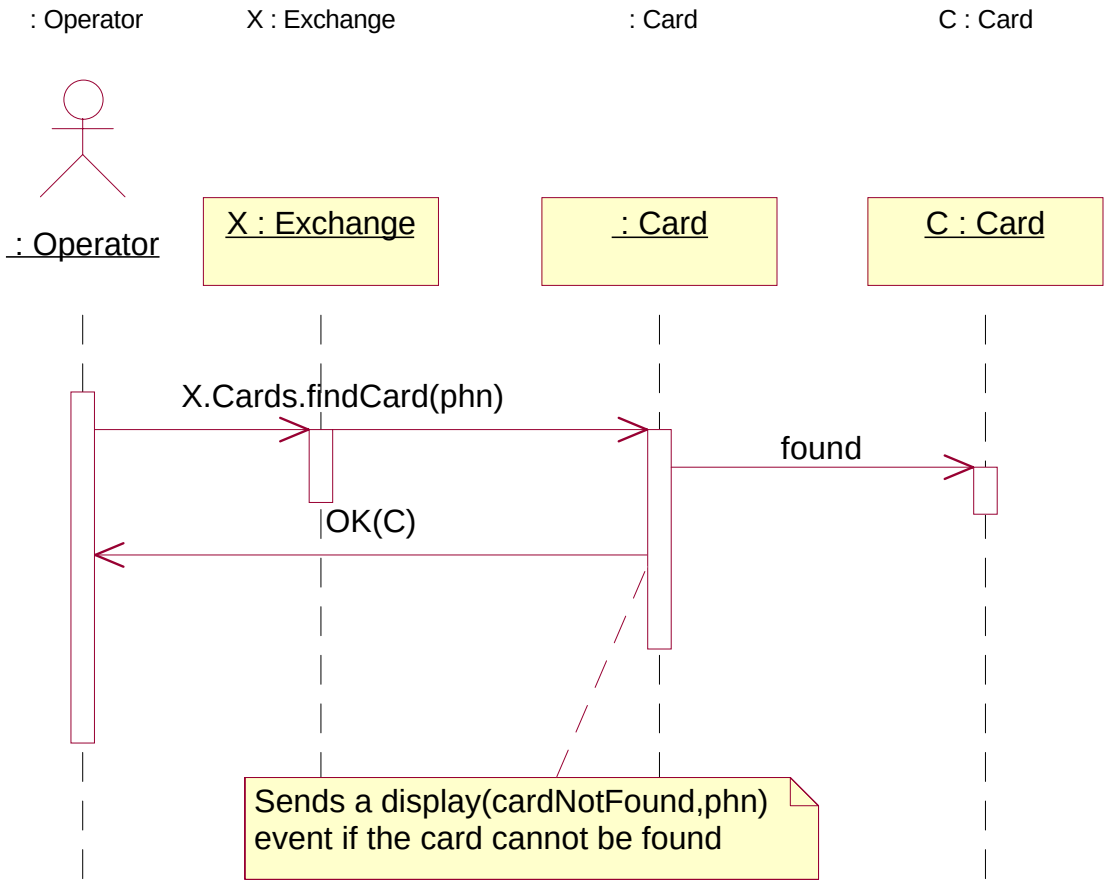

Figure 67: Sequence Diagram -Find Card by dialed number

The Find Card by Dialed Number Sequence Diagram traces the execution of finding a card given a dialed number. The Operator begins by sending a findCard event to the Card. The Card class then attempts to find a card that matches the number provided. If one is found, then the Card returns the found Card to the Operator. Not shown, only noted in the diagram is the fact that the Card can return cardNotFound if a card is not found that matches the provided phone number.

Figure 68: Sequence Diagram -Find Exchange

The Find Exchange Sequence Diagram traces the execution of finding an Exchange given an exchangeNum. The Operator begins by sending a findExchange event to the OAMConsole. This event is then passed on from the OAMConsole to the Exchange. The Exchange class searches through all know exchanges and returns to the Operator an Exchange that has an exchangeNum matching the exchangeNum provided.

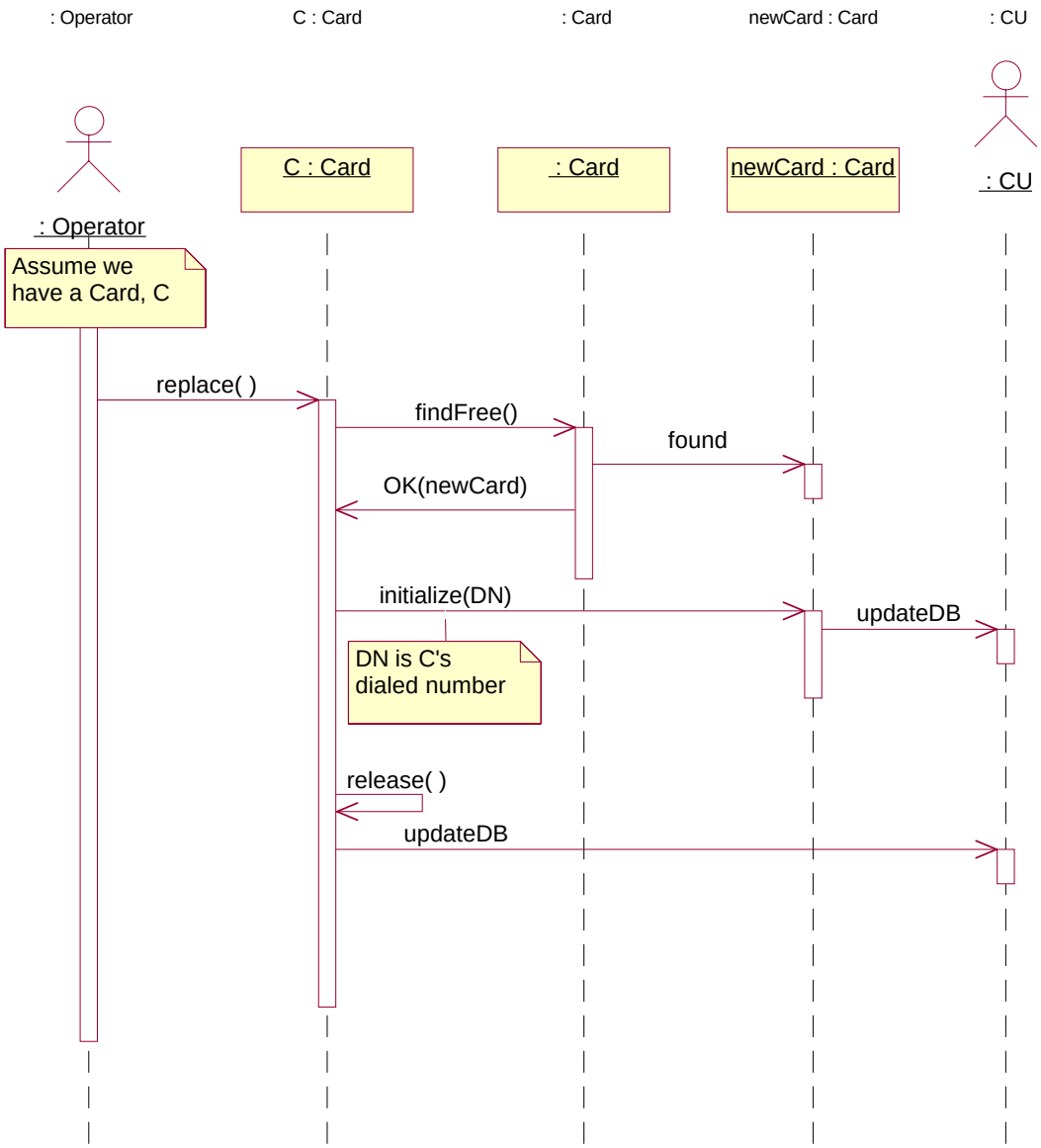

Figure 69: Sequence Diagram - Replace Card

The Replace Card Sequence Diagram illustrates the events that are required when a card is to be replaced. The Operator begins by issuing a replace event the Card that is to be replaced. (It is assumed that it has been previously found) Once this is done, the Card requests a new free card. If one is available the card is returned to the Card that initiated the request. The Card to be replaced then initializes the new card with the DN that the current card has. The current card's status is set to free. An updateDB message is sent to the CU indicating that cards have changed, changing the associations that were previously set up, completing the sequence.

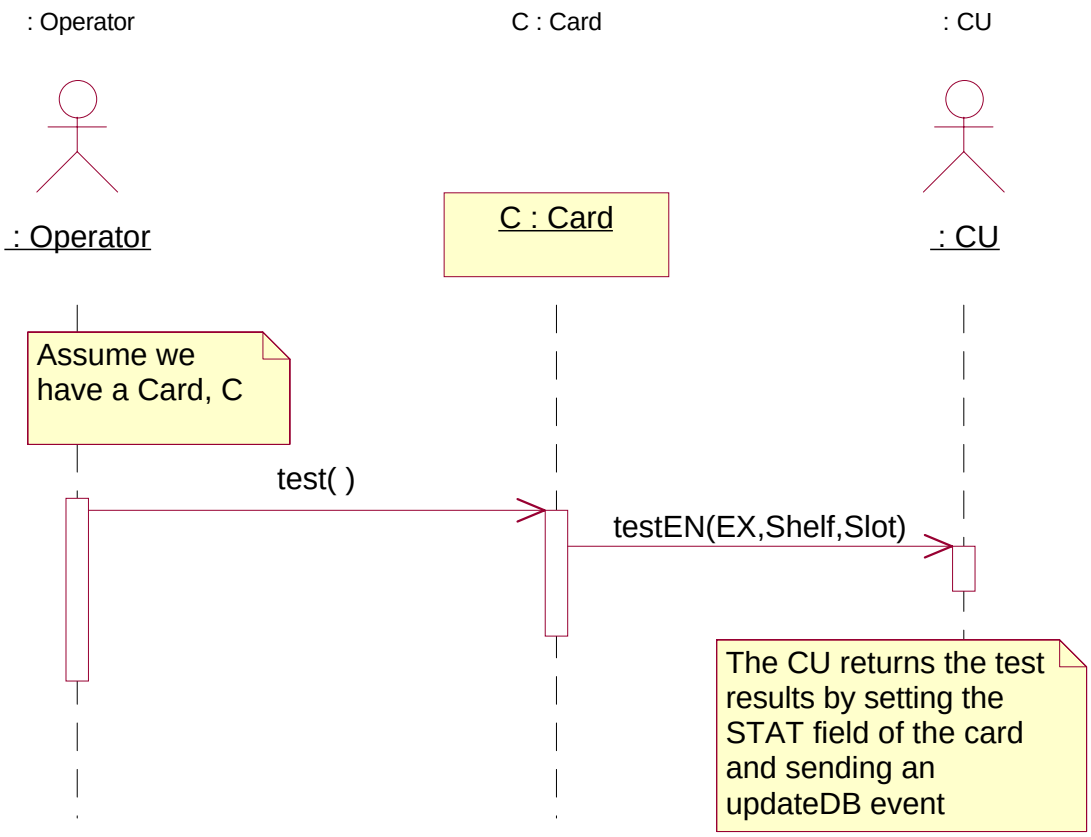

Figure 70: Sequence Diagram -Request Test of Card

The Request Test of Card Sequence Diagram illustrates the functionality of the test event. The Operator begins by sending a test event to the Card. It is assumed that a Card has already been found. The Card then receives the event and sends a testEN message to the CU. The CU receives the message and will respond if the card has a failure through a updateDB message.

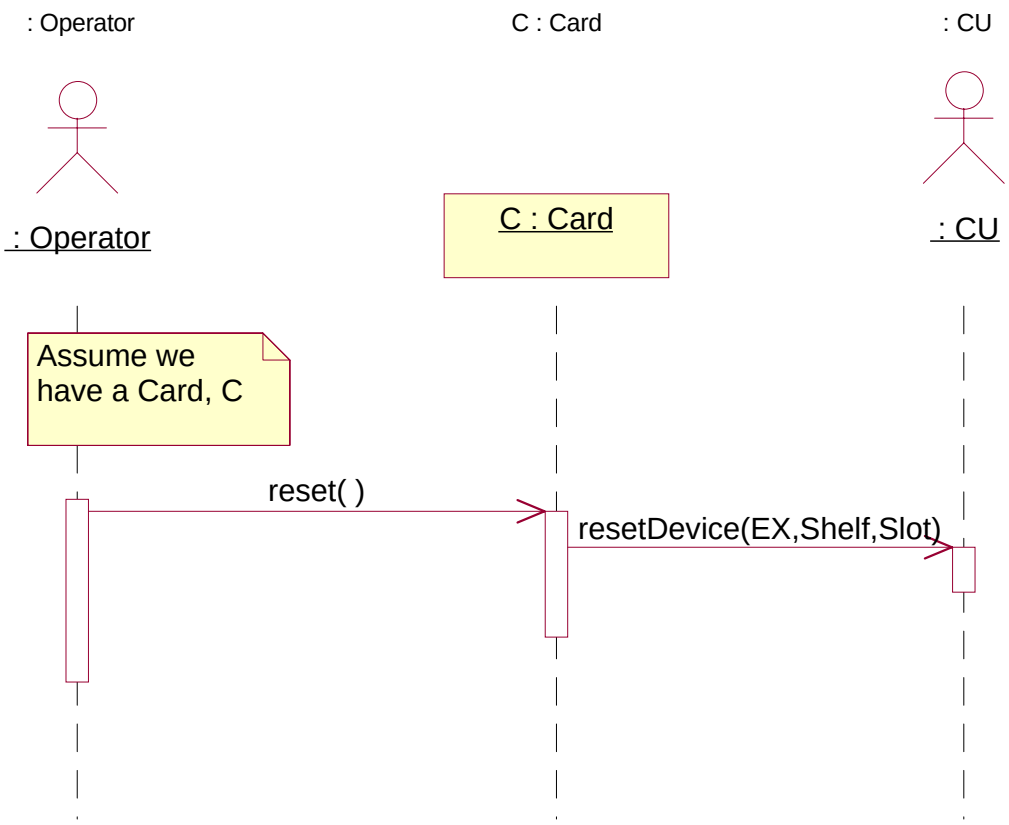

Figure 71: Sequence Diagram -Reset Card

The Reset Card Sequence Diagram traces the execution of the Operator resetting a hardware card. The Operator begins by sending a reset event to the card that was previously found. The Card then sends a resetDevice message to the CU. The CU will then receive the message and do the associated test, completing the sequence.

