ECSE321

Introduction to Software Engineering

ACE

Low-Level Design

Team 6
Michael Vizbara
Gabriel Lemonde-Labrecque
Kirill Selikov
Alexandru Ciobanu
Ionut Georgian Ciobanu

November 20, 2007

Table of content

1. Static Model

1.1. Package Diagram

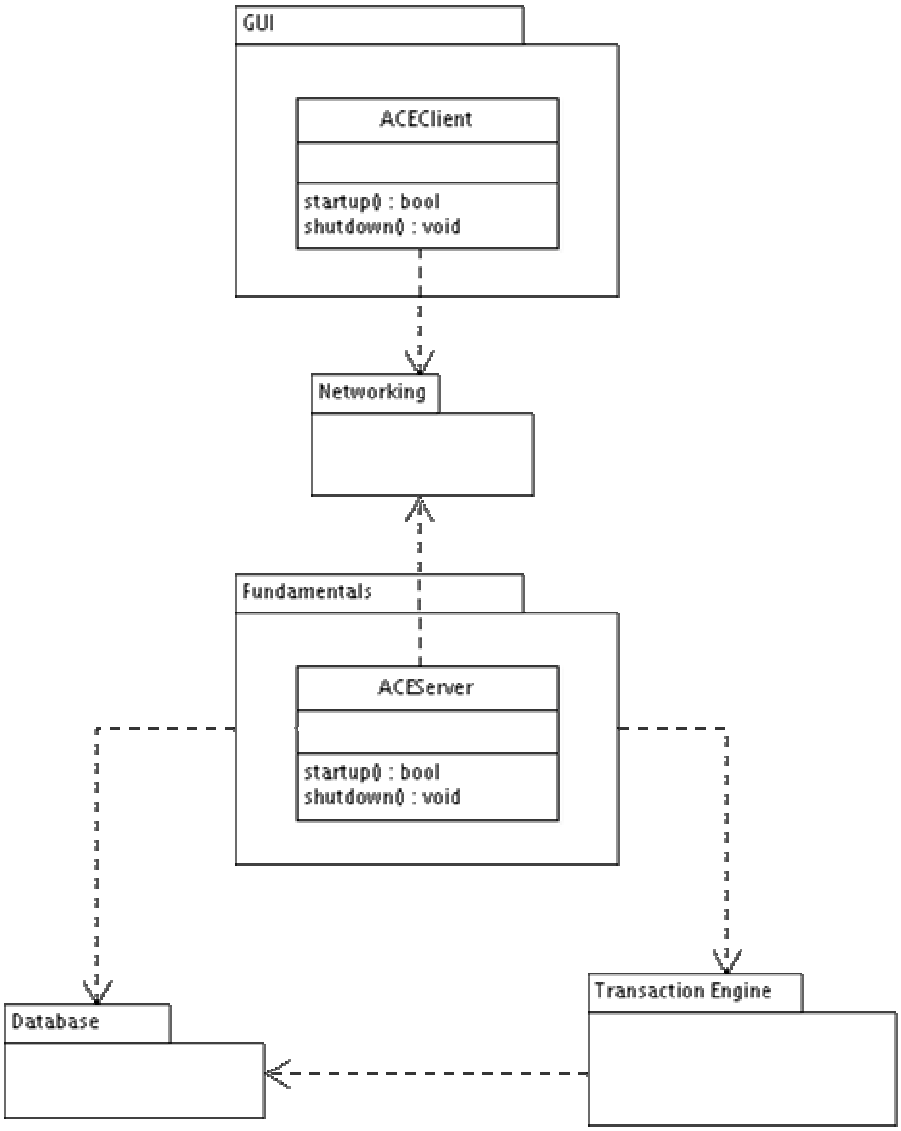

Figure 1. The packages diagram of the ACE system

1.2. Class diagram

1.2.1. Fundamentals Package (Entity classes)

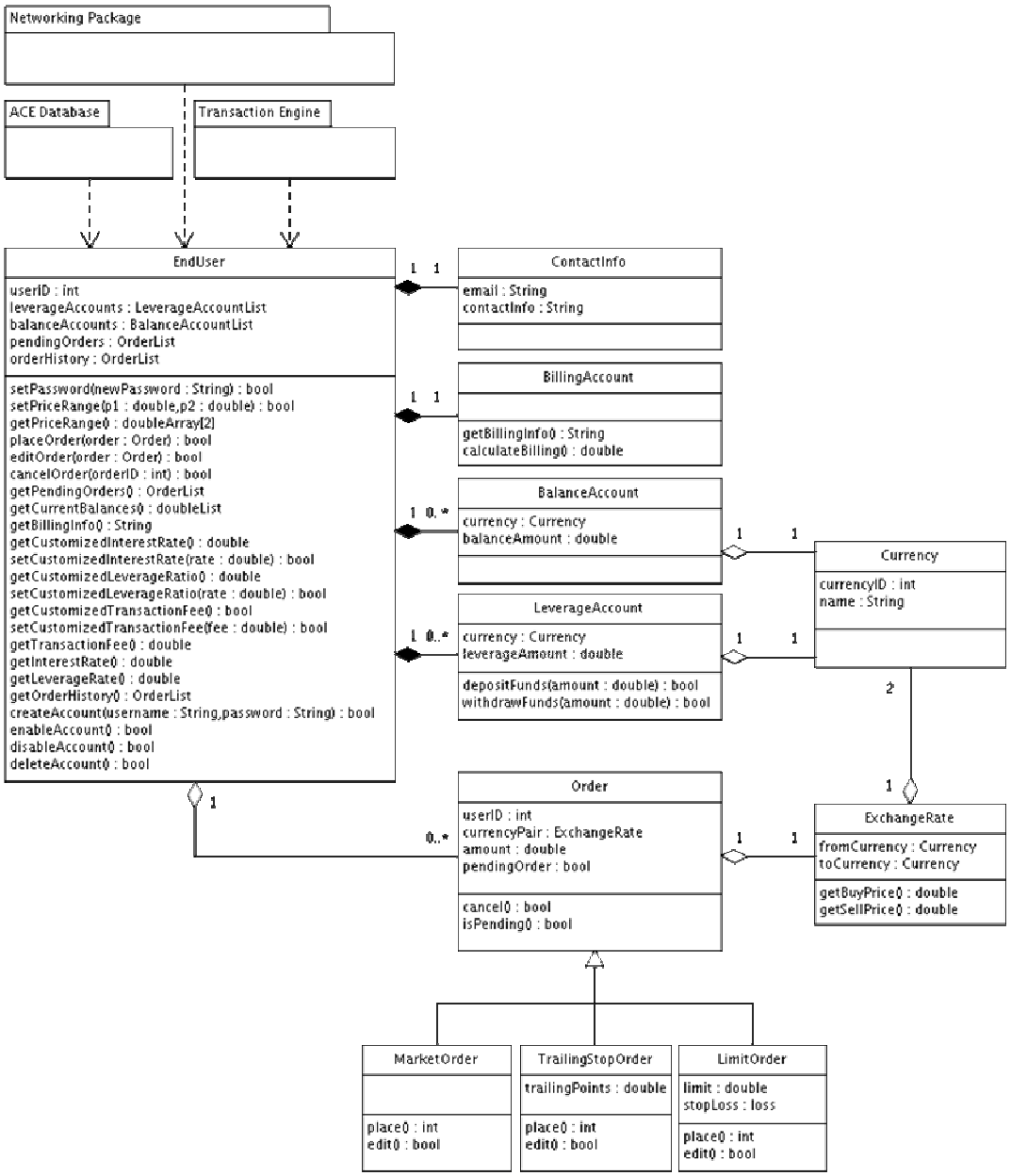

Figure 2. The class diagram for the Fundamentals package, part of ACE system

ClassName: EndUser	
Attributes:	• userID : int • leverageAccounts : LeverageAccountList • balanceAccounts : BalanceAccountList • pendingOrders : OrderList • orderHistory : OrderList
Operations:	• setPassword(newPassword : String) : bool • setPriceRange(p1 : double, p2 : double) : bool • getPriceRange() : doubleArray[2] • placeOrder(order : Order) : bool • editOrder(order : Order) : bool • cancelOrder(orderID : int) : bool (Won't be implemented according to the Implementation Focus sheet) • getPendingOrders() : OrderList • getCurrentBalances() : doubleList • getBillingInfo() : String (Won't be implemented according to the Implementation Focus sheet) • createAccount(username : String, password : String) : bool • getCurstomizedInterestRate() : double • setCurstomizedInterestRate(rate : double) : bool • getCurstomizedLeverageRatio() : double • getCurstomizedLeverageRatio(ratio : double) : bool • getCurstomizedTransactionFee() : double • getCurstomizedTransactionFee(fee : double) : bool • getTransactionFee() : double • getInterestRate() : double • getLeverageRate() : double • getOrderHistory() : OrderList (Won't be implemented according to the Implementation Focus sheet)
Description:	This class interfaces all EndUser related functionalities (except login and logout).
Relationship:	• Fundamentals Package • Networking Package: AdminParser, EndUserParser
Backward Traceability:	• Requirement 3.1.1.3 • Requirements 3.1.2.1-3.1.2.10 • Requirements 3.1.3.1-3.1.3.8

ClassName: ContactInfo	
Attributes:	• email : String • contactInfo : String
Operations:	• getEmail() : String • setEmail(email : String) : bool • getContactInfo() : String • setContactInfo(info : String) : bool
Description:	Read and write the end-user's contact information about an end-user.
Relationship:	• EndUser • DBConnection
Backward Traceability:	• Requirement 3.1.3.4

ClassName: BillingAccount	
Attributes:	
Operations:	• getBillingInfo() : String • setBillingInfo(info : String) : bool • calculateBilling() : double
Description:	Read and write the end-user's billing information to the database.
Relationship:	• EndUser • DBConnection
Backward Traceability:	• Requirement 3.1.2.5

ClassName: BalanceAccount	
Attributes:	• currency : Currency • balanceAmount : double
