Requirements Analysis Document

Joel Diana, Jennifer Fassett, Guangming He, Korrie Weisenberger

Realtime Commercial Bidding System
Team 4

April 26, 2002
# Contents

1 Introduction  
   1.1 Problem Description ........................................... 3  
   1.2 Motivation .................................................. 3

2 Overview  

3 Requirements  
   3.1 Register/Login/Logout ......................................... 4  
   3.2 Create Auction ................................................ 5  
   3.3 Cancel Auction ............................................... 5  
   3.4 Join Auction ............................................... 5  
   3.5 Bid on Auction ............................................. 6  
   3.6 Withdraw from Auction .................................... 6  
   3.7 Win Auction ................................................. 6  
   3.8 Check Bid History .......................................... 6

4 UML Analysis  
   4.1 Use Cases .................................................. 7  
   4.2 Object Model .............................................. 12  
      4.2.1 Class Diagram .................................... 12  
      4.2.2 Data Dictionary .................................... 13  
   4.3 Dynamic Model ............................................. 16  
      4.3.1 State Diagrams .................................... 16  
      4.3.2 Sequence Diagrams ................................ 18

5 Model Checking  
   5.1 Newbid == Maxbid ......................................... 21  
   5.2 Ending an Auction .......................................... 22

A Promela Code  


1 Introduction

Online brokering has become increasingly popular over the last few years. There is currently a role for a distributed online brokering system that is not strictly a consumer-oriented system. The purpose of this document is to provide all of the requirements for implementing such a system.

The following document is a requirements analysis of an online brokering system. It is divided up into four key sections: introduction, overview, requirements and UML analysis. The introduction section includes a problem description and a motivation section. The requirements section is further broken into the following eight sections: register/login/logout, create auction, cancel auction, join auction, bid on auction, withdraw from auction, win auction and check current bids. The UML analysis section includes use case diagrams, an object model consisting of a class diagram and data dictionary and a dynamic model consisting of state diagrams and sequence diagrams.

1.1 Problem Description

Our client believes that there is a demand for an online system that mirrors a real auction environment but is more flexible and cheaper to run. This system will facilitate the brokering of various kinds of commercial goods and commodities by providing a place for users, both consumers and commercial enterprises, to buy and sell goods in approximate realtime. The program will simulate real auctions, but as a distributed application, to allow users to participate globally.

We have been contracted to architect a distributed online brokering system for commercial products. The client program will allow users to register and then login to the system. Once logged in, users can create and participate in regular and/or reverse auctions in approximate realtime. A regular auction consists of a user putting an item up for auction and it goes to the high bidder. A reverse auction consists of a user soliciting an item and the sale goes to the low bidder. A user creates a regular auction if he wishes to sell an item, whereas a user creates a reverse auction if he wishes to purchase an item. As a result of asking questions of the customer, it was determined that a user may only participate in one auction at a time (with a single instantiation of the program), and he or she may not participate in an auction he or she created. An auction has an item, creator, starting time, and ending time associated with it, as well as an optional reserve price. An auction will end at or before its specified end time, or if it does not receive a bid for a specified amount of time. The system will administer all auctions and notify participating users of new bids. The system will maintain a list of bids for the auctions with which each user is associated.

1.2 Motivation

Due to the Internet, it is now possible to broker various kinds of commercial goods and commodities virtually. Buying and selling goods online eliminates the hassles of holding auctions that either include joining large groups of people together, or making numerous phone calls. Potential buyers and sellers can now browse through auction sites from their own homes, creating a more ideal environment. Our customer believes that an online brokering system will be financially profitable in the E-marketplace. The job of our team is to analyze the project requirements and sort through the customer’s specifications to determine what is needed for the system. We need to choose what additional features will be needed and also what unnecessary features have been asked for based on the customer’s goals and design. Our online brokering system will make both selling and buying goods over the Internet very efficient.
This is a generally low-risk endeavor. There is definitely a market for this kind of system, and it
would not involve high-risk technology or difficult design techniques. The only area that would need
special attention would be the legal ramifications associated with such a system. A legal agreement
between the system and the users is essential, as is a process by which users may file complaints
and/or law suits.