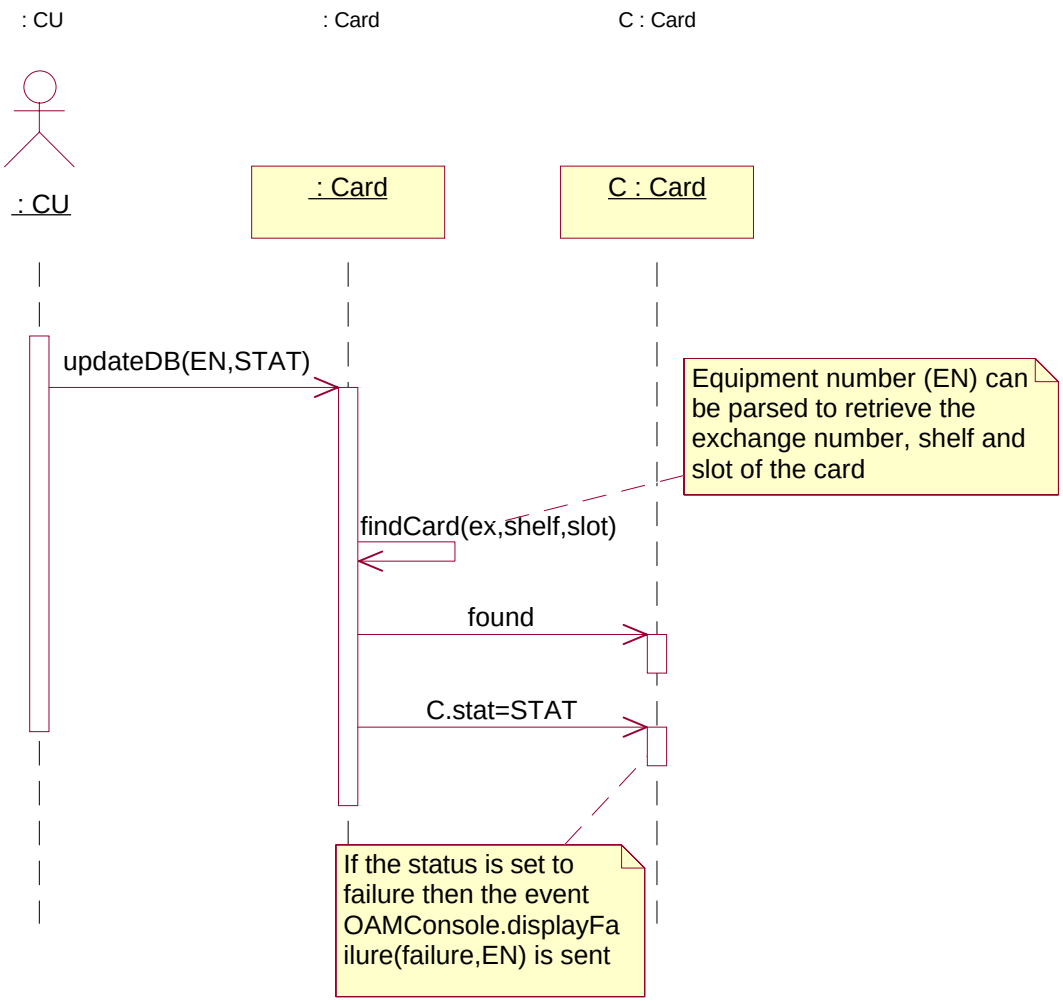

Figure 72: Sequence Diagram -UpdateDB from CU

The UpdateDB from CU Sequence Diagram shows the sequence of events that occur when the CU sends an UpdateDB message to the system. The CU sends the message to the Card class which finds the associate Card object by searching by equipment number (findCard). Once the card has been found, the status on the card is set to the one provided by the updateDB message. The setStatus event on the Card may cause a failure message to be sent to the Operator stating that the card has failed hardware tests.

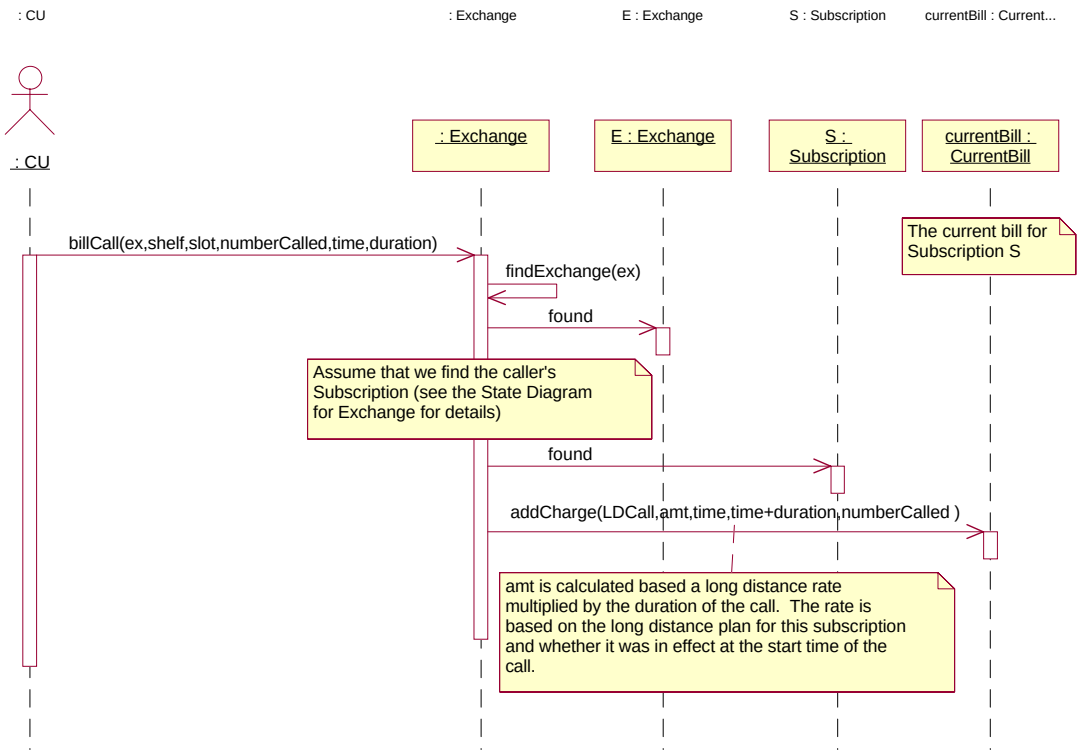

Figure 73: Sequence Diagram -Bill LD Call

The Bill LD Call Sequence Diagram depicts the sequence of events in which the CU sends a bill call message to the Exchange, and the call is a LD call. The Bill Local Call Sequence Diagram shows the case in which the call is a local call. The sequence begins with the CU sending a billCall message to the Exchange. The Exchange then attempts to find the associated exchange E. Once this is found, the subscription associated with the card is found. This is shown in the findSubscription sequence diagrams. After the subscription is found, the charge requested to be added to the current bill of the subscription. The Subscription will add the LDCall Charge to the bill, completing the sequence.

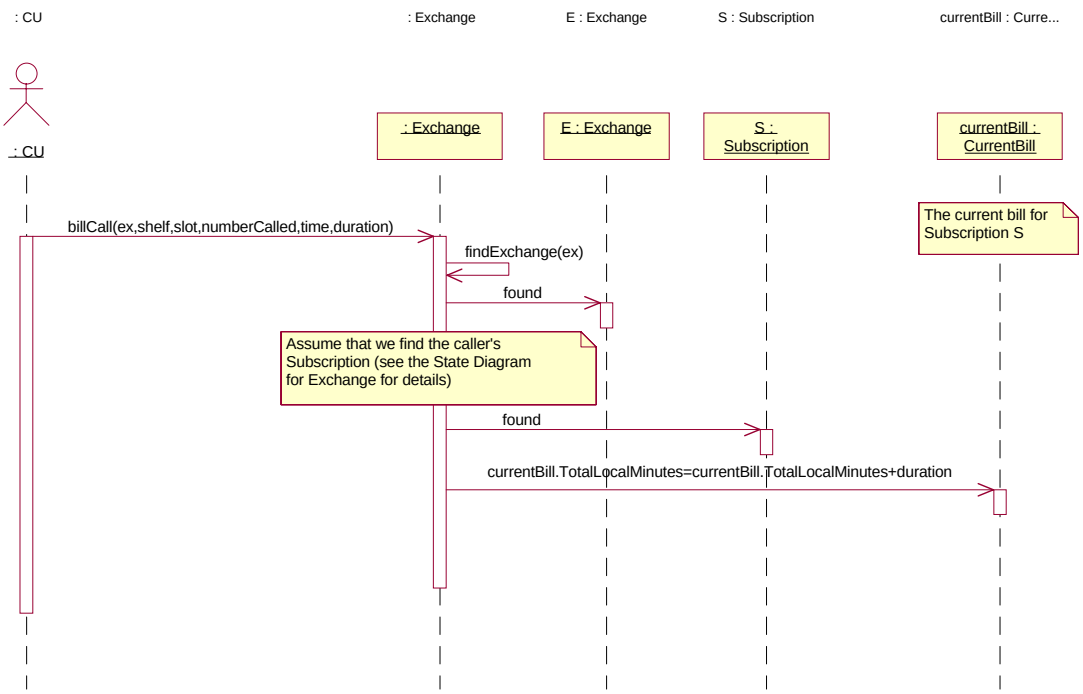

Figure 74: Sequence Diagram -Bill Local Call

The Bill Local Call Sequence Diagram depicts the sequence of events in which the CU sends a bill call message to the Exchange, and the call is a Local call. The Bill LD Call Sequence Diagram shows the case in which the call is a LD call. The sequence begins with the CU sending a billCall message to the Exchange. The Exchange then attempts to find the associated exchange E. Once this is found, the subscription associated with the card is found. This is shown in the findSubscription sequence diagrams. After the subscription is found, the total local minutes of the current bill are incremented on the subscription. This completes the sequence.

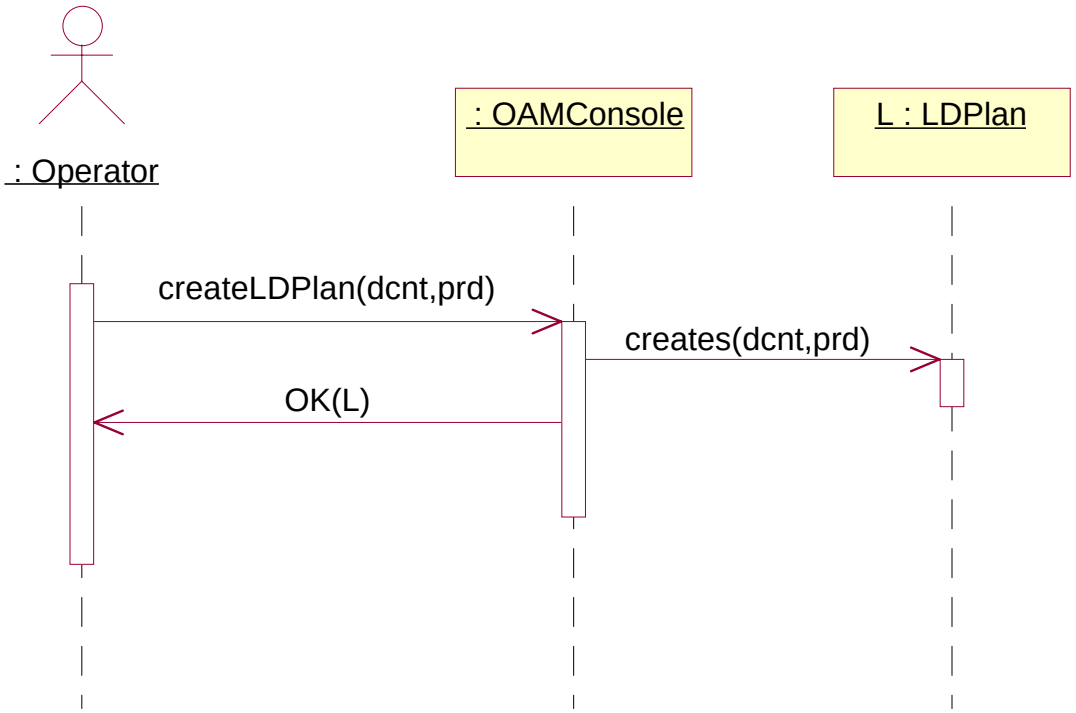

Figure 75: Sequence Diagram -Create LD Plan

The Create LD Plan Sequence Diagram shows how a Operator creates an LD Plan. The Operator begins the sequence by sending a createLDPlan to the OAMConsole. The OAMConsole then generates an event that creates the new LDPlan. The properties for the LDPlan are passed into the create of the new LDPlan. The OAMConsole then returns this newly created LDPlan to the Operator.

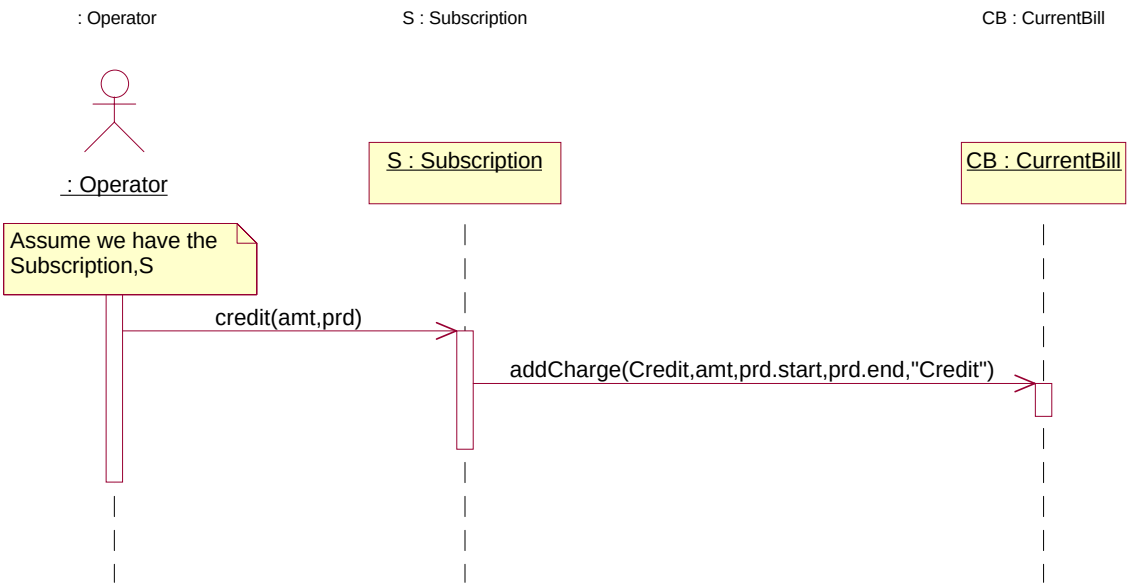

Figure 76: Sequence Diagram -Credit Account

The Credit Account Sequence Diagram illustrates the events that occur when the Operator requests a credit be added to a subscription. The Operator begins by sending a credit message with an amount and period to the Subscription. It is assumed that we have previously found the associated subscription which we want to make changes on. The message is received by the Subscription who requests a Credit Charge be added to the Current Bill of the subscription. Once this has been completed, the sequence is complete.