Operations:	• getCurrency() : Currency • setCurrency(currency : Currency) : bool • getBalanceAmount() : double • setBalanceAmount(amount : double) : bool
Description:	Holds the current position of an end-user in a particular currency.
Relationship:	• EndUser • DBConnection
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

ClassName: LeverageAccount	
Attributes:	
Operations:	• getLeverageAmount() : double • setLeverageAmount(amount : double) : bool • depositFunds(amount : double) : bool • withdrawFunds(amount : double) : bool
Description:	Holds the current amount of leveraged money in a particular currency.
Relationship:	• EndUser • DBConnection
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

ClassName: Order	
Attributes:	• currencyPair : ExchangeRate • amount : double • pendingOrder : bool
Operations:	• cancel() : bool (Won't be implemented according to the Implementation Focus sheet) • isPending() : bool
Description:	This class generalizes all three types of orders supported by the ACE Server.
Relationship:	• EndUser • ExchangeRate • MarketOrder • LimitOrder • TrailingStopOrder
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

ClassName: MarketOrder	
Attributes:	
Operations:	• place() : bool • edit() : bool
Description:	Place, Edit or Cancel a MarketOrder.
Relationship:	• Order • DBConnection
Backward Traceability:	• Requirements 3.1.2.2

ClassName: LimitOrder (limit order or stop loss)	
Attributes:	• stopLoss : bool • limit : double (percentage)
Operations:	• place() : bool • edit() : bool
Description:	Place, Edit or Cancel a LimitOrder.
Relationship:	• Order • DBConnection
Backward Traceability:	• Requirements 3.1.2.3

ClassName: TrailingStopOrder	
Attributes:	• trailingPoints : double
Operations:	• place() : bool • edit() : bool
Description:	Place, Edit or Cancel a TrailingStopOrder.
Relationship:	• Order • DBConnection
Backward Traceability:	• Requirements 3.1.2.4

ClassName: Currency	
Attributes:	• currencyID : int • name : String
Operations:	• getName() : String • getCurrencyID() : int • setCurrencyID(id : int) : bool
Description:	Defines a type for a currency.
Relationship:	• BalanceAccount • LeverageAccount • Transaction Engine Package • GUI Package
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

ClassName: ExchangeRate	
Attributes:	• fromCurrency : Currency • toCurrency : Currency
Operations:	• getBuyPrice() : double • getSellPrice() : double • getBuyPriceHistory(timeScale : int) : doubleArray • getSellPriceHistory(timeScale : int) : doubleArray
Description:	Defines a currency pair and provide the related functionality to a currency pair.
Relationship:	• Currency • BalanceAccount • LeverageAccount • DBConnection
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

1.2.2. Transaction Engine Package

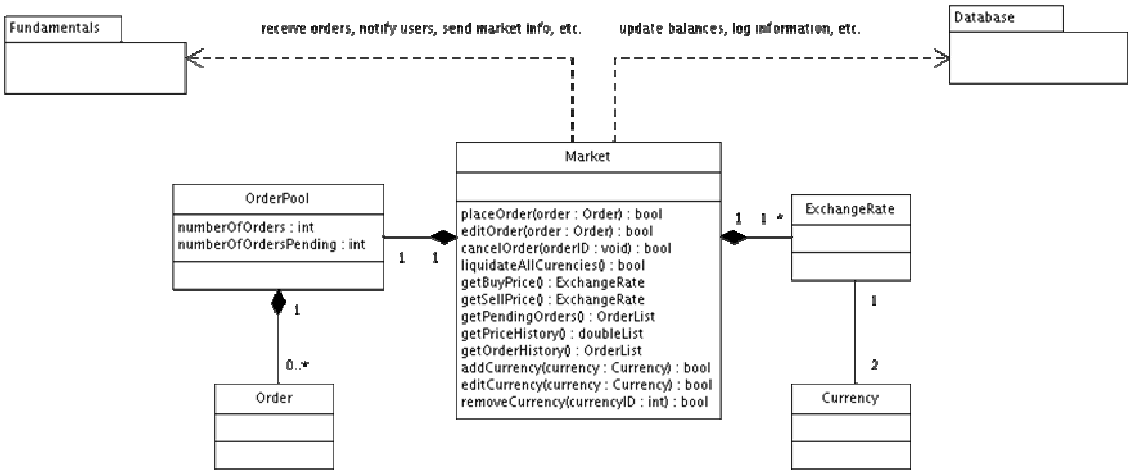

Figure 3. The class diagram for the Transactions Engine package, part of ACE system