2 Overview

This online system will enable the easy exchange of goods via regular and reverse auctions. A user
must first register and login in order to use the system, and then will be able to view a list of all
auctions. A user can create his or her own auction. A user may not participate in his or her own
auction, but he or she may cancel it. A user can join an auction and place a bid on the item being
auctioned, if his or her bid meets the reserve. A user can withdraw from an auction if he or she
does not hold the winning bid. The system will notify the winner of each auction after the auction
is over, and bill the auction’s creator for the auction site’s trading services. Each auction will occur
in approximate realtime. The auction ends when the specified ending time is reached or it times
out (no bid is placed for a set amount of time). The system will inform bidders of the winning bid,
the time remaining in the auction, and the number of bidders and bids associated with the auction.
The system will also allow the user to view a history of his or her bids. It is our company’s job to
analyze what the customer is asking and to give the customer a viable system for brokering goods.

3 Requirements

3.1 Register/Login/Logout

The system must authenticate users, and keep track of which users are currently logged into the site.
Upon connecting to the system, a user will be prompted to login or register. The user will not
be allowed to participate or view any auctions until logged in. The user cannot login without an
account, which provides them with a valid username and password. To create an account, the user
must register. The user must accept the terms and conditions of the site in order to register. The
site’s lawyers will draft these terms and conditions. The user will be prompted to enter pertinent
information including e-mail address, password, telephone number, user ID, mailing address, and
name. This information will be stored in a database by the system. Appropriate measures must be
taken to ensure the security of user information. A user that has already created an account can
login. They must enter their username/password combination, which will be authenticated by the
system. (The system will check this username/password combination with the database in order
to verify the identity of the user). Upon verification of the user’s identity, the user is allowed into
the site and the system adds this user to a list of active users. At any time after logging in, a user
may choose to logout of the system. The user will then be removed from the list of active users
and will not be allowed to interact with the system until they log back in.
3.2 Create Auction

A user can create an auction at any time, and the system must keep track of and administer the auction.

Any user that has successfully logged into the site will be allowed to create an auction. The request will be submitted to the Online Brokering System which will create an auctioneer and auction on behalf of the user. This prevents the user from directly interacting with the auction or auctioneer, protecting them. A user that wishes to create an auction must provide information including end time, reserve price, start time, the auction type (reverse or regular), and an image, name, and short and long descriptions of the item being auctioned. The auctioneer will store this information along with the user’s ID in a database. It will also assign an ID to the auction at its inception. The auctioneer will keep track of the best bid and its owner, and a timer for the auction. The system must maintain a list of auctions/auctioneers. The auctioneers ensure that auctions are started and ended on time.

3.3 Cancel Auction

The online brokering system must be able to cancel an auction.

The auctioneer will keep track of the user ID of the user who created the auction. The system will allow the creator of an auction to cancel that auction under extenuating circumstances. The auction’s creator will not be able to do this directly. There will be some outside procedure (such as e-mailing the site’s contact representative) for requesting an auction cancel. If the auction is underway and has been joined by any users, the auctioneer must notify all users involved in the auction, including all users that are currently joined with the auction and the owner of the best bid, of the cancellation. If a user cancels their auction, the auction’s creator will be charged a substantial fee. This is to deter users from canceling auctions, but if necessary, the option is available.

3.4 Join Auction

The auctioneer for each auction will maintain a list of users that are currently joined with the auction. A user must be joined with an auction before they can place bids in that auction. The system may refuse a user entry into an auction based on their credit rating. The auctioneer will perform the credit rating check, and the customer’s lawyer will determine the method.