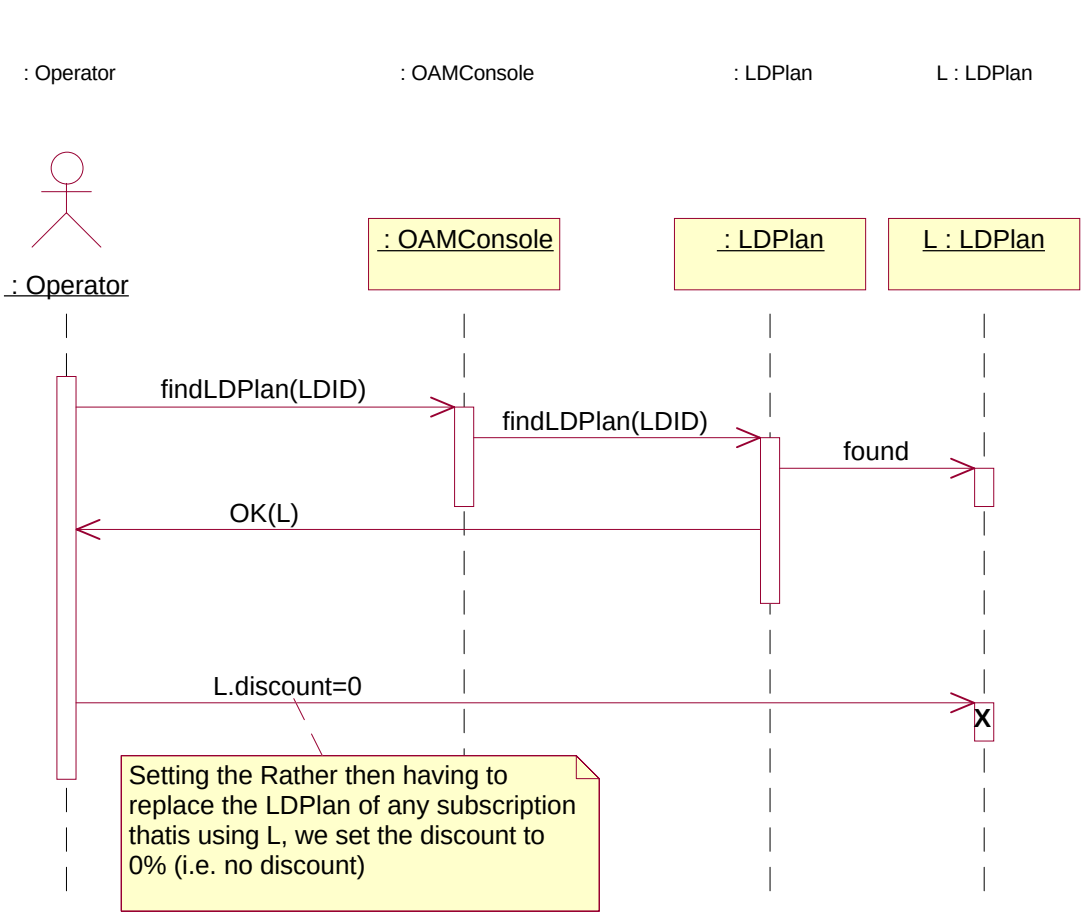

Figure 77: Sequence Diagram -Delete LD Plan

The Delete LD Plan Sequence Diagram traces the events that are sent when an LD Plan is requested to be removed. Firstly, the Operator needs to find the associated LD Plan. This is done by the Operator requesting a findLDPlan from the OAMConsole. The OAMConsole then sends this message to the LDPlan via a findLDPlan given a LD identifier. When the LDPlan is found, it is returned to the Operator. The Operator then sends a set discount event to the LDPlan. The discount rate is set to 0, stating that the LD Plan has no discount specified.

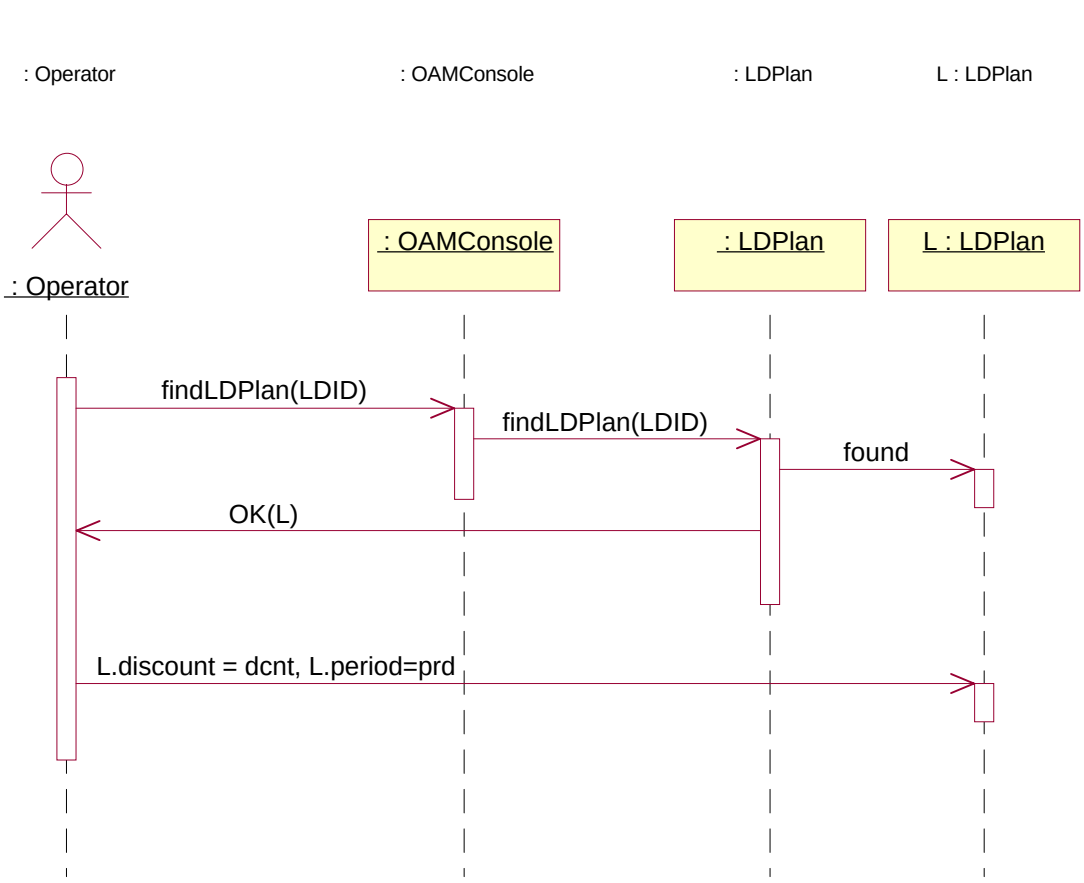

Figure 78: Sequence Diagram -Edit LD Plan

The Edit LD Plan Sequence Diagram traces the events that are sent when an LD Plan is requested to be edited. Firstly, the Operator needs to find the associated LD Plan. This is done by the Operator requesting a findLDPlan from the OAMConsole. The OAMConsole then sends this message to the LDPlan via a findLDPlan given a LD identifier. When the LDPlan is found, it is returned to the Operator. The Operator then sends a set event to the LDPlan assigning the values of the period and discount.

: Operator S : Subscription B : Bill

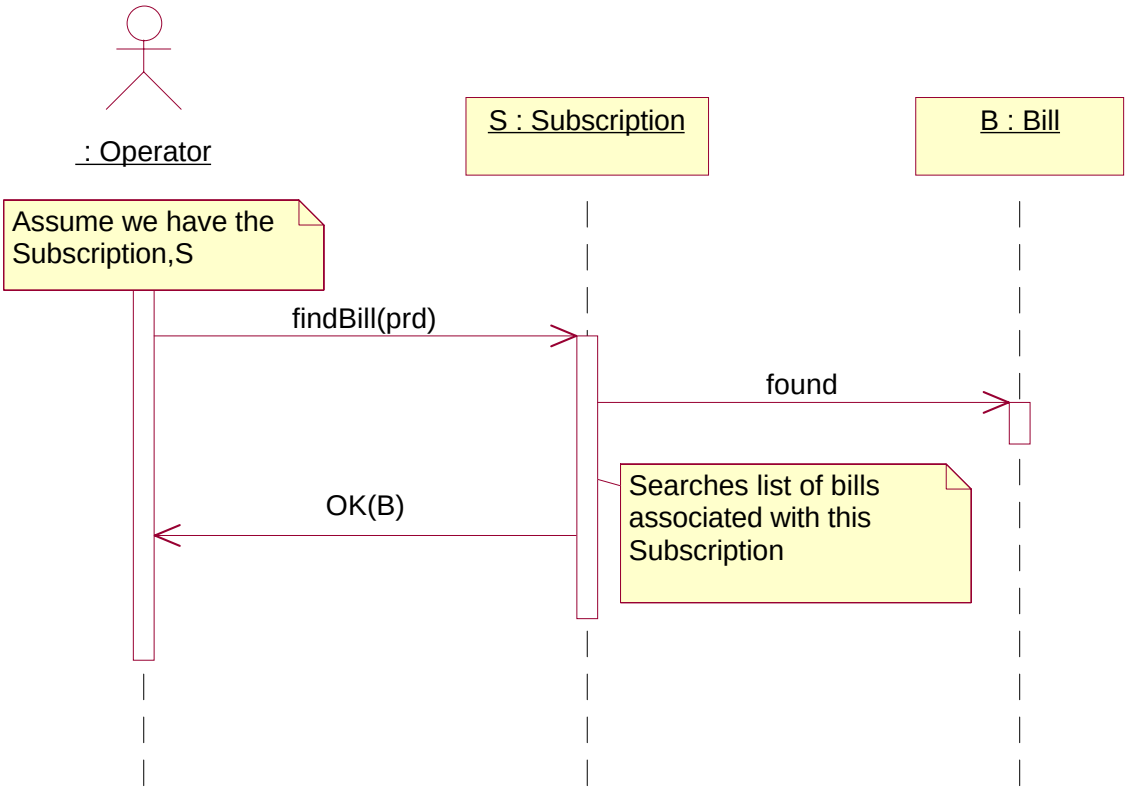

Figure 79: Sequence Diagram -Find Bill

The Find Bill Sequence Diagram traces the execution of finding a bill given a time period. The Operator is allowed to view any associated with a subscription. The Operator begins the sequence by sending a findBill event to a Subscription (that was previously found). The Subscription then searches the list of bills associated with the subscription attempting to find one that matches the period specified. If a bill is found, then a OK message with the found bill is returned to the Operator.

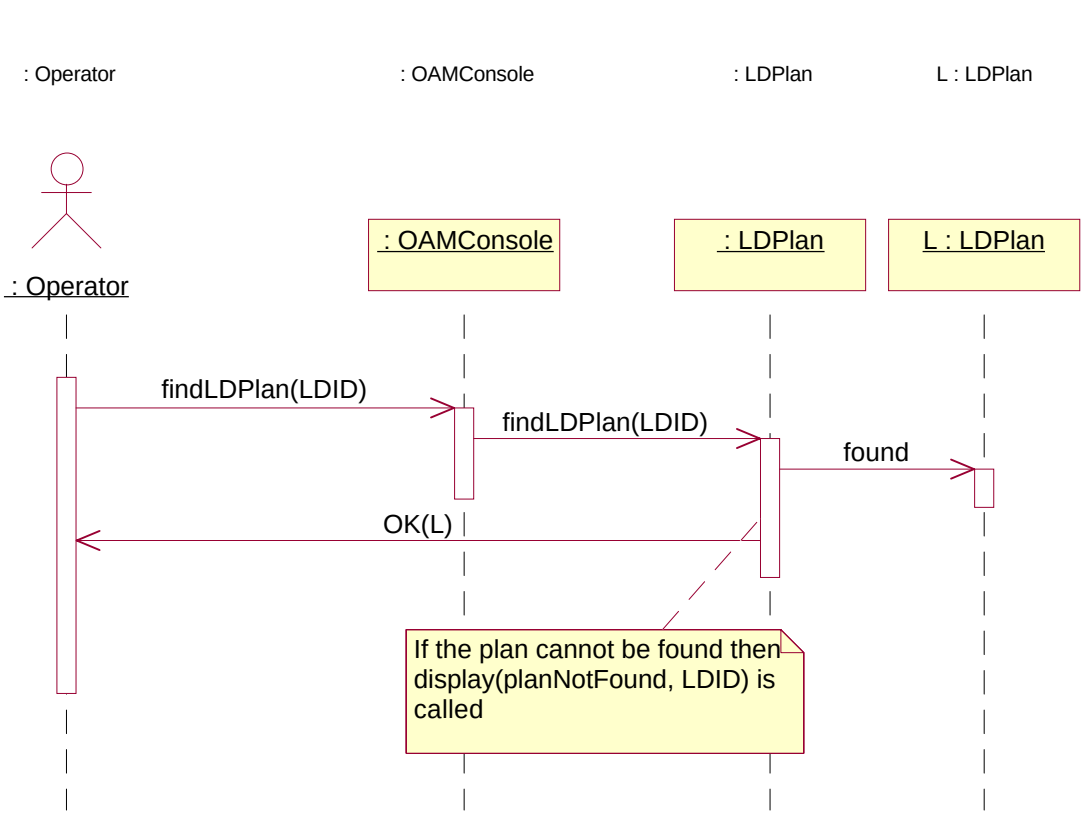

Figure 80: Sequence Diagram -Find LD Plan

The Find LD Plan Sequence Diagram traces the execution of the findLDPlan event. The Operator begins by sending a findLPLan to the OAMConsole. The OAMConsole the relays the findLDPlan message to the LDPlan. The LDPlan searches all of the available LDPlans and attempts to find one that has the same identifier as the one requested. Once the LDPlan is found it is returned to the Operator. If none is found, which is not shown in this diagram, an error is returned to the OAMConsole and is displayed to the Operator.

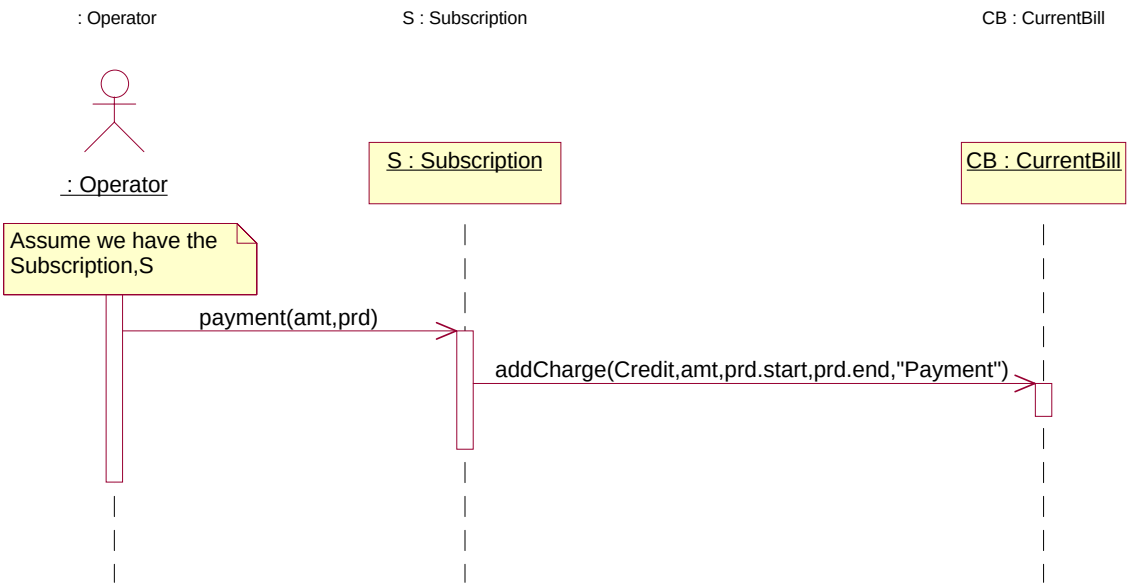

Figure 81: Sequence Diagram -Make Payment

The Make Payment Sequence Diagram shows the sequence of execution that the system takes when a payment is made. It is assumed that the Operator has already found the Subscription which the payment is to be applied to. The Operator begins by sending a payment message to the Subscription. The Subscription then invokes the add Charge method on the current bill. The CurrentBill then creates an associated charge to the list of charges, that is of type Credit. This diagram is very similar to the Credit Account Sequence Diagram.