ClassName: OrderPool	
Attributes:	• numberOfOrders : int • numberOfPendingOrders : int
Operations:	• placeOrder(order : Order) : bool • editOrder(order : Order) : bool • cancelOrder(orderID) : bool (Won't be implemented according to the Implementation Focus sheet)
Description:	This class is the pool containing all pending orders. When two orders are matched, they are removed from the pool. Of course, each order is time-stamped as it enters the pool.
Relationship:	• Market
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

ClassName: Market	
Attributes:	
Operations:	• placeOrder(order : Order) : bool • editOrder(order : Order) : bool • cancelOrder(orderID) : bool (Won't be implemented according to the Implementation Focus sheet) • getBuyPrice() : double • getSellPrice() : double
Description:	This class' role is to match two orders together. Essentially, when an order is placed, it is compared to each pending order in the pool in chronological order until a match is found. If no match was made, that order is added to the pool. If a match is found, the two orders are removed from the pool.
Relationship:	• Currency • ExchangeRate
Backward Traceability:	• Requirements 3.1.2.2-3.1.2.4

1.2.3. Networking Package

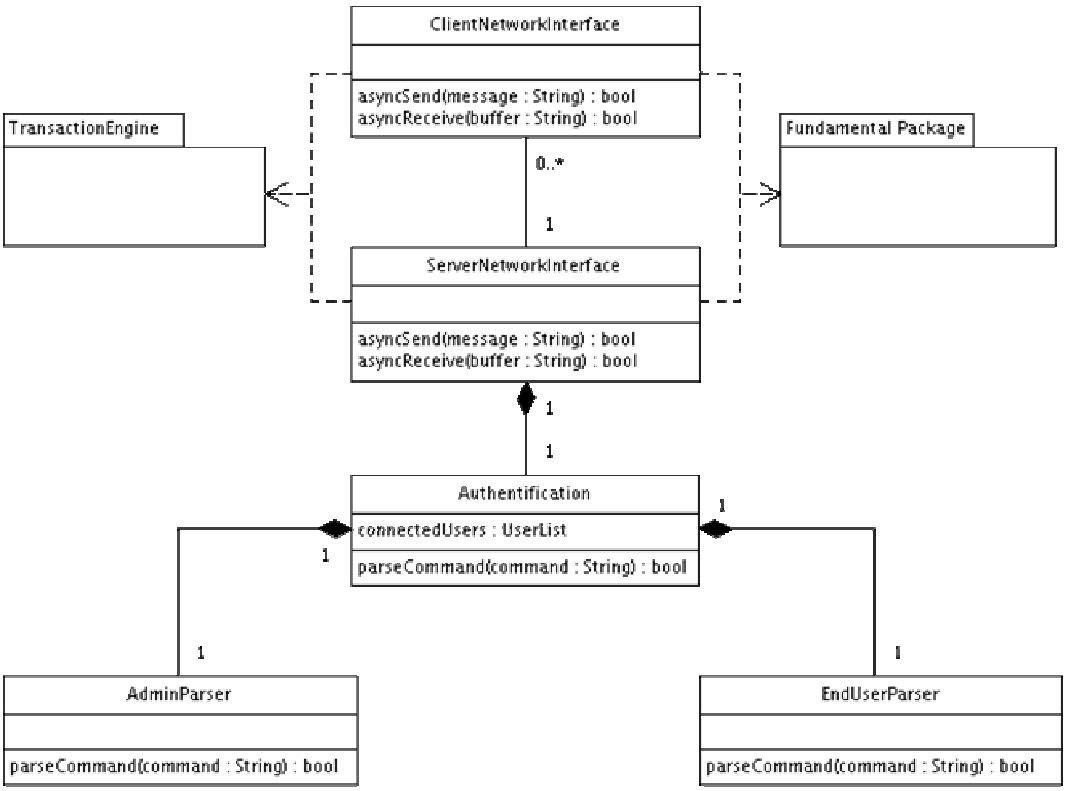

Figure 4. The class diagram for the Networking package, part of ACE system

ClassName: ClientNetworkInterface	
Attributes:	The attributes are implementation-specific.
Operations:	• connect(server : String, port : int) : int (Socket) • disconnect() : bool • asyncSend(message : String) : bool • asyncReceive(buffer : String) : bool
Description:	This class is essentially the facade between the ACE client and the rest of the ACE system that requires Internet access.
Relationship:	• GUI Package
Backward Traceability:	• All functional requirements (section 3) in the SRS Document (Mission 1) • Objective 1.3.3 of the SRS Document (Mission 1)

ClassName: ServerNetworkInterface	
Attributes:	The attributes are implementation-specific.
Operations:	• asyncSend(message : String) : bool • asyncReceive(buffer : String) : bool
Description:	This class sends and receives messages to and from, primarily, the client.
Relationship:	• Fundamentals Package • Transaction Engine Package
Backward Traceability:	• All functional requirements • Objective 1.3.3 of the SRS Document (Mission 1)

ClassName: Authentification	
Attributes:	• connectedUsers : UserList
Operations:	• parseCommand(command : String) : bool
Description:	When a message is received, the Authentication class verifies that the socket from which it was received is associated with an authenticated user. Then it checks what type of user the message originated from and redirects the message to the appropriate parser. Should the message be a login attempt, and it is the username and password are verified, then that user is granted a socket with authenticated status.
Relationship:	• Fundamentals Package • Database Package
Backward Traceability:	• Requirement 3.1.1.1 • Requirement 3.1.1.2

ClassName: AdminParser	
Attributes:	The attributes are implementation-specific.
Operations:	• parseCommand(command : String) : bool
Description:	The Network Interface only sends and receives String messages. Parsers actually read those messages and redirect the call to the proper classes.
Relationship:	• Fundamentals Package • Transaction Engine Package
Backward Traceability:	• All functional requirements

ClassName: EndUserParser	
Attributes:	The attributes are implementation-specific.
Operations:	• parseCommand(command : String) : bool
Description:	This class parses messages specific to end-users and redirects the message to the target class, or package facade.
Relationship:	• Fundamentals Package • Transaction Engine Package
Backward Traceability:	• All functional requirements