A user must be registered and logged into the site in order to join an auction. They may not join more than one auction at a time unless they have multiple windows running concurrently, as per the design document provided by the customer. In other words, only one auction per instantiation of the client code is allowed. When a user attempts to join an auction the auctioneer may look up that user’s credit rating, which will be maintained in a database. A user with a poor credit rating may not be allowed to join an auction. When a user joins an auction, the auctioneer adds them to that auction’s list of users. The auctioneer will notify all users that are involved in an auction of bids placed in that auction.
3.5 Bid on Auction

Once joined with an auction, a user can place a bid in that auction at any time (except if they hold the best bid). The auctioneer will keep track of the winning bid at all times.

A user that has joined an auction will be able to bid in that auction if they do not hold the winning bid. When a user wishes to bid in an auction, the auctioneer will prompt the user for a bid amount. If the bid amount is better than the current best bid, then the bid is accepted and the best bid is updated. The auctioneer will notify all users that the best bid has been updated.

3.6 Withdraw from Auction

A user may withdraw from an auction at any time. The auctioneer will no longer accept bids for that auction from that user.

A user that wishes to withdraw from an auction must be allowed to do so. However, if that user is the owner of the best bid, they are not relieved of their obligation, should they win the auction. The auctioneer will remove a user that has withdrawn from an auction from that auction’s list of active users, and that user will no longer be able to place bids in that auction.

3.7 Win Auction

When an auction ends, the auctioneer will determine the winner of the auction and notify that user and all users of the auction results.

The auctioneer will end an auction at that auction’s specified end time or alternatively, when a specified period of time has elapsed since the last bid was placed in that auction. The auctioneer will notify all users that are currently joined with the auction that the auction has been ended. For auctions that have a reserve price, if the reserve price has been met for the auction, then the owner of the best bid is the winner of the auction. If the reserve price has not been met then the auction will be determined to have no winner. For auctions that do not have a reserve price, the owner of the best bid is the winner of the auction. If there is not a best bid, then the auction has no winner. For all auctions, if a winner has been determined, the auctioneer must notify the winner and obtain that user’s payment information. The auctioneer will also inform all users joined with the auction. The auctioneer must also inform the owner of the auction of the auction’s results and bill them for the auction site’s trading services.

3.8 Check Bid History

The system will maintain a list of recent bids for each user.

A user that has registered and logged into the site can check their bids at any time. When a user wishes to check their bids the system will retrieve that user’s bid history from a database and display the information to the user. A user can check their bid history for any auction in which they have participated.
4 UML Analysis

4.1 Use Cases

The use case diagram (Figure 1) provides a graphical representation of the various functionalities the system needs to support. The system is made up of a system boundary, actors and use cases. In this system, the only actor is a user, so the Use Cases are all user actions. Our Use Case Diagram is intended to show the various ways that a user can interact with the system. Each case represents a different task that the system can perform for the user. Cases that have exceptional behavior can be extended to allow for these exceptional cases. Behavior that is common to two cases, or behavior that is logically independent from a case can be grouped separately and subsequently included. The descriptions provide a detailed view and explain the Use Cases more thoroughly.