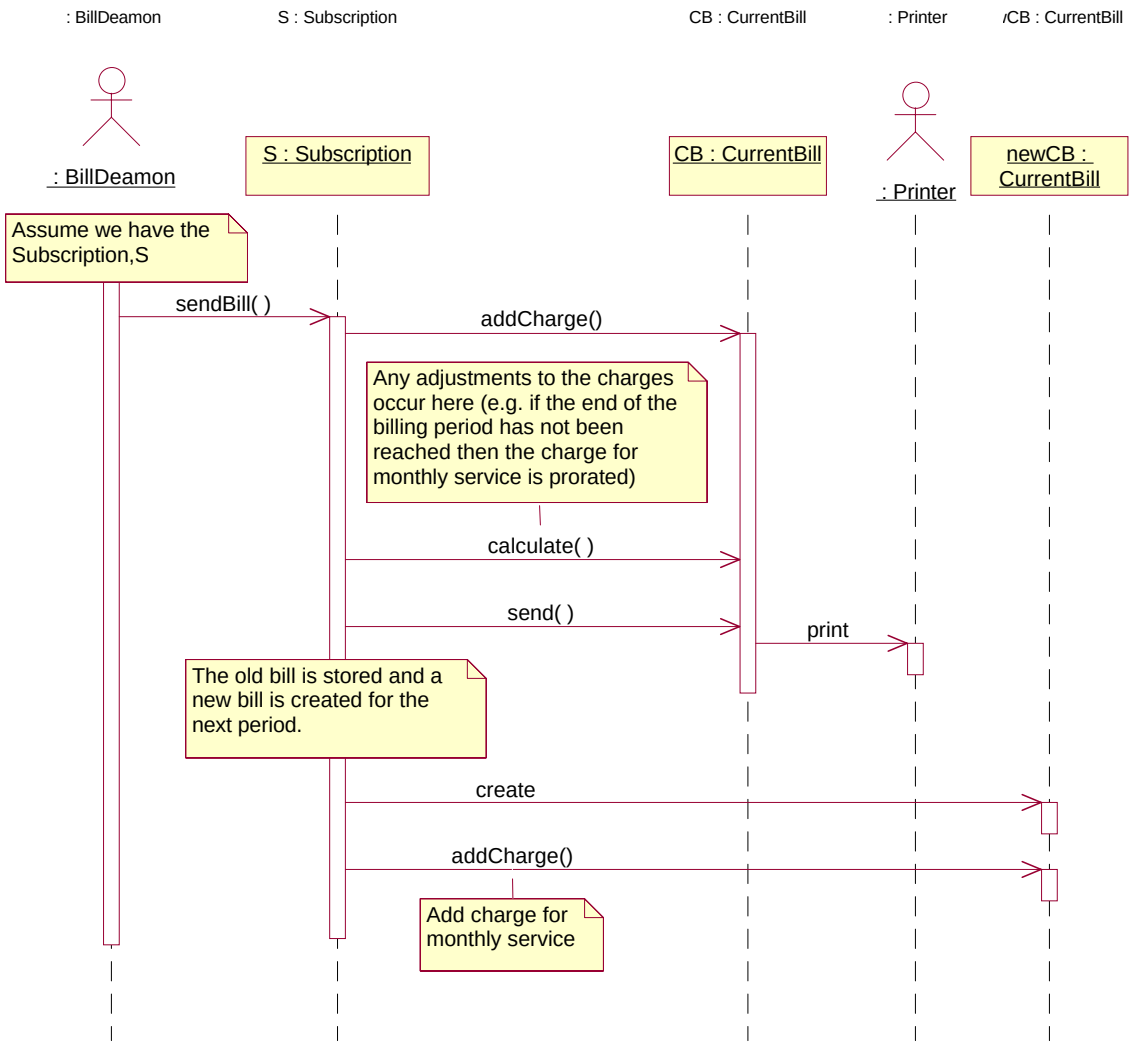

Figure 82: Sequence Diagram -Send Bill

The Send Bill Sequence Diagram depicts the situation where the BillDeamon sends periodic billing messages to print the bills. The sequence is similar to the sequence where the Operator requests printing of a bill, but that case is a little less complicated. This sequence begins with the BillDeamon sending a sendBill message to the Subscription. The Subscription then calculates the CurrentBill and then signals it to be sent. The send message sent to the CurrentBill causes the CurrentBill to be sent to the Printer. Once this is done, the Subscription stores the old bill in the list of previous bills and creates a new Current Bill.

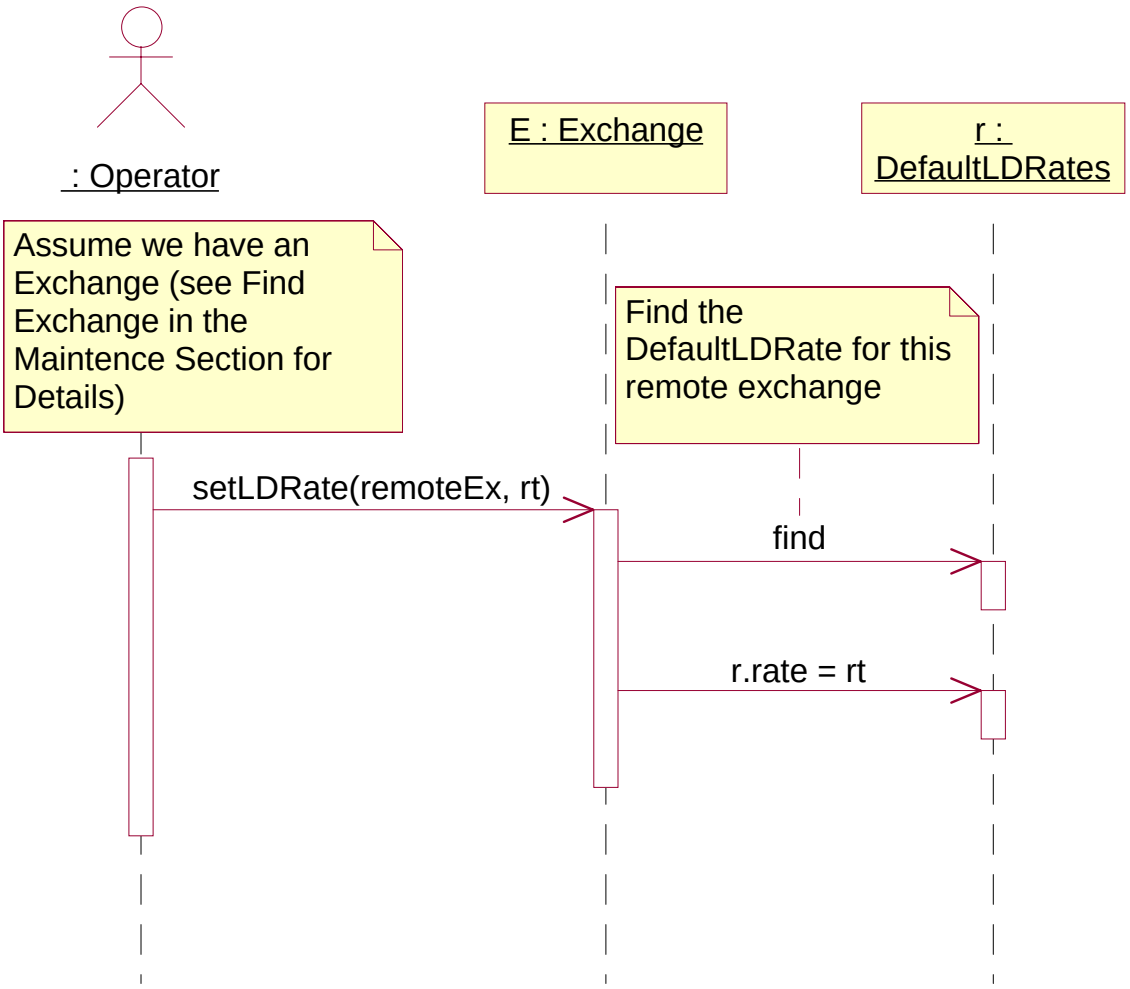

Figure 83: Sequence Diagram -Set Default LD Rates

The Set Default LD Rates Sequence Diagram sets the default LD rates on an exchange. It is assumed that the Exchange is found prior to calling setLDRate(). The sequence begins with the Operator sending a setLDRate message to the Exchange. The exchange then finds the associated Default LD Rates class so that it can update it. The Exchange then sets the rate on the DefaultL-DRates and this completes the sequence.

: Operator E : Exchange

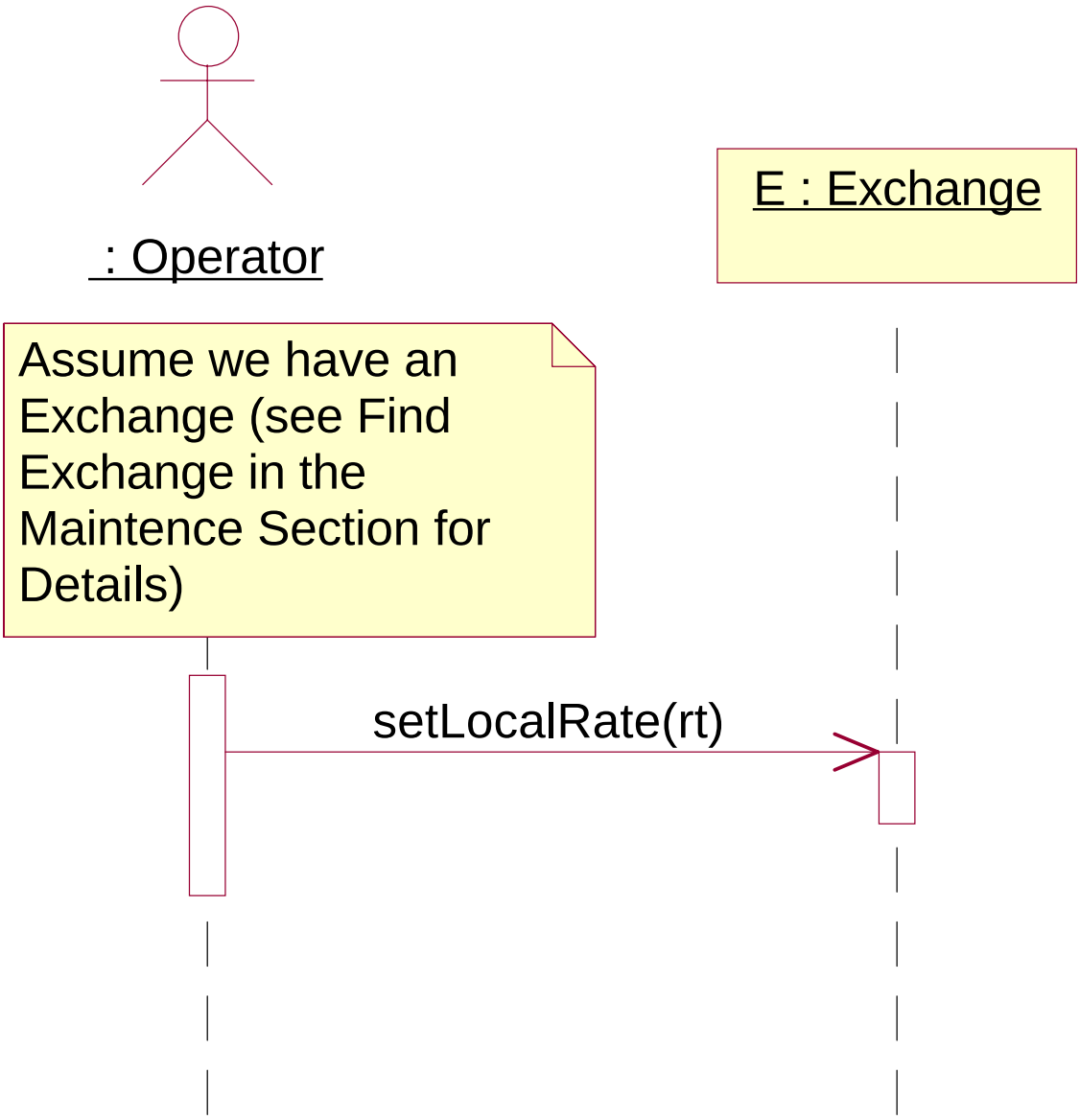

Figure 84: Sequence Diagram -Set Local Billing Rate

The Set Local Billing Rate Sequence Diagram traces the execution of updating an Exchange's local billing rate. The assumption is made that the Exchange has been previously found. The Operator begins by sending the event setLocalRate to the Exchange. After this the Exchange sets an internal attribute and the sequence completes.

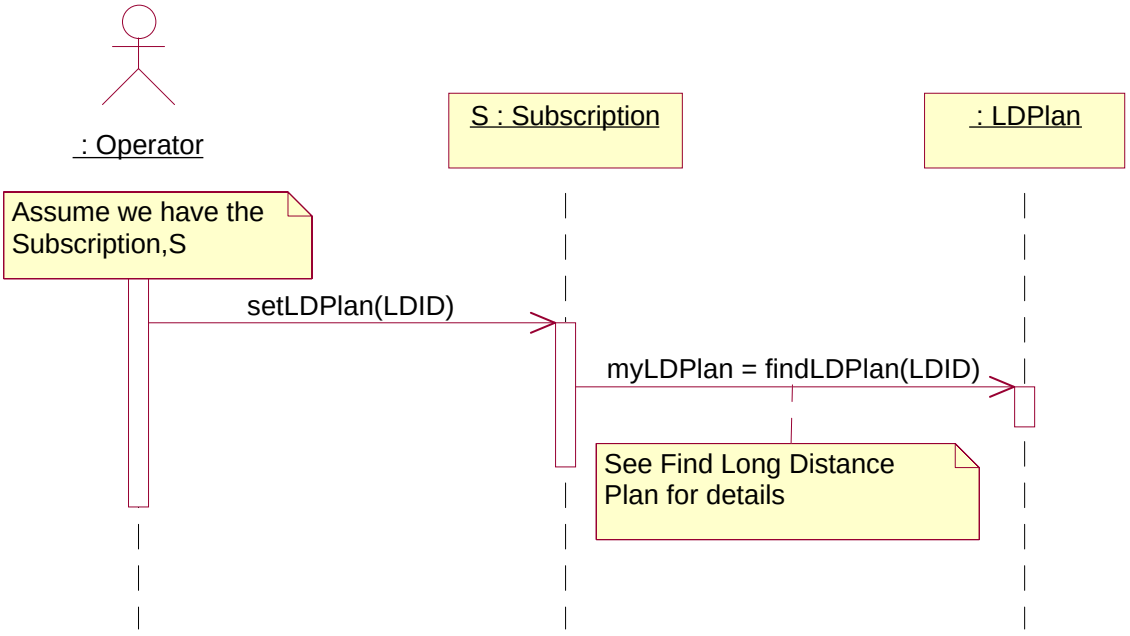

Figure 85: Sequence Diagram -Set Subscription's LD Plan

The Set Subscription's LD Plan Sequence Diagram traces the execution of the Operator requesting an LD Plan be set for a specific Subscription. The Subscription is assumed to be found previously. The Operator then sends a setLDPlan event to the Subscription which then assigns the private attribute of myLDPlan to be the LDPlan that is associated with the LD plan id that was specified.