1.2.4. GUI Package

1.2.4.1 End-User GUI Classes

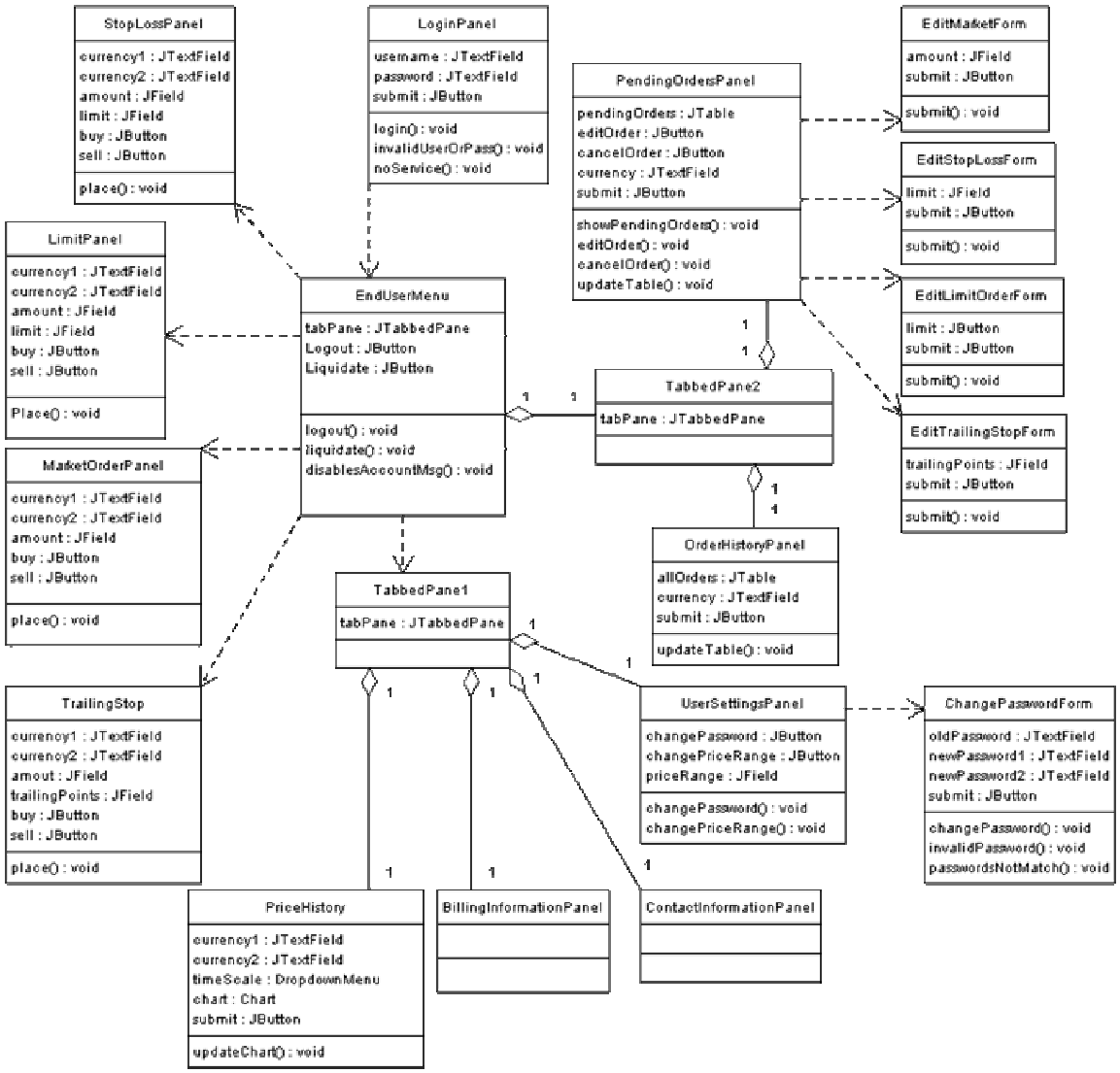

Figure 5. The class diagram for the GUI package (only end-user classes)

Below, among the GUI classes, you will notice that only the most important of attributes and methods are stated. Even then, not all the methods have the proper signatures. This is for legibility. Also, we are not entirely sure as of yet how currency pairs can be chosen by a user, the solution we implemented in this report involves a text field for each currency in the pair. As the user types in a currency name, the text field auto-completes the name for the user.

ClassName: LoginPanel	
Attributes:	• Username : JTextField • Password : JTextField • Submit : JButton
Operations:	• Login() : void • InvalidUserOrPass() : void • noService() : void
Description:	This is the login screen that greets anyone who starts their client, including administrators. In order for an end-user to access the ACE system and begin trading, he must successfully login.
Backward Traceability:	• 3.1.1.1.

ClassName: EndUserMenu	
Attributes:	• TabPane : JTabbedPane • Logout : JButton • Liquidate : JButton
Operations:	• Logout() : void • Liquidate : void • disableAccountMsg() : void
Description:	A successful Login will destroy the login screen and the main frame, the EndUserMenu, will appear. This is what the end-user will interact with. All information the end-user needs will be laid out in this frame. That is to say the end-user will not have to interact with any popup menus except for when editing orders or changing passwords.
Backward Traceability:	• 3.1.1.2. • 3.1.2.10

ClassName: LimitPanel	
Attributes:	• Currency1 : JTextField • Currency2 : JTextfield • Amount : JField • Limit : JField • Buy : JButton • Sell : JButton
Operations:	• Place() : void
Description:	This class is basically a form to fill out for a limit order. It is instantly visible after a user logs in.

Backward Traceability:	• 3.1.2.3.

ClassName: StopLossPanel	
Attributes:	• Currency1 : JTextField • Currency2 : JTextfield • Amount : JField • Limit : JField • Buy : JButton • Sell : JButton
Operations:	• Place() : void
Description:	This class is basically a form to fill out for a stop loss order. It is instantly visible after a user logs in.
Backward Traceability:	• 3.1.2.3.

ClassName: MarketOrderPanel	
Attributes:	• Currency1 : JTextField • Currency2 : JTextfield • Amount : JField • Buy : JButton • Sell : JButton
Operations:	• Place() : void
Description:	This is the form to fill out for a market order directly from the main interface. The Place() method behaves differently depending on whether the Buy or Sell button was pushed.
Backward Traceability:	• 3.1.2.2.

ClassName: TrailingStopPanel	
Attributes:	• Currency1 : JTextField • Currency2 : JTextfield • Amount : JField • TrailingPoints : JField • Buy : JButton • Sell : JButton
Operations:	• Place() : void
Description:	A form to fill out in order to place a trailing stop order.
Backward Traceability:	• 3.1.2.4.

ClassName: TabbedPane1	
Description:	This is just a container class to help organize the layout of the GUI.