Figure 1: Use Case Diagram for the Online Brokering System
<table>
<thead>
<tr>
<th>Use case: Create Reverse Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong> User</td>
</tr>
<tr>
<td><strong>Type:</strong> Primary</td>
</tr>
<tr>
<td><strong>Description:</strong> The User wants to create a reverse auction.</td>
</tr>
<tr>
<td><strong>Uses:</strong> Create Auction</td>
</tr>
<tr>
<td><strong>Extended by:</strong> none</td>
</tr>
<tr>
<td><strong>Extends:</strong> none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case: Create Regular Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong> User</td>
</tr>
<tr>
<td><strong>Type:</strong> Primary</td>
</tr>
<tr>
<td><strong>Description:</strong> The User wants to create a regular auction</td>
</tr>
<tr>
<td><strong>Uses:</strong> Create Auction</td>
</tr>
<tr>
<td><strong>Extended by:</strong> none</td>
</tr>
<tr>
<td><strong>Extends:</strong> none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case: Create Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong> User</td>
</tr>
<tr>
<td><strong>Type:</strong> Primary</td>
</tr>
<tr>
<td><strong>Description:</strong> The User provides information that is common to both types of auction to create an auction</td>
</tr>
<tr>
<td><strong>Uses:</strong> none</td>
</tr>
<tr>
<td><strong>Extended by:</strong> none</td>
</tr>
<tr>
<td><strong>Extends:</strong> Create Reverse Auction, Create Regular Auction</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case: Cancel Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong> User</td>
</tr>
<tr>
<td><strong>Type:</strong> Primary</td>
</tr>
<tr>
<td><strong>Description:</strong> The User wants to cancel an auction</td>
</tr>
<tr>
<td><strong>Uses:</strong> none</td>
</tr>
<tr>
<td><strong>Extended by:</strong> none</td>
</tr>
<tr>
<td><strong>Extends:</strong> none</td>
</tr>
<tr>
<td>Use case</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Actors</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>Uses</td>
</tr>
<tr>
<td>Extended by:</td>
</tr>
<tr>
<td>Extends</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description:</td>
<td>The User wants to login to the site. They must enter a valid username and password.</td>
</tr>
<tr>
<td>Uses</td>
<td>none</td>
</tr>
<tr>
<td>Extended by:</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>Logout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description:</td>
<td>The User wants to leave the site.</td>
</tr>
<tr>
<td>Uses</td>
<td>none</td>
</tr>
<tr>
<td>Extended by:</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>Join Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Type</td>
<td>Primary</td>
</tr>
<tr>
<td>Description:</td>
<td>The User wants to join an auction. They can now participate in the auction, and will be informed of new bids.</td>
</tr>
<tr>
<td>Uses</td>
<td>none</td>
</tr>
<tr>
<td>Extended by:</td>
<td>none</td>
</tr>
<tr>
<td>Extends</td>
<td>none</td>
</tr>
<tr>
<td>Use case</td>
<td>Bid On Auction</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>The User wants to bid on the current joined auction.</td>
</tr>
<tr>
<td><strong>Uses:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>Withdraw From Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>The User wants to withdraw from the current joined auction.</td>
</tr>
<tr>
<td><strong>Uses:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>Winning Bid</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>Winning Bid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>If the User holds the winning bid in the current joined auction, they will be allowed to withdraw, but they are not relieved of their obligations, should they win.</td>
</tr>
<tr>
<td><strong>Uses:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use case</th>
<th>Win Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>Primary</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>At the end of an auction the user holds the winning bid.</td>
</tr>
<tr>
<td><strong>Uses:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Extended by:</strong></td>
<td>Reserve Not Met</td>
</tr>
<tr>
<td><strong>Extends:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>
### Use case: Reserve Not Met

**Actors:** User  
**Type:** Primary

**Description:** The User held the winning bid at the end of an auction, but the reserve for that auction was not met, so there is no winner for that auction.

**Uses:** none  
**Extended by:** none

**Extends:** Win Auction

### Use case: Check Current Bids

**Actors:** User  
**Type:** Primary

**Description:** The User would like to check his/her bid history.

**Uses:** none  
**Extended by:** none

**Extends:** none
### 4.2 Object Model

#### 4.2.1 Class Diagram

The class diagram shows the different objects involved in defining the system. A class is broken up into three sections. The top section is the name of the class, the middle section contains the attributes that the class has and the bottom section contains all of the operations that the class has. A class diagram also shows how classes are associated with one another. This class diagram (Figure 2) contains the following five classes: Online Brokering System, Auction, Auctioneer, User, Item and Bid. Online Brokering System is the overall system and is made up of Auctions and Users. A User can create or participate in an auction. A User also may place a Bid and that Bid is placed in an Auction. An Item is what is being auctioned in an auction. The classes, attributes and operations are further defined in the data dictionary contained in the following section of the document.