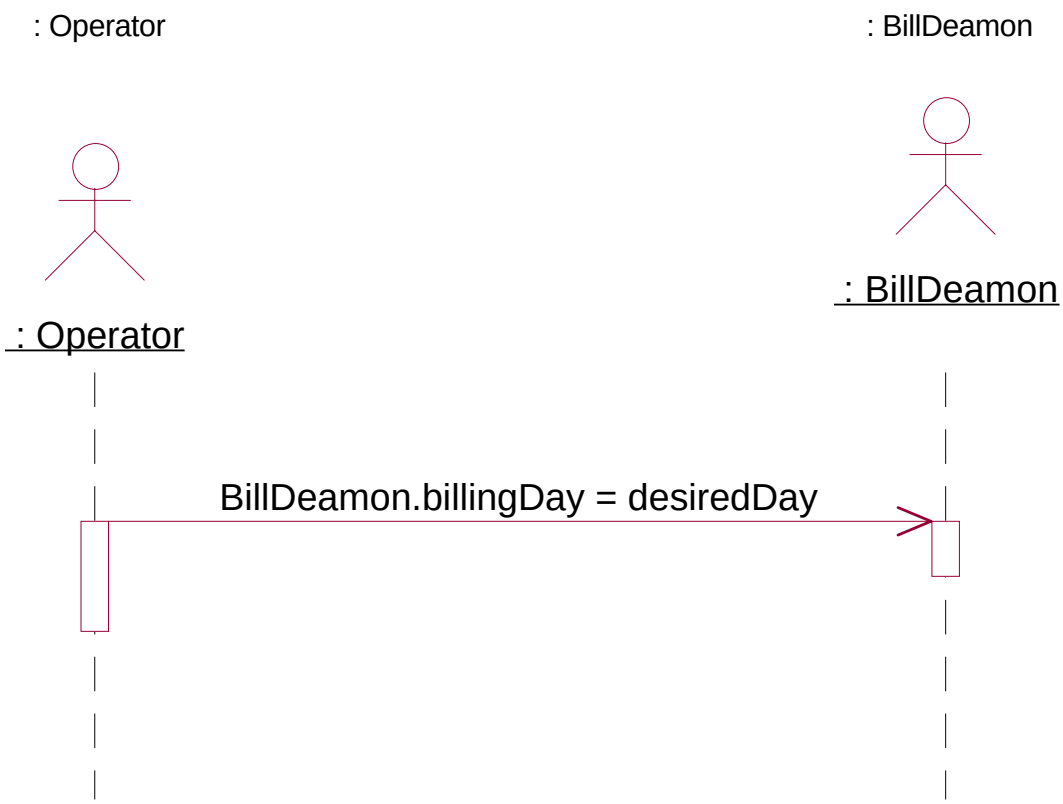

Figure 86: Sequence Diagram -Set Subscription's LD Plan

The Set Billing Date Sequence Diagram traces the execution of the Operator requesting a change of the current billing date. A request is sent to the BillingDeamon Actor so that the day can be updated. When the BillingDeamon encounters this day it will then send out bills corresponding to the new period.

3.2.3 Functional Requirements

The functional requirements have been specified in the Use Cases, Sequence Diagrams, State Diagrams and Conceptual View.

3.3 Performance Requirements

Performance is a key requirement in the functionality of the system. If the system's performance is slow and operator's requests take a long time, this in turn will cause customers to be annoyed. The system must have quick response time so that operators can have high throughput; any failure in this regard should be viewed as harshly as a system failure.

3.4 Accuracy Requirements

The system should perform in an accurate manner and not misrepresent data to the operator. An example of misrepresentation is: The system might have an implementation on a search for customer "Joe, Smith" the system may return only the first 100 matches. The system must then

present this to the operator that the query was truncated, and not misrepresent it by just displaying the 100 entries.

The system should ensure that all accounting of billing is accurate and report any errors to the operator.

3.5 Non-behavioral Requirements

3.5.1 Security and Integrity

It is a requirement that the application be secure. [14] An operator login and password have been provided so that the system can have controlled access. Before an operator can do any action on a OAM console, they must login and be authenticated. This is meant to ensure that only users that have the correct authentication are allowed to perform actions on the system.

The system should be robust to abuse. Operators that try to add bogus, invalid customers and those who abuse the system should be denied ability to perform such actions. To satisfy this requirement the system may be required to perform integrity checks and auditing.

The system should not permit Operators to change read only records such as bills (only changes are made to the current one and not previous ones).

3.5.2 Usability

The User Interface has been designed so that minimal training would be required to use the system. With this in mind, the forms were designed to have similar layouts, thus orientation within a any set of controls will be familiar to the operator once they have used a few of the screens.

While designing the interface it was noted that the operator must be able to navigate quickly, since the operator may be on the phone with a customer. To this end, a back button has been implemented so that it returns the user to the previous screen. If the user decides that the current screen chosen was not the correct one then the user is simply two clicks away from selecting the correct screen. Whereas if the user did not have this back button, they may be required to goto a main screen and re-enter information if necessary.

3.5.3 Scalability

The system was specified with scalability in mind. For example, the ability to add more operator consoles is available; however, a few problems would need to be addressed. One such problem being: which operator would receive failure messages.

The system could also support more than nine CUs with a only a few changes, most of which would involve hardware and CU-OAM interface changes.

Finally, the system provides the ability to extend the COS to encapsulate more types of services offered. Additional attributes in the COS could be added so that new services on phone lines could be added to the system.

3.5.4 Data Integrity

The system should ensure that the data within the system does not get corrupted or maliciously modified. This is important since the system maintains the master copy of the database for all of the exchanges. If entries in the OAM are corrupted, the system then could distribute these corrupted entries to the hardware devices causing the system to crash. Synchronization between the CU and OAM must have mutual exclusion, ie. the CU cannot try to update a line card record on the OAM if the OAM has sent a request to the CU to do something to that line card. This would cause the two copies of the database to be out of synch and could result in potential failure.

4 Data

4.1 General Types Used

General Typed Objects			
Purpose	This table presents the generic typed objects that are used in the system		
Object	**Contains/Extends**	**Purpose**	**Range**
CustId	Integer	An integer that each customer is associated with.	Infinite positive Integers
CustInfo	Strings, Integers	A structure that encapsulates any specific customer information such as Contact Name, Address, etc.	Any valid data that
ExchNum	Integer	An integer that each exchange is associated with.	[1..9]
LDID	Integer	An integer that each LD plan is associated with.	Inifinite postive Integers
Monetary-Value	Real	A real number that stores a moentary value of cost or credit.	Infinite Real Numbers
phoneNum	Integer	A three digit integer where the first digit	[000..999]
TimePeriod	Date	A structure that encapsulates two dates with specific times. The dates represent a start and end of a period.	Two Valid Dates with start <= end
Type OfCharge	Set	Predefined set that represent the various extentions of Charge	{LDCall, Service, Credit}

Table 59: General Typed Objects

4.2 Class Documentation

Actor	Purpose
BillDeamon	Represents a billing deamon which announces when the “Billing Day” occurs
CU	Represents the hardware CU
Operator	Represents the Human Operator
Printer	Represents the physical hardware printer

Table 60: Table of Actors

OAMConsole					
Purpose	Represents the OAMConsole that the operator uses				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	cID	CustID	A unique customer identifier in the OAM system	Any valid CustID	
	EX	ExchNum	A unique number for each exchange in the OAM system	Any valid ExchNum	
	LD	LDID	A unique number for the Long Distance plan	Any valid LDID	
Operations	**Name**	**Parameters**	**Pupose**	**Pre-condition**	**Post-condition**
	display	msg,args	Display some output	msg is a message for the operator the args contain context specific information	msg and args is displayed
	display-Failure	eq: Equip-Num	Displays a failure message to	-	The failure message

	login	userName: String, password: String	Authenicates the operator on the OAM system	userName, password are non-empty	The operator is permited to access the system if the username and and password are valid
	logout		Deauthenicates the operator on the OAM system	The operator associated with this console is logged in	The operator is denied further access to the system
	receiveInput	data	Receive some input from the operator	data is a valid	input is processed by the system

Table 62: OAM Class

Customer					
Purpose	Stores the information related to each customer				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	cID	custID	Stores the customer's unique identifier	Any valid custID	
	cInfo	CustInfo	Customer specific information	Any valid CustInfo	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	cancel	-	Cancel this customers account and related subscriptions	-	The customers subscriptions have been canceled

	create	cInfo: CustInfo	Creates a new customer account with the provided information	Information pretaining to the individual described in cInfo does not already exist in the system	new custID is allocated and information is assigned
	find	cInfo: CustInfo	Finds a customer given a partial set of customer information	cInfo must be valid, but need not be complete	a list of customers is returned that matches cInfo
	find	cID:custID	Finds a customer given a customer identifier	-	customer is returned that matches the cID
	findSub	ph: phoneNum	Finds a subscription that is mapped to the phone number	ph is valid phoneNum	subscription is returned that matches the ph
	newSub	ph: phoneNum	Creates a new subscription with the specified phone number	ph is valid phoneNum	if ph is a free phone number, then subscription is created otherwise an error is returned

	newSub	-	Creates a new subscription (phone number unspecified), the system will allocate a free phone number	-	if there are free numbers available in the exchange, then a subscription is created, otherwise an error is returned
	setCInfo	cInfo: CustInfo	Updates the customer's information	cInfo must be valid	The customers information has been updated

Table 64: Customer Class

Exchange					
Purpose	Represents a telephone exchange				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	localRate	Monetary-Value	Stores the rate at which local service will be billed	Any valid Monetary-Value	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	associate-Exchange	ex: Exch-Num	Allocates a trunk card and	ex exists	a trunk card is allocated
	billCall	EX, Shelf, Slot,	Bills call to the associated	-	the call is created
	disassociate-Exchange	ex: Exch-Num	Deallocates a trunk card and severes the connection with	ex exists	the trunk card is freed that has the DN equal to the ex specified

	findExchange	ex: ExchNum	Returns an instance of the exchange that matches the ExchNum requested	-	an exchange is returned iff the exchange number requested equals the exchange number of the exchange found, otherwise an error is returned
	setLocalRate	newRate:	Changes the rate at which	-	localRate is set to
	setLDRate	ex: ExchNum, newRate: MonetaryValue	The LD default rate to call the specified exchange is changed	ex is a valid	the associated LD Default rate is changed so that it reflects the new value specified

Table 66: Call Class

Bill					
Purpose	Represents a billing period for a customer's subscription				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	myCharges	Charge[s]	An explicit named attribute to the	Any valid Charges	
	mySub	Subscription	An explicit named attribute to the	Subscription owner of bill	
	period	TimePeriod	Stores the duration of the bill	Any valid TimePeriod	
	total	Monetary-Value	Stores the total monetary cost of the bill	Any valid Monetary-Value	
	totalLD	Monetary-Value	Stores the total monetary cost of the long distance charges associated with the bill	Any valid Monetary-Value	
	totalLocal	Monetary-Value	Stores the total monetary cost of the local charges associated with the bill	Any valid Monetary-Value	
	totalLocal-Minutes	Integer	Stores the total local minutes called	Non-negative Integers	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	findBill	time: TimePeriod	Finds a bill that is associated with the time period provided	time is a valid TimePeriod	if a bill exists associated with the time period, then it is returned, otherwise an error is returned
	send	-	Prints a bill to output device	-	bill is printed to output device immediately

Table 67: Bill Class

Current Bill					
Purpose	Represents the current bill, the one that charges on a subscription are applied to.				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	-	-	-	-	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	addCharge	type: TypeOfCharge, amt: MonetaryValue, start: Integer, duration: Integer, ph: phoneNum	Adds a charge to the current bill	-	A charge of the type specified is added to the bill
	calculate	-	Calculates the current amount of the bill	-	The total amount of the bill is updated.

Table 68: Current Bill Class

Charge					
Purpose	Represents a monetary charge on a bill				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	amount	Monetary-Value	Stores the monetary cost of the charge	Any valid Monetary-Value	
	description	String	Stores the textual description of the charge	Any valid String	
	ID	Integer	Unique id of charge, so it can be identified uniquely in the bill	Unique 1..n integers	
	period	TimePeriod	Stores the time and duration of the charge	Any valid TimePeriod	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 69: Charge Class

Service					
Purpose	General object that represents a service charge				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	-	-	-	-	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 70: Service Class

Credit					
Purpose	General object that represents a credit charge				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	-	-	-	-	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 71: Credit Class

LDCall					
Purpose	Represents a long distance phone call				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	-	-	-	-	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 72: Long Distance Call Class

DefaultLDRates					
Purpose	Represents the billing rate of a long distance phone call				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	rate	Monetary-Value	Stores the rate at which the associated exchange calling another exchange is billed at for long distance.	Any valid Monetary-Value	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 73: Long Distance Call Class

LDPlan					
Purpose	Represents a long distance phone call discount plan				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	discount	Real	Stores the percent discount of the plan	[-1..1]	
	period	TimePeriod	Stores the time period in which the plan in effect	Any valid TimePeriod	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	findLDPlan	LDID	Returns a LD Plan that is associated with the identifier passed in.	LDID is Valid	A LD Plan is returned iff the LDID matches the LD Plan found.

Table 74: Long Distance Plan Class

Subscription					
Purpose	Represents a long distance phone call				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	balance	Monetary-Value	Stores the oustanding balance on the subscription	Any valid Monetary-Value	
	billing-Address	String	Stores the billing address where the bill will be addressed	Any valid String that represents an Address	
	currentBill	Bill	An explicit reference to the current	Any valid Bill	
	myCOS	Class of Service	An explicit attribute that shows the ownership of the COS	Any valid COS	
	myBills	Bill	An explicit reference to the previous	Any valid Bill	
	myLDPlan	LDPlan	An explicit attribute that shows the reference to a LD Plan	Any valid LDPlan	
	myLineCard	LineCard	An explicit reference to the line card associated with the subscription.	Any valid LineCard	
	myPhone-Number	Phone-Number	An explicit reference to the phone number associated with this class	Any valid PhoneNumber	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**

	activate	-	Activate a subscription by acquiring a line card.	subscription has a PhoneNumber already allocated	susbscription is associated with a line card if room in exchange
	cancel	-	Cancels the customer's subscription	-	bill is printed and sent, line card is disassociated with subscription, phone number is reclaimed
	credit	amt: MonetaryValue, prd: TimePeriod	Credits a customer's bill with value	amt is a valid	a credit is added to the current bill
	findBill	prd: TimePeriod	Find the bill for th given TimePeriod	prd is a valid TimePeriod	A Bill for the specified period is returned if one exists; otherwise an error is returned
	findSub	ph: phoneNum	Finds a subscription that corresponds to the phoneNum specified	-	A Subscription is returned iff the subscription's phoneNum matches the one
	initialize	ph: phoneNum	Initializes a subscription with the	-	If the phone number is
	initialize	ex: ExchNum	Initializes a subscription in the	-	If the exchange specified

	payment	amt: Monetary-Value, prd: TimePeriod	Pays a customer's bill; in whole	value is a valid, positive Monetary-Value	a credit added to the current
	sendBill	-	Calculates and prints the current bill	-	Calculates and sends the
	setCOS	cos: Integer	Sets the class of service to be COS as specified in the cos parameter	cos represents a valid COS	myCOS is set to be the value of the parameter specified and the CU is updated
	setLDPlan	plan: LDID	Sets the LD plan for this subscription to be the one identified by the "plan" parameter	plan is a valid LDID	myLDPlan is set to reference the LD Plan associated with the LDID, plan
	suspend	-	Suspends a customer's subscription	subscription is	subscription is

Table 76: Subscription Class

ClassOfService					
Purpose	Stores the Class of Service on a subscriptionn				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	callLD	Boolean	Stores whether the Subscription has the ability to dial LD numbers	{TRUE, FALSE}	
	callLocal	Boolean	Stores whether the Subscription has the ability to dial local numbers	{TRUE, FALSE}	
	receiveCall	Boolean	Stores whether the Subscription has the ability to receive incoming calls	{TRUE, FALSE}	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 77: Class of Service Class

Phone Number					
Purpose	Represents a phone number in the OAM				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	inUse	Boolean	Stores whether the phone number is allocated to a subscription or not	{TRUE, FALSE}	
	PHN	phoneNum	Stores the phoneNum that the object is associated with	Any valid phoneNum	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	allocate	-	Allocates the Phone Number	Phone Number is	isUsed is set to TRUE
	findFree	exchange: Integer	Returns a Phone Number that is not in use in the exchange specified	exchange is a Integer between [1..9]	if there is a free phone number in the exchange then a phone number is returned
	find	ph: phoneNum	Returns a Phone Number that is associated with a specific phoneNum	ph is a valid phoneNum	returns the Phone Number associated with the given ph if it not in use; otherwise, and error is returned
	release	-	Releases the Phone Number	Phone Number is in use	isUsed attribute is set to FALSE