ClassName: PriceHistoryPanel	
Attributes:	• Currency1 : JTextField • Currency2 : JTextfield • TimeScale : DropDownMenu • Chart : Chart • Submit : JButton
Operations:	• UpdateChart() : void
Description:	After selecting the currency pair and the time scale, the user ca view a chart graphically representing the change in a currency's value with respect to the other.
Backward Traceability:	• 3.1.2.9.

ClassName: BillingInformationPanel	
Description:	This class just shows information regarding the billed amount of the end-user.
Backward Traceability:	• 3.1.2.5.

ClassName: ContactInformationPanel	
Description:	Contact information of the end-user displayed here.
Backward Traceability:	None

ClassName: UserSettingsPanel	
Attributes:	• ChangePassword: JButton • ChangePriceRange : JButton • PriceRange : JField
Operations:	• ChangePassword() : void • ChangePriceRange() :void
Description:	An end-user may change his password or the price range, which impacts the buying and selling of his market orders. Choosing to change the password will cause a form to pop up.
Backward Traceability:	• 3.1.1.3. • 3.1.2.1.

ClassName: ChangePasswordForm	
Attributes:	• OldPassword: JTextField • NewPassword1: JTextField • NewPassword2: JTextField • Submit : JButton
Operations:	• ChangePassword() : void • InvalidPassword() : void • PasswordsNotMatch() : void
Description:	To change a password, all the user need to do is fill out the form and click the submit button.
Backward Traceability:	• 3.1.1.3.

ClassName: TabbedPane2	
Description:	This is just a container class to help organize the layout of the GUI.

ClassName: OrderHistoryPanel	
Attributes:	• AllOrders : JTable • Currency : JTextField • Submit : JButton
Operations:	• UpdateTable() : void
Description:	This is basically a table showing all non-pending orders that the end-user placed. The choice to view only those orders involving a specific currency, or all currencies, is available.
Backward Traceability:	• 3.1.2.7.

ClassName: PendingOrdersPanel	
Attributes:	• PendingOrders : JTable • Currency : JTextField • Submit : JButton • EditOrder : JButton • CancelOrder : JButton
Operations:	• ShowPendingOrders() : void • UpdateTable() : void • EditOrder() : void • CancelOrder() : void
Description:	This table shows all pending orders that are waiting to be matched up with another order. Since all these orders are pending, next to each order is a cancel and edit order button. Pushing the edit button will bring up a form depending on what kind of order it was.
Backward Traceability:	• 3.1.2.3. • 3.1.2.4. • 3.1.2.6.

ClassName: EditTrailingStopForm	
Attributes:	• Limit : JField • Submit : JButton
Operations:	• Submit() : void
Description:	Change the number of trailing points of the order.
Backward Traceability:	• 3.1.2.4.

ClassName: EditLimitOrderForm	
Attributes:	• Limit : JField • Submit : JButton
Operations:	• Submit() : void
Description:	Allows the end-user to change the threshold of the order.
Backward Traceability:	• 3.1.2.3.

ClassName: EditStopLossForm	
Attributes:	• Limit : JField • Submit : JButton
Operations:	• Submit() : void
Description:	Change the threshold of the order.
Backward Traceability:	• 3.1.2.3.

ClassName: EditMarketForm	
Attributes:	• Amount : JField • Submit : JButton
Operations:	• Submit() : void
Description:	Change the amount of the market order.
Backward Traceability:	None

1.2.4.2. Administrator GUI Classes

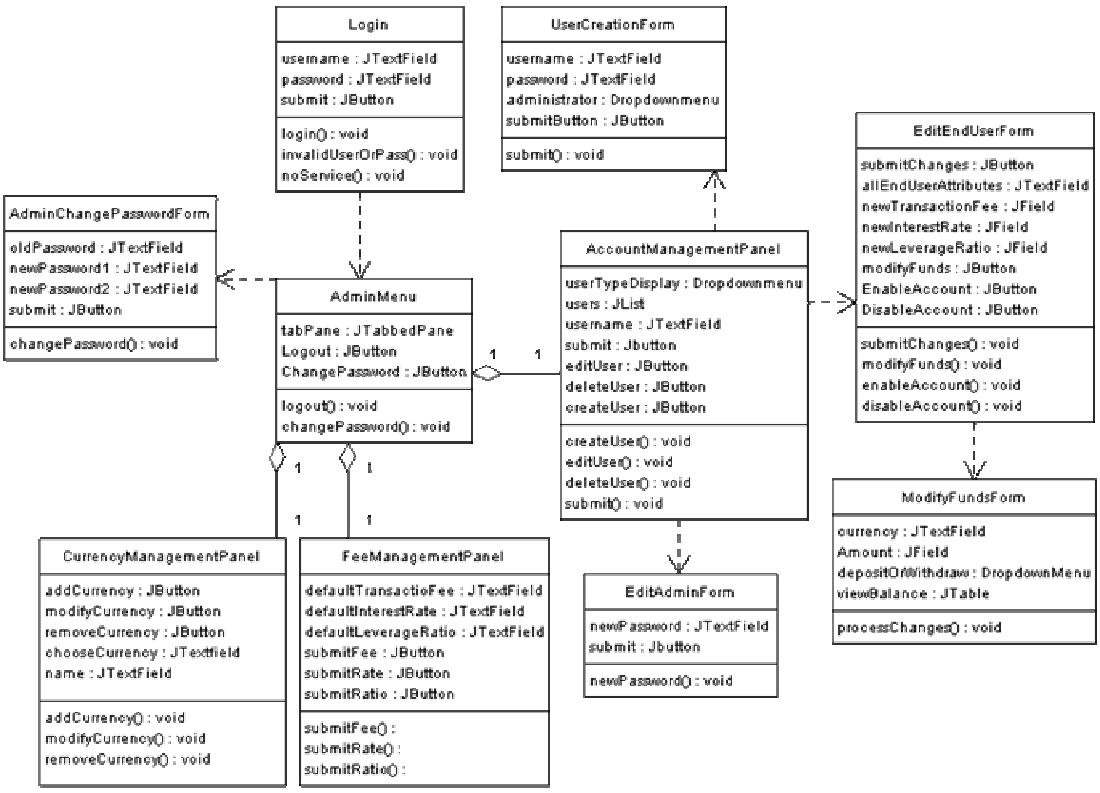

Figure 6. The class diagram for the GUI package (only administrator classes)

ClassName: Login	
Description:	This class was already discussed in the end-user GUI section.

ClassName: AdminChangePasswordForm	
Description:	Again, this class is analogous to the end-user's counterpart.

ClassName: AdminMenu	
Attributes:	• tabPane : JTabbedPane • Logout : JButton • ChangePassword : JButton
Operations:	• Logout() : void • ChangePassword() : void
Description:	This classes main purpose it to provide the user interface's look and feel. All of the major functionality is contained in its "adjacent" classes that are connected to it in the class diagram. Most of the frame's space will be occupied by one panel. The panel that is currently displayed is chosen by the user via the tabs above the panel.
Backward Traceability:	• 3.1.1.2. • 3.1.1.3.