![Class Diagram for the Online Brokering System](image)

Figure 2: Class Diagram for the Online Brokering System
4.2.2 Data Dictionary

The data dictionary defines each term in the class diagram and provides explicit clarification for any system-specific uses of broad terms (such as regular and reverse, referring to auctions). It explains interactions, and it also uses reader-friendly cross-references to other entries for easy understanding.

- Online Brokering System (OBS): The Online Brokering System is an E-system to facilitate the easy exchange of goods. It consists of Auctions, Auctioneers, and Users.

  - Methods:
    * login: The login method authenticates a User’s name and Password to authorize them to use the Online Brokering System.
    * create_auction: The create_auction method creates an Auctioneer, which oversees/manages the Auction and everything associated with it.
    * register: The register method stores the User’s information in the Online Brokering System’s database; this is how the User becomes known to the Online Brokering System.
    * logout: This method logs the User out from the Online Brokering System, keeping their information secure. It effectively withdraws the User from any joined Auctions, but does not relieve them of any financial obligation should any of their Bids win an Auction.
    * check_history: This is a method called by the User to display their bid history for all Auctions they joined.

  - Cross References: User, Password attribute (User), check_credit method (User), cancel method (Auctioneer), AuctionID attribute (Auction)

- Auctioneer: An Auctioneer is a piece of code that manages an Auction. One Auctioneer exists for each Auction. The Auctioneer makes sure that Auctions begin and end on time and handles the Users joined with the Auction.

  - Methods:
    * add_bidder: This method adds the User’s name to the list of Users joined with the Auction; it allows User to place Bids with the Auctioneer.
    * cancel: This method is called by the Online Brokering System when extenuating circumstances allow for an externally site-approved Auction cancel.
    * end: This method is called by Auction when the EndTime has been reached or the Timer times out. It triggers the Auctioneer to call the notify_end method of User to notify the joined Users of the Auction’s end, and determine a winner.
    * remove_bidder: This method is called when a User requests to withdraw from an Auction or is being removed for some other reason, such as an updated credit rating.
    * place_bid: This method checks a Bid submitted by a User against the ReservePrice (if one is present) and the BestBid; if the submitted Bid is better, then BestBid is replaced, and all joined Users are notified.
    * start: This method is called by the Auction when its StartTime is reached. The Auctioneer opens the Auction and begins accepting join requests and Bids.
- Cross References: Online Brokering System, login method (Online Brokering System), Auction, User, notify_end method (User), update_bid method (User), check_credit method (User)

* Auction: A real-time, Auctioneer-initiated object that represents an event during which an Item is sold or purchased.

- Attributes:
  * EndTime(int): EndTime is the User-specified ending time for the Auction.
  * AuctionID(int): AuctionID is the Auction's identification number (assigned by Brokering System inside method create_auction).
  * ReservePrice(int): This is the minimum Bid allowed in a Regular Auction and the maximum Bid allowed in a Reverse Auction.
  * StartTime(int): StartTime is the User-specified starting time for the Auction.
  * Type(bool): Stores Reverse or Regular specification for Auction.
  * BestBid(int): BestBid is the current winning Bid in any Auction. highest Bid (Regular Auction); lowest Bid (Reverse Auction)
  * InitiatorID(int): This is the identification number of the User who started the Auction.
  * OwnerOfBestBid(int): This is the identification number of the User who currently holds the BestBid.
  * Timer (int): This is the countdown for the Auction; the Auction ends when this reaches a set value or the EndTime is reached. Time is kept military style so there is no confusion with AM and PM.

- Methods:
  * reset_timer: This method allows the Timer to reset when new Bids are placed.

* Cross References: Online Brokering System, create_auction method (Online Brokering System), logout method (Online Brokering System), Auctioneer, add_bidder method (Auctioneer), end method (Auctioneer), remove_bidder method (Auctioneer), start method (Auctioneer), Regular Auction, Reverse Auction, Bid, AuctionID attribute (Bid), Item, User, notify_end method (User), check_credit method (User), update_bid method (User)

* Regular Auction: An Auction in which an Item is offered, and the highest Bid wins.