Table 78: Phone Number Class

Card					
Purpose	Represents a physical hardware device in the CU				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	DN	Integer	Stores the dialed number that this	0255	
	EqNumber	Integer	Stores a unique identifier of the	The concatenation of EX, Shelf and Slot	
	EX	Integer	Stores the exchange which the card is associated with	0255 See Hardware Interface section for specific meaning of values	
	inUse	Integer	Stores whether the card is currently allocated to service	0255 See Hardware Interface section for specific meaning of values	
	Shelf	Integer	Stores the shelf that the card is on in the CU	0255 See Hardware Interface section for sepcific meaning of values	
	Slot	Integer	Stores the slot in which the card is located	0255 See Hardware Interface section for specific meaning of values	
	Status	BitSet	Stores the current status of the	07 bits	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	disable	-	Sets the state of the card to be	The card is	The status on the
	enable	-	Sets the state of the card to be enabled	The card is currently disabled	The status on the card is set to enabled
	findCard	dn: DN	Finds a card that is associated with a dialed number.	-	a card is returned if the DN of the card found matches the DN passed in. NB: this is exchange specific

	findCard	EX, Shelf,	Returns a card that is associated with	-	Returns a card that has EqNumber equal to the value requested, otherwise a card is not returned
	findFree	ex:ExchNum	Finds a free card in the given exchange	ex is a valid Exchange number	Returns a free card if one exists; otherwise, an error is returned
	initialize	dn: DN	Returns the card to virgin status and sets the dialed number	-	The card is initalized to default values and the DN is set to dn
	queryDB	-	Returns the CU's copy of the information for this line card	-	Returns the EX, Shelf, Slot, STAT fields that the CU has for the device
	release	-	Releases a card that is currently allocated	card is currently allocated	inUse is set to false

	replace	-	Replaces a card.	card is currently allocated	the card is deallocated and a new card is allocated if one exists if one does not then an error is returned and action is not taken
	reset	-	Resets the card	-	Sends a reset message to the CU
	test	-	Tests the card	-	Requests that the CU run a test on this card
	updateDB	eq: EquipNum,	Updates the cards status that is	-	If the status has a

Table 80: Card Class

Trunk Card					
Purpose	Represents a trunk card that connects to a remote exchange				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	remoteExNum	Integer	Stores the number of the remote exchange	[1..9]	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 81: Trunk Card Class

Line Card					
Purpose	Represents a hardware device that connects to the customer's phone line				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	-	-	-	-	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	-	-	-	-	-

Table 82: Line Card Class

DebugConsole					
Purpose	Output device for debug messages				
Attributes	**Name**	**Type**	**Purpose**	**Range**	
	-	-	-	-	
Operations	**Name**	**Parameters**	**Purpose**	**Pre-condition**	**Post-condition**
	display	msg,args	Displays the message to the debug console. args contains context specific information	-	Data is written to the Operator's debug console window

Table 83: Debug Console Class

4.3 Event Documentation

4.3.1 Operator Events

findCustomer(cID)	
Purpose	Allows the Operator to find a customer given a CustID
Parameters	cID: CustID
Source	Operator
Destination	OAMConsole

Table 84: findCustomer(cID) Event

findCustomer(cInfo)	
Purpose	Allows the Operator to find list of customers that have the given information
Parameters	cInfo: CustInfo
Source	Operator
Destination	OAMConsole

Table 85: findCustomer(cInfo) Event

createCustomer(cInfo)	
Purpose	Allows the Operator to create a customer providing customer information.
Parameters	cInfo: CustInfo
Source	Operator
Destination	OAMConsole

Table 86: createCustomer(cInfo) Event

findExchange(ex)	
Purpose	Allows the Operator to find and access an exchange
Parameters	ex: ExchNum
Source	Operator
Destination	OAMConsole

Table 87: findExchange(ex) Event

findLDPlan(LDID)	
Purpose	Allows the Operator to find an existing LD Plan by a unique identifier for the plan.
Parameters	LDID: LDID
Source	Operator
Destination	OAMConsole

Table 88: findLDPlan(LDID) Event

createLDPlan(discount, period)	
Purpose	Allows the Operator to create a new LD Plan.
Parameters	discount: Integer, period: TimePeriod
Source	Operator
Destination	OAMConsole

Table 89: createLDPlan(discount, period) Event

findCard(eq)	
Purpose	Allows the Operator to find a hardware card.
Parameters	eq: EquipNum
Source	Operator
Destination	OAMConsole

Table 90: findCard(eq) Event

4.3.2 OAMConsole Events

cannotCreateCustomer(cInfo)	
Purpose	Returns an error to the Operator stating that the customer cannot be created.
Parameters	cInfo: CustInfo
Source	OAMConsole
Destination	Operator

Table 91: cannotCreateCustomer Event

OK(msg, args)	
Purpose	Returns the message (or object) and arguments to the Operator.
Parameters	msg, args: Data
Source	OAMConsole
Destination	Operator

Table 92: OK(msg, args) Event

custNotFound(cID)	
Purpose	Returns to the Operator a message stating that the requested customer could not be found.
Parameters	cID: CustID
Source	OAMConsole
Destination	Operator

Table 93: custNotFound(cID) Event

noMatchingCust(cInfo)	
Purpose	Returns to the Operator a message stating that no Customers were found that matched the customer information provided.
Parameters	cInfo: CustInfo
Source	OAMConsole
Destination	Operator

Table 94: noMatchingCust(cInfo) Event

cardNotFound(phn)	
Purpose	Returns to the Operator a message stating that the requested card could not be found.
Parameters	phn: phoneNum
Source	OAMConsole
Destination	Operator

Table 95: custNotFound(cID) Event

noMoreCards	
Purpose	Returns to the Operator a message stating that there are no more free cards and the operator requested cannot proceed.
Parameters	-
Source	OAMConsole
Destination	Operator

Table 96: noMoreCards Event

noFreePhoneNumber	
Purpose	Returns to the Operator a message stating that there are no more free phone numbers in the an exchange.
Parameters	-
Source	OAMConsole
Destination	Operator

Table 97: noFreePhoneNumber Event

planNotFound(LDID)	
Purpose	Returns to the Operator a message stating that the plan requested could not be found.
Parameters	LDID: LDID
Source	OAMConsole
Destination	Operator

Table 98: planNotFound(LDID) Event

exchangeNotFound(ex)	
Purpose	Returns to the Operator a message that exchange requested could not be found.
Parameters	ex: ExchNum
Source	OAMConsole
Destination	Operator

Table 99:exchangeNotFound(ex) Event

receiveInput(data)	
Purpose	Receives input comands (data) from the Operator.
Parameters	Data
Source	Operator
Destination	OAMConsole

Table 100: receiveInput(data) Event

display(msg,args)	
Purpose	Displays the msg and context specific information to the Operator console
Parameters	msg,args
Source	Bill, Card, Customer, Exchange, LDPlan, PhoneNumber, Subscription
Destination	OAMConsole

Table 101: display(data) Event

login(userName:String, password:String)	
Purpose	Authenicate the operator via a username and password.
Parameters	userName: String, password: String
Source	Operator
Destination	OAMConsole

Table 102: Login Event

logout()	
Purpose	Deauthenicate the operator that is associated with the console
Parameters	none
Source	Operator
Destination	OAMConsole

Table 103: Logout Event

displayFailure(eq: EquipNum)	
Purpose	Displays a failure message to the Operator's console
Parameters	eq: EquipNum
Source	Card
Destination	OAMConsole

Table 104: Event

4.3.3 Customer Events

create(info: CustInfo)	
Purpose	Creates a customer with the associated customer information
Parameters	info: CustInfo
Source	Operator/OAMConsole
Destination	Customer

Table 105: Add Customer Event

find(info: CustInfo)	
Purpose	Finds a list of customers who's informatiuon matches that given
Parameters	info: CustInfo
Source	Operator/OAMConsole
Destination	Customer

Table 106: Find Customer Event

find(cid: CustID)	
Purpose	Finds a customer with the given ID
Parameters	cid: CustID
Source	Operator/OAMConsole
Destination	Customer

Table 107: Find Customer Event

findSub(ph: phoneNum)	
Purpose	Finds a subscription belonging to the recipient that is mapped to the given phone number
Parameters	ph: phoneNum
Source	Operator/OAMConsole
Destination	Customer

Table 108: Find Subscription Event

newSub(ex: ExchNum)	
Purpose	Creates a new subscription for the recipient in the exchange specified. The phone number is assigned a random one that is not in use.
Parameters	ex: ExchNum
Source	Operator/OAMConsole
Destination	Customer

Table 109: New Subscription without Phone Num Event

newSub(ph: phoneNum)	
Purpose	Adds a subscription to the recipient in the exchange specified by the phone number with the DN specified in the phone number
Parameters	ph: phoneNum
Source	Operator/OAMConsole
Destination	Customer

Table 110: New Subscription with Phone Num Event

4.3.4 Exchange Events

changeLocalRate(rate: MonetaryValue)	
Purpose	Changes the local rate that the operator requested to change in that exchange. The new value will be reflected in bills in the next billing period
Parameters	rate: MonetaryValue
Source	Operator/OAMConsole
Destination	Exchange

Table 111: Change Local Rate Event

findExchange(ex: ExchNum)	
Purpose	Returns the exchange that is associated with the specified “ex” ExchNum.
Parameters	CustID
Source	Customer, Exchange, PhoneNumber, Susbscription
Destination	Exchange

Table 112: Find Exchange Event

billCall(EX, Shelf, Slot, Numcalled, Duration, TimeOfCall)	
Purpose	Bills a call to the appropriate subscription's current bill
Parameters	EX, Shelf, Slot, Numcalled, Duration, TimeOfCall
Source	CU
Destination	Exchange

Table 113: Bill Call Event

associateExchange(ex: ExchNum)	
Purpose	Associates the recipient exchange with another exchange by allocating a trunk card. ex is the exchange to be associated with.
Parameters	ex: ExchNum
Source	Operator/OAMConsole
Destination	Exchange

Table 114: Associate Exchange Event

disassociateExchange(ex: ExchNum)	
Purpose	Disassociates the recipient exchange with the exchange "ex" specified.
Parameters	ex: ExchNum
Source	Operator/OAMConsole
Destination	Exchange

Table 115: disassociate Exchange Event

findBill(time: TimePeriod)	
Purpose	Finds a bill given a time period.
Parameters	time: TimePeriod
Source	Operator/OAMConsole
Destination	Bill

Table 116: Find Bill Event

send()	
Purpose	Sends the recipient Bill to the printer.
Parameters	-
Source	Operator, Subscription, BillDeamon
Destination	Exchange

Table 117: Send Bill Event

4.3.5 CurrentBill Events

addCharge(type: TypeOfCharge, amt: MonetaryValue, start: Integer, duration: Integer, ph: phoneNum)	
Purpose	Adds a charge to the current bill
Parameters	type: TypeOfCharge, amt: MonetaryValue, start: Integer, duration: Integer, ph: phoneNum
Source	Subscription, Exchange
Destination	CurrentBill

Table 118: Add Charge Event

calculate()	
Purpose	Calculates the value of the current bill.
Parameters	-
Source	Subscription
Destination	CurrentBill

Table 119: Calculate Event

4.3.6 LDPlan Events

findLDPlan(LdId: LDID)	
Purpose	Finds a LDPlan given an associated LDID
Parameters	LdId: LDID
Source	Operator/OAMConsole
Destination	LDPlan

Table 120: FindLDPlan Event

createLDPlan(period: TimePeriod, discount: Real)	
Purpose	Creates a LD Plan with the period and discount specified
Parameters	period: TimePeriod, discount:Real
Source	Operator/OAMConsole
Destination	LDPlan

Table 121: Create LDPlan Event

removeLDPlan(LdId: LDID)	
Purpose	Removes a LD Plan from the system, and disassociates any connections related to it
Parameters	LdId: LDID
Source	Operator/OAMConsole
Destination	LDPlan

Table 122: Create LDPlan Event

4.3.7 Subscription Events

suspend()	
Purpose	Suspends a subscription, deallocating a line card that is allocated
Parameters	-
Source	Operator/OAMConsole
Destination	Subscription

Table 123: Suspend Subscription Event

activate()	
Purpose	Activates a subscription, allocating a line card and assigning the line card the associated PhoneNumber
Parameters	-
Source	Operator/OAMConsole
Destination	Subscription

Table 124: Activate Subscription Event

cancel()	
Purpose	Cancels a subscription, sending out a bill immediately.
Parameters	-
Source	Operator/OAMConsole
Destination	Subscription

Table 125: Cancel Subscription Event

payment(amt: MonetaryValue, prd:TimePeriod)	
Purpose	Adds a credit of size amt to the current bill.
Parameters	amt: MonetaryValue, prd:TimePeriod
Source	Operator/OAMConsole
Destination	Subscription

Table 126: Payment Event

credit(amt: MonetaryValue, prd Period)	
Purpose	Adds a credit of size amt to the current bill.
Parameters	amt: MonetaryValue, prd:TimePeriod
Source	Operator/OAMConsole
Destination	Subscription

Table 127: Credit Event

setCOS(cos: ClassOfService)	
Purpose	Sets the COS of the subscription to be the cos specified.
Parameters	cos: ClassofService
Source	Operator/OAMConsole
Destination	Subscription

Table 128: Set COS Event

setLDPlan(LdId: LDID)	
Purpose	Sets the LD Plan of the subscription to be the one referenced by the LdId parameter
Parameters	LdId: LDID
Source	Operator/OAMConsole
Destination	Subscription

Table 129: Set LD Plan Event

initialize(ph: phoneNum)	
Purpose	Initializes a subscription with the phoneNum provided. This will only be successful if the phoneNum is not currently in use, otherwise an error is returned
Parameters	ph: phoneNum
Source	Customer
Destination	Subscription

Table 130: Initialize with Phone Number Event

initialize(ex: ExchNum)	
Purpose	Initializes a subscription in the exchange provided. This will only be successful if there is a free phoneNum and line card available in the exchange, otherwise an error is returned
Parameters	ex: ExchNum
Source	Customer
Destination	Subscription

Table 131: Initialize with ExchNum Event

sendBill()	
Purpose	Sends the current bill to the printer and creates a new bill, moving the old one to the list of old bills.
Parameters	-
Source	BillDeamon, Operator
Destination	Subscription

Table 132: Initialize with ExchNum Event

findSub(phn: phoneNum)	
Purpose	Finds a subscription given a phone number
Parameters	ph: phoneNum
Source	Customer, Operator
Destination	Subscription

Table 133: Find Subscription Event

4.3.8 PhoneNumber Events

findFree(ex: ExchNum)	
Purpose	Finds a free phone number given an exchange to look in
Parameters	ex: ExchNum
Source	Subscription
Destination	PhoneNumber

Table 134: Find Free Phone Number Event

find(ph: phoneNum)	
Purpose	Find the PhoneNumber corresponding to the given number
Parameters	ph: phoneNum
Source	Exchange, Subscription
Destination	PhoneNumber

Table 135: Find Phone Number Event

allocate()	
Purpose	Allocates the PhoneNumber, setting the inUse Flag to TRUE
Parameters	-
Source	Subscription
Destination	PhoneNumber

Table 136: Allocate Phone Number Event

release()	
Purpose	Deallocates the PhoneNumber, setting the inUse Flag to FALSE
Parameters	-
Source	Subscription
Destination	PhoneNumber

Table 137: Release Phone Number Event

4.3.9 Card Events

enable()	
Purpose	Enables the card, setting the status bits to active
Parameters	-
Source	Exchange
Destination	Card

Table 138: Enable Card Event

disable()	
Purpose	Disables the card, setting the status bits to inactive
Parameters	-
Source	Exchange
Destination	Card

Table 139: Disable Card Event

reset()	
Purpose	Sends a reset message to the CU
Parameters	-
Source	Exchange
Destination	Card

Table 140: Reset Card Event

initialize(dn: DN)	
Purpose	Initializes the card to a default state and sets the DN to dn
Parameters	dn: DN
Source	Exchange, Subscription
Destination	Card