ClassName: CurrencyManagementPanel	
Attributes:	• AddCurrency : JButton • ModifyCrrency : JButton • RemoveCurrency : JButton • ChooseCurrency : JTextField • Name : JTextField
Operations:	• AddCurrency() : void • ModifyCurrency() : void • RemoveCurrency() :void
Description:	In this panel, an administrator can add, modify, or delete a currency. The only thing that can be modified is the currency's name.
Backward Traceability:	• 3.1.3.11.

ClassName: FeeManagementPanel	
Attributes:	• DefaultTransactionFee : JTextField • DefaultInterestRate : JTextField • DefaultLeverageRatio : JTextField • SubmitFee : JButton • SubmitRate : JButton • SubmitRatio : JButton
Operations:	• SubmiFeet() : void • SubmitRate() : void • SubmitRatio() : void
Description:	This is where an administrator can modify the default

	transaction fee, interest rate, and leverage ratio.
Backward Traceability:	• 3.1.3.9. • 3.1.3.10. • 3.1.3.12.

ClassName: AccountManagementPanel	
Attributes:	• UserTypeDisplay : DropDownMenu • Users : JList • Username : JTextField • Submit : JButton • EditUser : JButton • DeleteUser : JButton • CreateUser : JButton
Operations:	• Submit() : void • CreateUser() : void • EditUser() : void • DeleteUser() : void
Description:	This is where an administrator can create, edit, and delete administrators and end-user accounts. Depending on which choice the administrator makes, this panel can lead him to different popup forms.
Backward Traceability:	• 3.1.3.1. to 3.1.3.8. • 3.1.3.13

ClassName: UserCreationForm	
Attributes:	• Username : JTextField • Password : JTextField • Administrator : DropdownMenu (Boolean) • submitButton : JButton
Operations:	• Submit() : void
Description:	By filling out this form, and administrator may create another administrator account, or an end-user.
Backward Traceability:	• 3.1.3.1.

ClassName: EditEndUserForm	
Attributes:	• submitChanges : JButton • allEndUserAttributes (this means contact info) : JTextField • NewTransactionFee : JField • NewInterestRate : JField • NewLevarageRatio : JField • ModifyFunds : JButton • EnableAccount : JButton • DisableAccount : JButton
Operations:	• SubmitChanges() : void • ModifyFunds() : void • EnableAccount() : void • DisableAccount() : void
Description:	Here, the administrator has the power to disable or enable an end-user account, change their contact information, as well as personalize their transaction fee, interest rate, and leverage ratio.
Backward Traceability:	• 3.1.3.2. • 3.1.3.4. to 3.1.3.8. • 3.1.3.13.

ClassName: ModifyFundsForm	
Attributes:	• Currency : JTextField • Amount : JField • DepositOrWithdraw : DropDownMenu • ViewBalance : JTable • Submit : JButton
Operations:	• ProcessChanges() : void
Description:	This form allows the administrator to add and withdraw funds to/from an end-user's account for each individual currency.
Backward Traceability:	• 3.1.3.5. • 3.1.3.6.

ClassName: EditAdminForm	
Attributes:	• newPassword : JTextField • submit : JButton
Operations:	• newPassword() : void
Description:	Should ever an administrator forget his password, he can ask another administrator to reset his password for him.
Backward Traceability:	None

1.2.5. Database Package

ClassName: Database	
Operations:	• connect(server : String, port : int) : bool • query(message : String) : String • disconnect() : bool
Description:	Interacts with the Database (PostGRE SQL) Server.
Relationship:	• ACEServer • Authentication • End-User • Market • OrderPool
Backward Traceability:	All functional requirements.

2. The Dynamic Model

2.1 Login

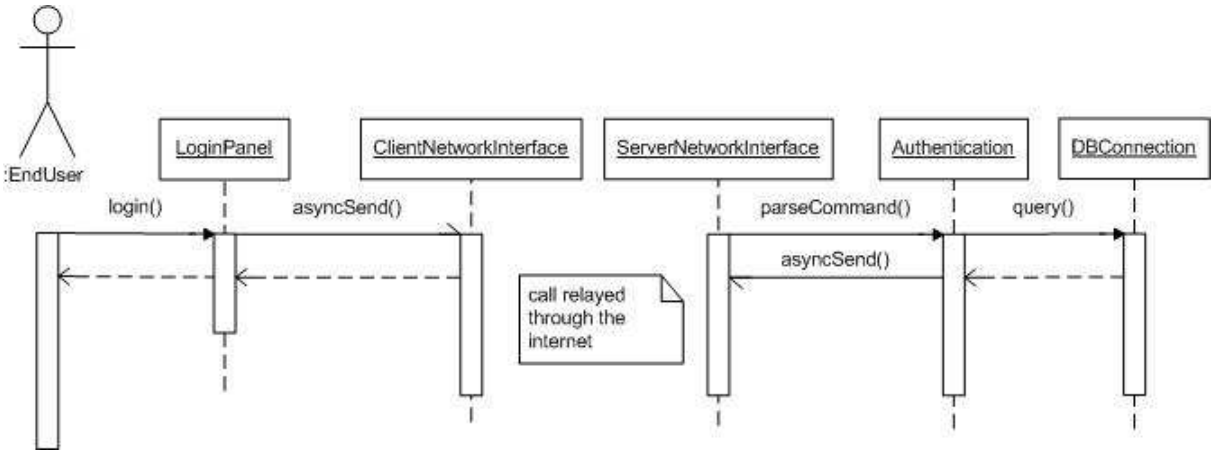

Figure 7. The sequential diagram for the Login use case

2.2 Place Market Order

The end-user places a market order, specifying the currency pair, amount, and whether to buy or sell.

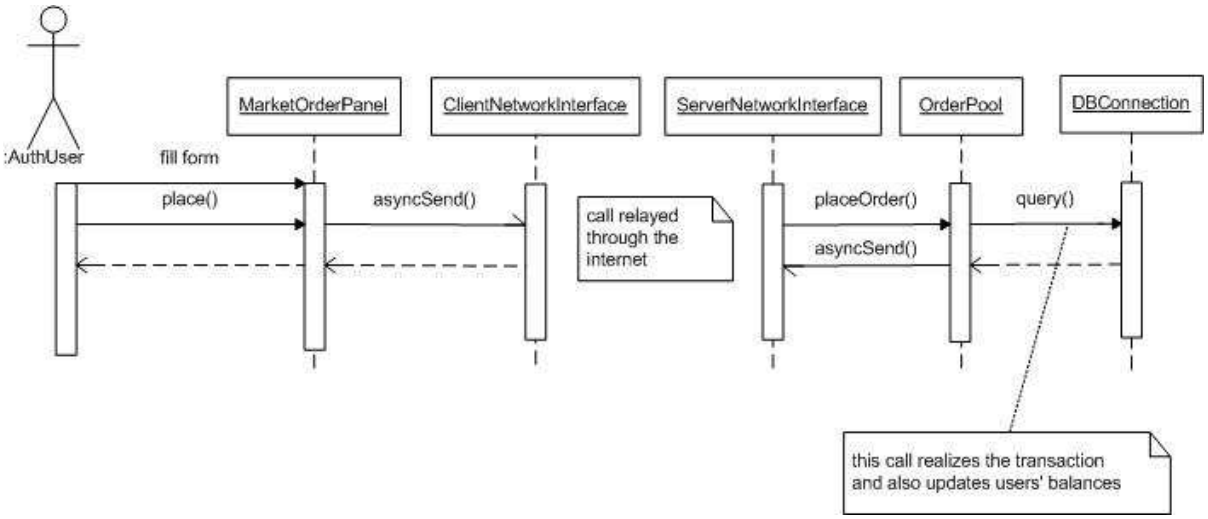

Figure 8. The sequential diagram for the Place Market Order use case

2.3 View Pending Orders

The end-user chooses to view all his pending orders and the ACE client presents the user with a table with this information. It is from this table which the end-user may edit or cancel any pending orders. The end-user may choose to view only those orders involving a particular currency, or for all currencies.

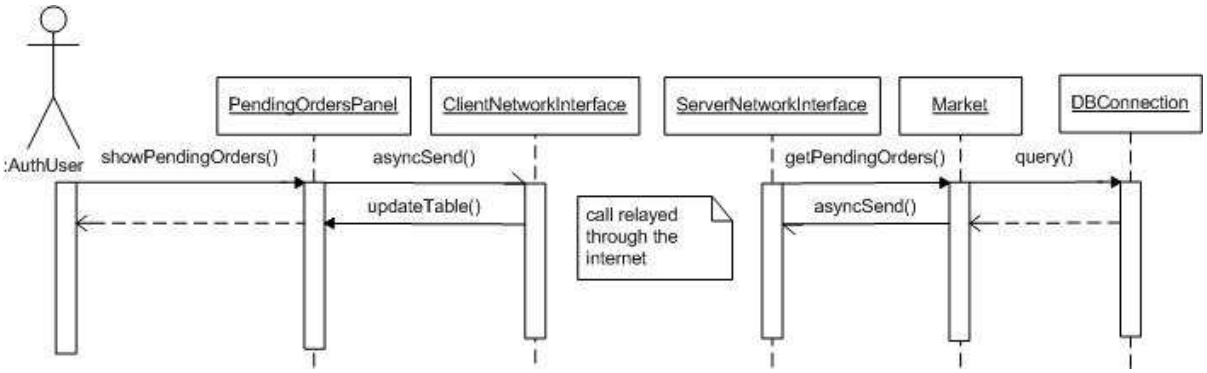

Figure 9. The sequential diagram for the View Pending Orders use case

2.4 View Order History

The end-user chooses to view his order history, and the ACE client presents this information in a table.

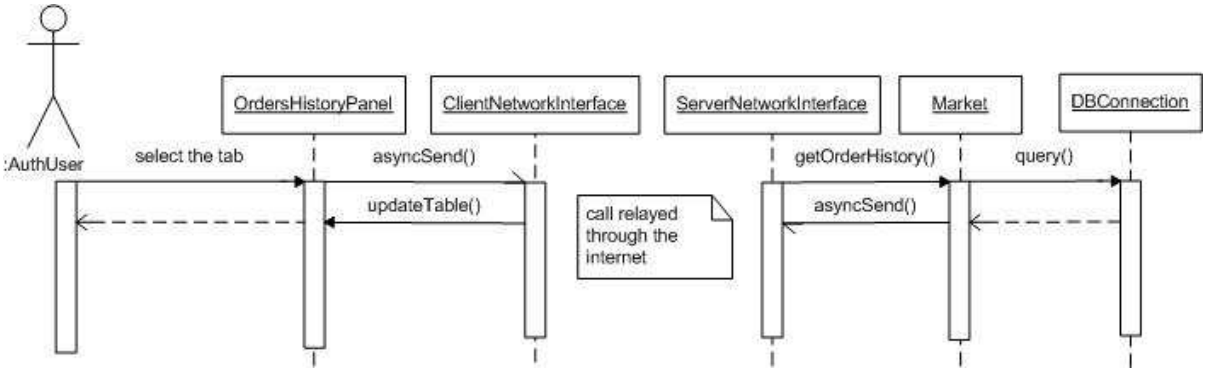

Figure 10. The sequential diagram for the View Order History use case

2.5 View Chart (previous name - View Price History)

The ACE client presents in the form of a chart the history of a currency pair rate. The end-user may choose which currency pair to display as well as the time period.

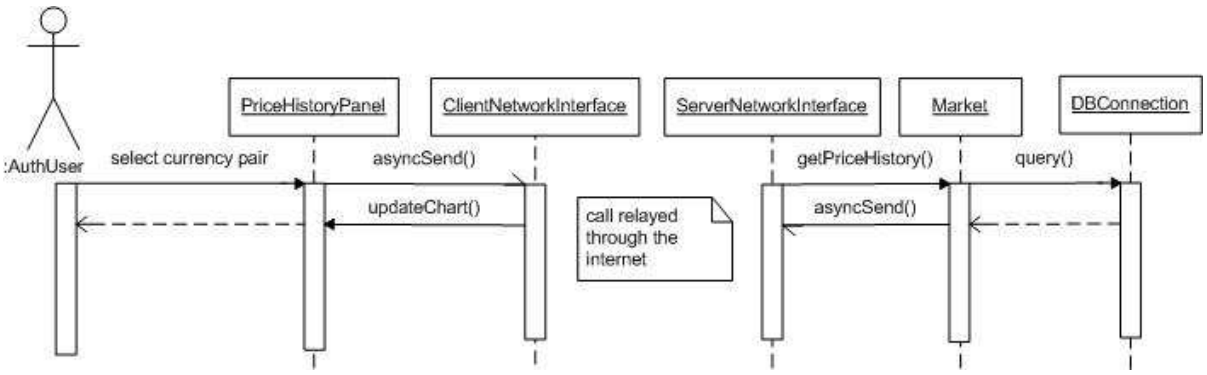

Figure 11. The sequential diagram for the View Chart use case

2.6 Liquidate

The end-user decides to liquidate all his currencies, and the ACE client asks the end-user if he is sure before proceeding with his demand.