Table 141: Inititalize Card Event

findCard(EX: Integer, Shelf: Integer, Slot: Integer)	
Purpose	Finds a card that is associated with the equipment number that is specified in the three parameters
Parameters	EX: Integer, Shelf: Integer, Slot: Integer
Source	CU, Card, Exchange
Destination	Card

Table 142: Find Card Event

updateDB(eqNum: EquipNum, status:Integer)	
Purpose	Updates the card's status bits for the associated card that the eqNum specifies
Parameters	eqNum:EquipNum, status:Integer
Source	CU
Destination	Card

Table 143: UpdateDB Event

queryDB()	
Purpose	Returns the CU's copy of the data that is held for this card
Parameters	-
Source	DebugConsole
Destination	Card

Table 144: QueryDB Event

release()	
Purpose	Releases a card and sets it as free
Parameters	-
Source	Subscription, Exchange
Destination	Card

Table 145: Release Card Event

replace()	
Purpose	Replaces a line card by disabling the card and then allocating another one results in an error if there are noe free cards to allocate
Parameters	-
Source	Exchange
Destination	Card

Table 146: Replace Card Event

findCard(dn: DN)	
Purpose	Finds a card in an exchange with the DN specified
Parameters	-
Source	Exchange
Destination	Card

Table 147: Find Card (from exchange) Event

4.4 State Diagram Activity Documentation

4.4.1 Bill Activities

Find Bill	
Purpose	Finds a bill that matches the given period
Parameters	prd: TimePeriod
Initiator	Bill
Target	OAMConsole

Table 148: Find Bill Activity

Send Bill	
Purpose	Sends the bill to the printer
Parameters	prd: TimePeriod
Initiator	Bill
Target	Printer

Table 149: Send Bill Activity

4.4.2 Card Activities

Initialize Card	
Purpose	Sets the DN to the dn specifed, Updates the CU by UpdateDB, sets the card inUse
Parameters	dn: DN
Initiator	Card
Target	CU, Card

Table 150: Initialize Card Activity

Release Card	
Purpose	Sets the DN to the 99, Updates the CU by UpdateDB, sets the card inUse to FALSE
Parameters	-
Initiator	Card
Target	CU, Card

Table 151: Release Card Activity

Testing Card	
Purpose	Sends a request for a test to the CU. Note this is an asynchronous message, if an error is determined on the hardware, the result will be returned via an UpdateDB message.
Parameters	-
Initiator	Card
Target	CU

Table 152: Testing Card Activity

Disabling Card	
Purpose	Sends a message to the CU to disable the card, if it is inUse, it does a replace on the card. Sends a setStatus message to CU with the active status bit off.
Parameters	-
Initiator	Card
Target	CU

Table 153: Disabling Card Activity

Enabling Card	
Purpose	Sends a setStatus message to the CU, with the active status bit on
Parameters	-
Initiator	Card
Target	CU

Table 154: Enabling Card Activity

Reset Card	
Purpose	Sends a reset message to the CU.
Parameters	-
Initiator	Card
Target	CU

Table 155: Reset Card Activity

Find Card	
Purpose	Finds a card given a unique EquipNum. If the card is found OK with the instance is returned, otherwise NOK is returned.
Parameters	eq: EquipNum
Initiator	Card, CU
Target	Exchange, Subscription

Table 156: Find Card Activity

Query Card	
Purpose	Sends a queryDB message to the CU,and awaits a response from the CU. It will timeout if it does not hear back from the CU after a specified time and it will return with a NOK. Invalid entries or valid daa can be returned to the DebugConsole
Parameters	-
Initiator	Card
Target	CU, DebugConsole

Table 157: QueryDB Card Activity

Set Status Card	
Purpose	Sets the status after receiving a UpdateDB message from the CU. A display failure message could be generated if the hardware failed tests.
Parameters	eq: EquipNum, status: Integer
Initiator	CU
Target	Card, OAMConsole (on failure)

Table 158: Set Status Activity

4.4.3 Line Card Activities

Replace Card LineCard	
Purpose	Finds a free line card in the exchange, then initializes the card, setting the DN to the DN of this card. Assigns the subscription the new line card, and releases this card.
Parameters	-
Initiator	Operator
Target	LineCard, Subscription

Table 159: Replace Line CardActivity

4.4.4 Trunk Card Activities

Replace Card TrunkCard	
Purpose	Finds a free trunk card in the exchange, then initializes the card, setting the DN to the DN of this card. Then it releases this card.
Parameters	-
Initiator	Operator
Target	TrunkCard

Table 160: Replace Line CardActivity

4.4.5 Current Bill Activities

Calculate Period Current Bill	
Purpose	Creates a time period given a start time and duration.
Parameters	startTime: Integer, duration: Integer
Initiator	CU
Target	CurrentBill

Table 161: Calculate Period Activity

Service Current Bill	
Purpose	Creates a service charge given an amount, period and description and appends it to the current charges on the bill
Parameters	amnt: Integer, period: TimePeriod, description: String
Initiator	CurrentBill
Target	CurrentBill

Table 162: Service Activity

Credit Current Bill	
Purpose	Creates a credit given an amount, period and description and appends it to the current charges on the bill
Parameters	amnt: Integer, period: TimePeriod, description: String
Initiator	CurrentBill
Target	CurrentBill

Table 163: Credit Activity

LDCall Current Bill	
Purpose	Creates a LDCall given an amount, period and phoneNum and appends it to the current charges on the bill
Parameters	amnt: Integer, period: TimePeriod, phoneNum: String
Initiator	CurrentBill
Target	CurrentBill

Table 164: LDCall Activity

Calculate Totals Current Bill	
Purpose	Calculates the totals for LD and local service up to and including the current day.
Parameters	-
Initiator	Subscription
Target	CurrentBill

Table 165: Calculate Totals Activity

4.4.6 Customer Activities

Cancel Account Customer	
Purpose	Cancels all subscriptions related to the customer account
Parameters	-
Initiator	Operator
Target	Subscription

Table 166: Cancel Account Activity

Edit Customer	
Purpose	Updates the cInfo for the customer. cInfo will be set to newInfo
Parameters	newInfo: CustInfo
Initiator	Operator
Target	Customer

Table 167: Edit Customer Activity

Create Customer	
Purpose	Creates a new Customer given the cInfo
Parameters	cInfo: CustInfo
Initiator	Operator
Target	Customer

Table 168: Create Customer Activity

Query for Cust Customer	
Purpose	Give some cInfo, a list of matches of cInfo of customers is created and returned.
Parameters	cInfo: CustInfo
Initiator	Operator
Target	Operator

Table 169: Query For Cust Activity

Find Cust Customer	
Purpose	Finds a customer given a CustId, the cID of the found customer must match the cID passed in
Parameters	cID: CustId
Initiator	Operator
Target	Operator, Customer

Table 170: Find Customer Activity

FindSub Customer	
Purpose	Finds a subscription given a phoneNum
Parameters	ph: phoneNum
Initiator	Operator
Target	Operator, Customer

Table 171: FindSub Activity

NewSub Customer	
Purpose	Creates a new subscription given a phoneNum. It will create it only if the phoneNum is free.
Parameters	ph: phoneNum
Initiator	Operator
Target	Customer, Subscription

Table 172: NewSub Activity

NewSub #2 Customer	
Purpose	Creates a new subscription given an exchange. It will create a subscription only if there is room in the exchange
Parameters	ex: ExchNum
Initiator	Operator
Target	Customer, Subscription

Table 173: NewSub #2 Activity

4.4.7 Exchange Activities

FindExchange Exchange Bill	
Purpose	Finds an exchange given an ExchNum, NOK is returned if the exchange is not found.
Parameters	ex: ExchNum
Initiator	Exchange
Target	Exchange

Table 174: FindExchange Activity

FindLineCard Exchange Bill	
Purpose	Finds a line card by asking the LineCard to find it. NOK is returned if the line card is not found
Parameters	eq: EquipNum: ex: Exchange
Initiator	Exchange
Target	LineCard

Table 175: FindLineCard Activity

FindSub Exchange Bill	
Purpose	Finds a subscription by asking the Subscription to find it. NOK is returned if the subscription cannot be found.
Parameters	eq: EquipNum, ln: LineCard
Initiator	Exchange
Target	Subscription

Table 176: FindSub Activity

GetBill Exchange Bill	
Purpose	Retrieves the current bill on the subscription
Parameters	sub: Subscription
Initiator	Exchange
Target	Subscription

Table 177: GetBill Activity

IsLocal Exchange Bill	
Purpose	Determines if the called number is a local one, given an exchange.
Parameters	eq: EquipNum
Initiator	Exchange
Target	Exchange

Table 178: IsLocal Activity

IncreaseLocalMinutes Exchange Bill	
Purpose	Increments the number of local minutes that are on a bill.
Parameters	bill: Bill, duration: Integer
Initiator	Exchange
Target	Exchange, Bill

Table 179: IncreaseLocalMinutes Activity

LDCall Exchange Bill	
Purpose	Retrieves the information on the LD Rates for the exchange that was called. Also retrives the customer's LD calling Plan.
Parameters	S: Subscription
Initiator	Exchange
Target	Exchange, Subscription

Table 180: LDCall Exchange Activity

Bill LDCall Exchange Bill	
Purpose	Bills the LD Call to the bill provided. This is accomplished by calling AddCharge on the CurrentBill.
Parameters	S: Subscription, B: CurrentBill
Initiator	Exchange
Target	Exchange, CurrentBill

Table 181: Bill LD Call Exchange Activity

Validate Exchange Number Exchange Associations	
Purpose	Validates the exchange number by finding the Exchange that is associated with the ex: ExchNum requested.
Parameters	ex: ExchNum
Initiator	Exchange
Target	Exchange

Table 182: Validate Exchange Number Exchange Activity

Find Trunk Card Exchange Associations	
Purpose	Finds the trunk card with the associated remote exchange.
Parameters	ex: ExchNum
Initiator	Exchange
Target	Exchange

Table 183: Find Trunk Card Exchange Activity

Remove Association Exchange Associations	
Purpose	Deallocates the Trunk Card.
Parameters	T: TrunkCard
Initiator	Exchange
Target	Exchange

Table 184: Remove Association Exchange Activity

Allocate Trunk Card Exchange Associations	
Purpose	Allocates a Trunk Card.
Parameters	-
Initiator	Exchange
Target	Exchange

Table 185: Allocate Trunk Card Exchange Activity

Create Association Exchange Associations	
Purpose	Initializes the Trunk Card with the remote exchange number.
Parameters	T: TrunkCard, ex: ExchNum
Initiator	Exchange
Target	Exchange

Table 186: Create Association Exchange Activity

Set Local Rate Exchange Misc.	
Purpose	Sets the local rate attribute on the Exchange.
Parameters	newRate: Integer
Initiator	Exchange
Target	Exchange

Table 187: Set Local Rate Exchange Activity

SetLDRate Exchange Misc.	
Purpose	Sets the LD rate for calls to the remote exchange from this Exchange.
Parameters	newRate: Integer, remoteExchange: ExchNum
Initiator	Exchange
Target	Exchange

Table 188: Set LD Rate Exchange Activity

Acknowledge CU online Exchange Misc.	
Purpose	Acknowledges that a CU has come online correctly by checking that the given ExchNum is valid.
Parameters	ex: ExchNum
Initiator	Exchange
Target	Exchange

Table 189: Acknowledge CU online Exchange Activity

Find Exchange Misc.	
Purpose	Finds an exchange given an ExchNum.
Parameters	ex: ExchNum
Initiator	Exchange
Target	Exchange

Table 190: Find Exchange Activity

4.4.8 Subscription Activities

ReleaseLineCard Subscription	
Purpose	Releases the line card that is associated with the subscription
Parameters	-
Initiator	Subscription
Target	LineCard

Table 191: Release Line Card Subscription Activity

Cancel Subscription	
Purpose	Cancels a subscription, releasing the phoneNumber and lineCard
Parameters	-
Initiator	Subscription
Target	PhoneNumber, LineCard

Table 192: Cancel Subscription Activity

Suspend Subscription	
Purpose	Suspends a subscription, releasing the lineCard
Parameters	-
Initiator	Subscription
Target	LineCard

Table 193: Suspend Subscription Activity

Allocate Line Card Subscription	
Purpose	Allocates a line card and sets it as the attrribute myLineCard
Parameters	-
Initiator	Subscription
Target	LineCard

Table 194: Allocate Line Card Subscription Activity

Associate Line Card Subscription	
Purpose	Initializes a given line card to this subscription by setting the DN
Parameters	lc: LineCard
Initiator	Subscription
Target	LineCard

Table 195: Associate Line Card Subscription Activity

Find PhoneNumber Subscription	
Purpose	Finds a PhoneNumber given a phoneNum
Parameters	ph: phoneNum
Initiator	Subscription
Target	PhoneNumber

Table 196: Find PhoneNumber Subscription Activity

Allocate PhoneNumber Subscription	
Purpose	Allocates a PhoneNumber given a PhoneNumber
Parameters	ph: PhoneNumber
Initiator	Subscription
Target	PhoneNumber

Table 197: Allocate PhoneNumber Subscription Activity

Find FreeNumber Subscription	
Purpose	Finds a free PhoneNumber given a phoneNumber
Parameters	ph: phoneNumber
Initiator	Subscription
Target	PhoneNumber

Table 198: Find FreeNumber Subscription Activity

Init Subscription	
Purpose	Initializes the subscription by setting myPhoneNumber to the PhoneNumber provided
Parameters	ph: PhoneNumber
Initiator	Subscription
Target	Subscription

Table 199: Find FreeNumber Subscription Activity

Change LDPlan Subscription	
Purpose	Sets myLDPlan to the LDPlanID provided
Parameters	ldPln: LDPlanID
Initiator	Subscription
Target	Subscription

Table 200: Change LDPlan Subscription Activity

ChangeCOS Subscription	
Purpose	Sets myCOS to the given COS, also sends an updateDB message to the CU to state the line card has change
Parameters	cos: Integer
Initiator	Subscription
Target	Subscription, CU

Table 201: ChangeCOS Subscription Activity

Find Subscription	
Purpose	Finds a subscription given a phoneNumber. If a subscription cannot be found then NOK is returned.
Parameters	ph: phoneNumber
Initiator	Subscription, Customer
Target	Subscription, Customer

Table 202: Find Subscription Activity

FindBill Subscription	
Purpose	Finds a bill that has the specified time period.
Parameters	time: TimePeriod
Initiator	Subscription, Customer
Target	Subscription, Customer

Table 203: FindBill Subscription Activity

Credit Subscription	
Purpose	Credits the current bill with the charge that is specified.
Parameters	amount: MonetaryValue, time: TimePeriod
Initiator	Operator
Target	CurrentBill

Table 204: Credit Subscription Activity

Payment Subscription	
Purpose	Credits a current bill with the payment that is specified.
Parameters	amount: MonetaryValue, time: TimePeriod
Initiator	Operator
Target	CurrentBill

Table 205: Payment Subscription Activity

Charge For Service Subscription	
Purpose	Calculates the current bill, by summing the totals and if need be prorates the . current monthly charge.
Parameters	-
Initiator	Operator, BillDaemon
Target	Subscription, CurrentBill

Table 206: Change For Service Subscription Activity

Send and Store Current Bill Subscription	
Purpose	Sends the recently calculated current bill to the printer and moves it into . the list of previous bills.
Parameters	-
Initiator	Subscription
Target	Subscription, Printer

Table 207: Send and Store Current Bill Subscription Activity

4.4.9 OAMConsole Activities

Await Login OAMConsole	
Purpose	Await an input of username and password from the Operator
Parameters	-
Initiator	Operator
Target	OAMConsole

Table 208: Await Login Activity

Validate OAMConsole	
Purpose	Validate the username and password returning an error to the operator, or allow the operator to issue commands henceforth.
Parameters	uName: String, pWord: String
Initiator	Operator
Target	OAMConsole

Table 209: Validate Login Activity

4.4.10 Find PhoneNumber Activity

FindFree PhoneNumber	
Purpose	Finds an unused phone number in the exchange. If there are no numbers left in the exchange then NOK is returned
Parameters	-
Initiator	PhoneNumber
Target	Subscription

Table 210: Find Free PhoneNumber Activity

Validate Exchange PhoneNumber	
Purpose	Checks to see if an exchange is valid given a ExchNum. If it is not NOK is returned
Parameters	ex: ExchNum
Initiator	PhoneNumber
Target	Subscription

Table 211: Validate Exchange PhoneNumber Activity

Find PhoneNumber	
Purpose	Finds a phoneNumber given a phoneNum. A PhoneNumber is returned if the phone number of the found entry matches the phoneNum, otherwise NOK is returned.
Parameters	ph: phoneNum
Initiator	Subscription
Target	PhoneNumber

Table 212: Find PhoneNumber Activity

Release PhoneNumber	
Purpose	Releases a PhoneNumber by setting inUse to FALSE
Parameters	ph: phoneNum
Initiator	Subscription
Target	PhoneNumber

Table 213: Release PhoneNumber Activity

Allocate PhoneNumber	
Purpose	Allocates a PhoneNumber by setting inUse to TRUE. NOK is returned if the PhoneNumber is already inUse.
Parameters	ph: phoneNum
Initiator	Subscription
Target	Subscription, PhoneNumber

Table 214: Allocate PhoneNumber Activity

5 Customer Sessions

5.1 Summary of Customer session #1, 2:

Note the customer session is marked as 1 & 2, however this was a double session.

5.1.1 PLANS:

Q: Can a customer have more than one long distance plan active at once?
A: No. There is a default plan which specific billing rates for calling other exchanges. A customer may also have 1 discount plan, which amount to a percentage discount off of all long distance calls. E.g “20% off long distance”

Q: What happens when a call spans 2 or more calling plan times?
A: Calls are charged based on the discounts in effect at the start of the call. That is, charges are based on start time.

Q: Can a long distance plan be removed (even if customers are still subscribed to it)? If this is allowed how do we handle it?
A: Yes. All customers who were subscribed to the cancelled plan will be changed based on the rates specified in the default plan.

Q: Are long distance rates the same for each outside exchange, or do they vary from exchange to exchange?
A: No, they can vary. Long distance billing rates for an exchange are specified on that exchange’s Default long distance plan the local rate may also be different for each exchange.

5.1.2 OPERATOR:

Q: Are there different levels of operator (normal, supervisor, etc)?
A: No.

Q: Do operators have to log-on to the system?
A: Yes. This is to prevent unqualified personnel from accessing the system. E.g. The Janitor can not change his/her billing information

5.1.3 GENERAL:

Q: What is the initial COS state?
A: Line can do everything (i.e. send local and long-distance calls, receive calls

Q: Can a customer have a number in their local exchange and a second line in another exchange? (And have the bill mailed to their local address) i.e. do we support a “Mailing Address” for each subscription?

A: Yes

Q: Are line cards 11 for each location? What happens when a customer moves and wants to keep the same phone number?

A: line cards can connect to any phone in the exchange. Model the above situation as a cancelled subscription followed by a request for a new line with a number specified. Note: There is no fee for moving (i.e. reassociating a phone number to a different line card)

Q: For line card tests, what sort of user interface is desired? E.g. “Test all” “Test Some” or “Test One” card.

A: One or all is adequate.

Q: Do we have to consider time zones? E.g. Customer in one, Exchange in another.

A: No.

Q: If CU tells us that a call was complete, we assume the call was completed properly. E.g. the call was not disconnected because of a faulty line.

A: OK

Q: Should we gear the GUI towards a specific platform? Can/Should it be web based?

A: Create a window-based application, e.g. use dialog boxes and menus. Note: The operator can view 1 set of information at a time. E.g. the operator can not view 2 customers’ information at the same time

.

5.1.4 BILLING:

Q: Is each customer’s bill sent on one “Billing Day”, or does each customer have their own billing date?

A: Each customer in each exchange is billed on the same day.

Q: Is the “flat-rate” charge for local calls on a “per call” or “per month” basis? E.g. 10 cents for each local call, or $20 for local calling this month.

A: per month. Note: We do not need to track local calls; we only need to count them. decide to track local calls then the information complied should not be visible to the customer.

Q: For overdue accounts, what kind of interest is accumulated (if any)?

A: No interest is charged.

Q: Does a customer with 2 or more lines receive one consolidated bill?

A: No. One bill per line.

Q: If an account is overdue for “too long”, what happens? Do we suspend the customers account (i.e. suspend all his/her lines) or just the line associated with of the overdue subscription?

A: If an account is overdue for more than 3 consecutive months the overdue line is suspended (not the customers account).

Q: Can a customer's account be cancelled for any reason?
A: No.

Q: How are billing errors handled? Does the operator change/remove/credit charges?
A: Credit. Do not remove the error.

Q: Are there different classes of customers? E.g. corporate, residential, etc?
A: No.

Q: Are there different types of suspend? (E.g. Overdue account, vacation, etc.) Are there charges for a vacation suspend?
A: 1 type. An operator can suspend an account for any reason. No additional charges accumulate while the line is suspended

5.1.5 ADDITIONAL NOTES:

1. No activation fee when registering a new line
2. Two (or more) phone numbers can not be routed to the same phone
3. The Operator can view all account information and filter the accounts based on "any" field. E.g. list of suspended accounts
4. Create a sequence diagram for:
    - Moving within the scope of the same exchange customer is guaranteed to get a subscription).
    - Moving outside of the exchange (no guarantee, i.e. exchange may be full).
5. An operator should be able to view the status of all cards in the exchange at once (E.g. A grid of line cards. Each cell represents a card. Cell is green(OK), yellow(TESTING) or red(DISABLED) depending on card status)
6. Customer Bill Format:
    (a) Customer info
    (b) Previous month's balance, amount paid, new balance
    (c) Monthly charges
    (d) Summary of local calls (e.g. number of local call made)
    (e) Summary of long distance charges -List of all calls with start time, duration cost, etc. -Subtotal
    (f) Total of all charges (e.g. "Amount Owing")

5.2 Email from Customer

- Operator should be able to view current and past bills for any account at any time. The current charge should include all long distance charges since last billing date to today. The local charge should be a fraction of the monthly rate. (monthly rate * # of days / 31)
- Bills should be sent to printer (modeled as another external actor) for printing on billing day, when customer want to close an account and anytime on operator's demand.
- I know this new requirement is something different from what I said during customer session. But some groups brought it up today and I want to standardize it cross all groups. Sorry for groups want early submission, but I don't think this will add too much extra work.

5.3 Summary of Customer Session #3

Here is the Customer Session Summary from Nov 22nd, 4:00pm

Q: Does billing require an accounting department actor? Or should the operator issue the bills on a regular interval?
A: The billing requests should be handled by a deamon. Yes, the deamon can be modelled as an actor if you wish.

Q: How does the system make new bills, ie. when the billing period ends?
A: When the deamon says to print a bill, then a new clean bill should be created and the old one stored.

Q: Do LD Plans have time restrictions?
A: Yes, they have time restrictions. That is they LD Plans have a time which it is in effect, like 7pm to 7am.

Q: Can the operator change default LD Plan Rates?
A: Test there should be 72 LD Rates in total, and the operator can change any of them.

Q: Can the operator change the "Billing Date"?
A: Yes, the operator can change the billing date. Note: the date is the same for all customers/exchanges.

Q: Can the operator ask for bill reprint?
A: Yes the operator can view any previous bills and print any of them.

Q: Can trunk cards be change?
A: The operator can change trunk cards in an exchange, ie. assigning it a new value to a different exchange. Note trunk cards can be tested, enabled and disable much like lien cards.

6 Traceability Matrix

Req #	Description	Sources	SRS References	Related Req #s
1	Add a customer	SRS Outline	Section 3.2.1 Page 61 Section 3.2 Page 53 Section 3.1.1 Page 12	2, 3, 4, 36
2	Modify Customer	SRS Outline	Section 3.2 Page 53 Section 3.2.1 Page 61 Section 3.1.1 Page 13	3, 4, 36
3	Cancel Customer	SRS Outline	Section 3.2 Page 53 Section 3.1.1 Page 18	2, 4, 36
4	View Customer Information	SRS Outline	Section 3.2 Page 53 Section 3.1.1 Page 16	36
5	Announce Failed Device	SRS Outline	Section 3.2 Page 58	
6	Trunk card init, change, free	SRS Outline Partial SRS Marking Scheme	Section 3.2 Page 53	
7	Update Device S-tatus	SRS Outline	Section 3.2 Page 58 Section 3.2.2 Page 75	12, 13
8	View bill	Customer Ses-sion SRS Outline	Section 3.2 Page 55 Section 3.1.1 Page 38	4, 9
9	Print bill	SRS Outline	Section 3.2	4, 8

			Page 55 Section 3.1.1 Page 38	
10	Cancel subscription	SRS Outline	Section 3.2 Page 53 Section 3.2.1 Page 61 Section 3.1.1 Page 18 Section 3.1.1 Page 13 Section 3.1.1 Page 31	16, 17
11	Change subscription's LD Plan A Subscription has only one LD Plan associated with it	SRS Outline Customer Session	Section 3.2 Page 55 Section 3.2.1 Page 61 Section 3.1.1 Page 41	22, 23
12	UpdateDB from CU	SRS Outline	Section 3.2.2 Page 74	13
13	UpdateDB to CU	SRS Outline	Section 3.2.2 Page 74	12
14	Operator must login with username, password	Customer Session	Section 3.2.1 Page 61 Section 3.1.1 Page 9 Section 2.5 Page 4	
15	Data must not be destroyed, all subscriptions bills, customer records will be maintained after they are terminated.	SRS Outline	Section 3.2.1 Page 61	

16	Cancelling a subscription will cause a bill to be sent immediately and the monthly service to be prorated for the month	Customer Email	Section 3.2 Page 53 Section 3.1.1 Page 18 Section 3.1.1 Page 13 Section 2.5 Page 5	10, 17
17	Suspend subscription	SRS Outline	Section 3.2 Page 53 Section 3.2.1 Page 61 Section 3.1.1 Page 31	18
18	Resume subscription	SRS Outline	Section 3.2 Page 53 Section 3.2.1 Page 61 Section 3.1.1 Page 31	17
19	Bill Payment	SRS Outline	Section 3.2 Page 55 Section 3.1.1 Page 38	
20	Bill Credit	Customer Session	Section 3.2 Page 55 Section 3.1.1 Page 38	
21	Add Subscription with either a specific phone number or an exchange numbe	Customer Session SRS Outline	Section 3.2.1 Page 61 Section 3.2	22, 23, 24

			Page 53 Section 3.2.1 Page 61	
22	Edit Subscription	SRS Outline	Section 3.2 Page 53 Section 3.1.1 Page 27	36
23	View Subscription	SRS Outline	Section 3.2 Page 53 Section 3.1.1 Page 25	36
24	Bills are preiodically sent to the printer	Customer Session	Section 3.2.1 Page 61 Section 3.2.1 Page 61	34
25	Change billing rates local for an exchange	Customer Session	Section 3.2 Page 55 Section 3.1.1 Page 42	
26	Change billing rates LD for an exchange each has their own unique mapping to other exchanges for rates	Customer Session	Section 3.2 Page 55 Section 3.1.1 Page 42	
27	Add Global LD Plan	SRS Outline	Section 3.2.1 Page 62 Section 3.2 Page 55 Section 3.1.1 Page 42	28, 29
28	Edit Global LD Plan	SRS Outline	Section 3.2.1 Page 62 Section 3.2 Page 55	27, 29

			Section 3.1.1 Page 42	
29	Remove Global LD Plan	SRS Outline	Section 3.2.1 Page 62 Section 3.2 Page 55 Section 3.1.1 Page 42	27, 28
30	Billing Rates each exchange has their own LD and local rates	Customer Session	Section 3.2 Page 55 Section 3.2.1 Page 62	25, 26
31	Billing Rates LD plan is only a discount off of the regular rates	Customer Session	Section 3.2.1 Page 62	30
32	Check that LD plans do not overlap	Not required as per Customer Session		
33	Bill Calls -intra-exchange phone numbers are charged a fixed rate, outside the exchange calls are billed at a rate specified by the exchange. LD Calls need to be listed seperately.	SRS Outline	Section 3.2 Page 55 Section 3.2.1 Page 62 Section 3.2.1 Page 62	
34	Change billing dates, all customers / subscriptions will have the same billing day.	Customer Session	Section 3.2.1 Page 61	

			Section 3.2 Page 55 Section 3.1.1 Page 42	
35	Supsend Customer	SRS Outline	Section 3.2 Page 53 Section 3.1.1 Page 18	
36	Find a Customer given information A list of matching customers is returned if one and only one cannot be found	SRS Outline	Section 3.2 Page 53 Section 3.2 Page 55 Section 3.2.1 Page 61 Section 3.1.1 Page 13 Section 3.1.1 Page 16 Section 3.1.1 Page 25	1, 2, 3, 4
37	Debug commands, display debugging	SRS Outline	Section 3.2 Page 58	
38	Find a Subscription given information	SRS Outline	Section 3.2 Page 53 Section 3.2 Page 55	22, 23
39	Card enable	SRS Outline	Section 3.2 Page 58 Section 3.2.1 Page 62 Section 3.1.1 Page 32	
40	Card disable	SRS Outline	Section 3.2 Page 58	

			Section 3.2.1 Page 62 Section 3.1.1 Page 32	
41	Card test	SRS Outline	Section 3.2 Page 58 Section 3.2.1 Page 62 Section 3.1.1 Page 32	
42	Card reset	SRS Outline	Section 3.2 Page 58 Section 3.2.1 Page 62 Section 3.1.1 Page 32	
43	QueryDB messages to be sent to CU	SRS Outline	Section 3.2.2 Page 75	
44	InvalidDBEntry messages received from CU	SRS Outline	Section 3.2.2 Page 75	
45	ContentsDB messages received from CU	SRS Outline	Section 3.2.2 Page 75	
46	QueryDD messages, and responses (InvalidDBEntry, ContentsDB) are synchronous events.	Customer Session	Section 3.2.2 Page 75	
47	TestEN messages sent to CU	SRS Outline	Section 3.2.2 Page 74	
48	The initial COS is set to “all” when a	Customer Session	Section 3.1.1	

	new subscription is created		Page 21	
49	The machine on which the OAM software runs, will be capable of handling the requests of all CUs, and the console, in an efficient and timely manner	SRS Outline	Section 2.5 Page 4	
50	Phone numbers are unique	SRS Outline	Section 2.5 Page 4	
51	Each phone has at most one phone number mapped to it	Customer Session	Section 2.5 Page 4	
52	A unique identification number (CustID) will be provided for each customer	SRS Outline	Section 2.5 Page 4	
53	A suspended subscription will not maintain an association with a line card	Customer Session	Section 2.5 Page 4	
54	A suspended subscription will maintain an association with a phone number	Customer Session	Section 2.5 Page 4	
55	The GUI will be dialog modal based, and the operator can only view one screen at a time	Customer Session	Section 2.5 Page 4	

56	System integrity will be maintained, regardless of environmental interference (eg: hardware failure will not corrupt system)	TA Session	Section 2.5 Page 4	
57	There is only one operator using the system at any given time	Customer Session	Section 2.5 Page 4	
58	Local calls are billed at a fixed rate per month	Customer Session	Section 2.5 Page 4	
59	Each phone line has a unique bill	Customer Session	Section 2.5 Page 4	

Index