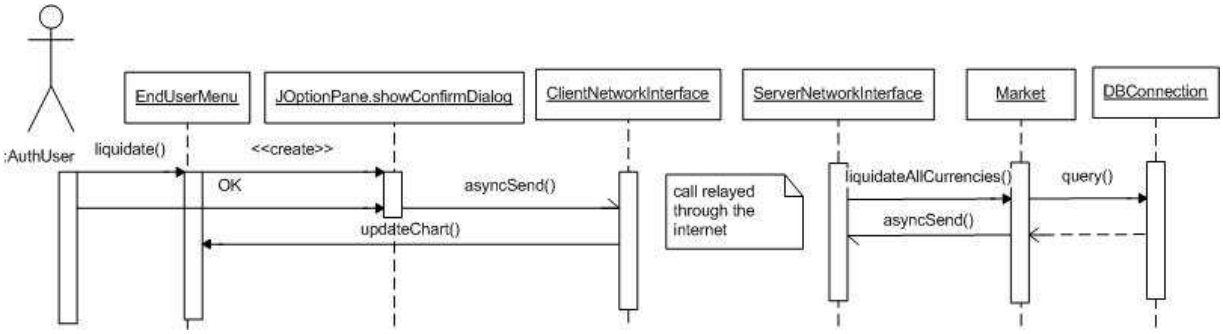

Figure 12. The sequential diagram for the Liquidate use case

2.7 Edit Limit Order

While viewing all of his pending orders, the end user chooses to edit one of his limit orders in order to change one of its attributes.

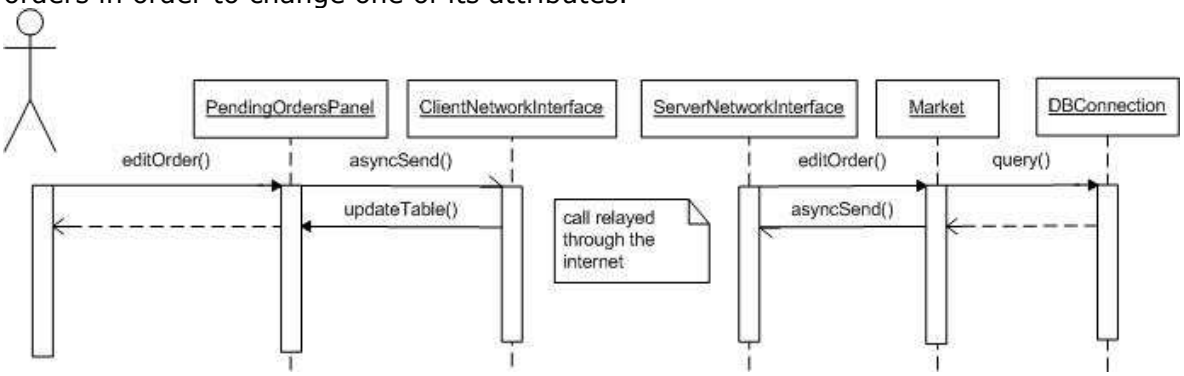

Figure 13. The sequential diagram for the Edit Limit Order use case

2.8 Add Currency (Admin)

The admin adds a new currency to the ACE system.

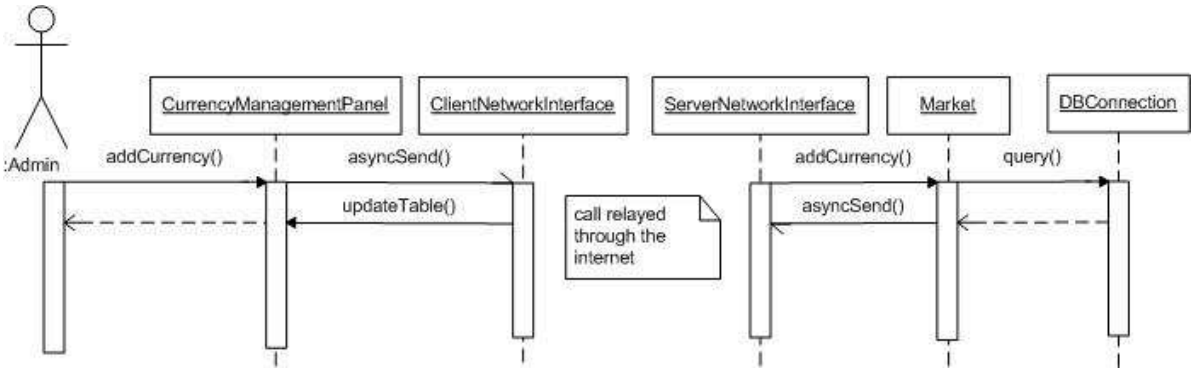

Figure 14. The sequential diagram for the Add Currency (Admin) use case

3. Work Plan & Schedule

3.1 Full-time Responsibilities:

- Source Code Standardization (Kirill, Michael)
- Database Management (George)
- Integration & SVN support (Alex)
- Meeting the deadlines (Gabriel)

3.2 Overview Schedule:

Phase 1 (20/11 - 22/11)

- Database Server Setup (George)
- Architecture Skeleton
    - ACEServer class (Alex)
    - Fundamental (Entity) Package (Alex, Gabriel)
    - Transaction Engine Package (Alex)
    - Networking Package (Gabriel)
    - Database Package (George)
    - ACEClient End-User Application GUI (Kirill)
    - ACEClient Administrator Application GUI (Michael)
- Sketching Test Plan (Sub-task A) (Michael, George)
    - 6 scenarios (Michael)
    - 6 scenarios (George)
- Sketching Network API (Gabriel)
- Database Architecture (George)

Phase 2 (23/11 - 30/11)

- Coding (Everyone)
    - see the Coding Schedule section for further details
- Testing (Functionality & Performance) & QA (Everyone)

Phase 4 (31/11 - 02/12)

Documentation (Everyone must help)

- Sub-task A:
    - Finalizing Test Plan (Michael)
- Sub-task B:
    - README Instruction file (Gabriel)
    - JavaDoc generated list of classes (George)
- Sub-task C:

    - System Design Goals of the complete system
    - Proposed System Architecture
    - Subsystems
    - Deployment Diagram
    - Reliability Issues
- Appendix: Final Network API (Gabriel)
- Formatting (Gabriel)

Phase 5 (04/12)

Presentation (Everyone)

3.3 Coding Schedule

Fundamental (Entity) classes (Alex, Gabriel)

- EndUser (Alex)
- ContactInfo (Gabriel)
- BillingAccount (Gabriel)
- BalanceAccount (Gabriel)
- LeverageAccount (Gabriel)
- Order (Alex)
- MarketOrder (Alex)
- LimitOrder (Alex)
- TrailingStopOrder (Alex)
- Currency (Alex)
- ExchangeRate (Alex)

Transaction Engine Package (Alex, George)

- Market (Alex)
- OrderPool (George)

Networking Package (Gabriel)

- ClientNetworkInterface
- ServerNetworkInterface
- Authentification
- EndUserParser
- AdminParser

GUI Package (Michael, Kirill)

- ACEAdminClient (Michael)
- ACEEndUserClient (Kirill)

Database Package (George)

- DBConnection