- Cross-References: ReservePrice attribute (Auction), Type attribute(Auction), BestBid attribute (Auction)

* Reverse Auction: An Auction in which an Item is solicited, and the lowest Bid wins.

- Cross-References: ReservePrice attribute (Auction), Type attribute (Auction), BestBid attribute (Auction)
• Bid: Monetary offer made by a User for the Item in an Auction
  
  – Attributes:
  * BidAmount(int): This is the dollar amount of the Bid.
  * AuctionID(int): This is the Auction identification number with which the Bid is associated.
  * OwnerID(int): This is the ID of the bidding User.
  
  – Cross-References: logout method (Online Brokering System), add_bidder method (Auctioneer), place_bid method Auctioneer), start method (Auctioneer), ReservePrice attribute (Auction), BestBid attribute(Auction), place_bid method (Auction), reset_timer method (Auction), Regular Auction, Reverse Auction, notify_end method (User), update_bid method (User)

• Item: An Item is that which is being sold in an Auction.
  
  – Attributes:
  * Image(char): This is a picture of the Item.
  * Name(char): This is the name of the Item.
  * LongDesc(char): The LongDesc provides detailed information about the Item.
  * ShortDesc(char): The ShortDesc provides an abbreviated description of the Item.
  
  – Cross-References: Auction, Regular Auction, Reverse Auction, Bid

• User: A User is a client that interacts with the Online Brokering System; a User may create an Auctioneer or join an Auction, among other things.
  
  – Attributes:
  * EmailAdd(char): This is the E-mail address of the User.
  * Password(char): Password is a User-defined password for authenticating their use of the Online Brokering System.
  * Telephone(char): This is the Telephone number of the User.
  * ID(int): ID is the User's identification number.
  * Address(char): Address is the User's mailing address.
  * Name(char): This is the User's name.
  
  – Methods:
  * notify_end: This method is called by the Auctioneer when an Auction's EndTime is reached or its Timer times out. It notifies the User of the Auction's end, what User won (if anyone), and what the winning Bid was. If the ReservePrice was not met, the Auction has no winner, and all joined Users and the Auction's creator are notified. If a User wins an Auction, this is a special case of the notify_end method, and the User will be prompted somehow for their shipping address, payment method, etc.
  * check_credit: This method is called by the Online Brokering System or an Auctioneer to verify the User's credit rating before allowing them to use the site or join/bid on an Auction, respectively.
  * update_bid: This is a method called by the Auctioneer to notify Users joined with the Auction that the BestBid has been updated.
4.3 Dynamic Model

4.3.1 State Diagrams

A state diagram is used to show what states a class can have, what it can be doing, and how it can get from state to state (transitions). A transition is a directed line that shows what events need to occur for an object to get from one state to another.

**User State Diagram** A user starts out in an idle state and can only register or log in. To register or log in the user sends an appropriate message to the Online Brokering System (OBS) and awaits a response. Once logged in a user can create auctions by sending a message to the OBS, or log into an existing auction by sending a message to an auctioneer. Once a user has joined an auction it can place bids in that auction by sending a message to the auctioneer. The user is notified of new bids by messages sent from the auctioneer. Any time after joining an auction, the user can withdraw from it. Any time after logging in, the user can log out.

![User State Diagram](image-url)

Figure 3: User State Diagram
**Auctioneer State Diagram**  An auctioneer starts in an idle state and waits for a start message from an auction before allowing users to join. Once the auctioneer has been started it allows users to join, place bids and withdraw. When a user joins or withdraws they are added to or removed from the auctioneer's list of active bidders. When user places a bid the auctioneer tells the auction object to reset its timer, and notifies each user in its list of active bidders of the new bid. When the auctioneer receives an end message from the auction, or a cancel message from the OBS all active bidders are notified that the auction is ending and the auctioneer moves into a closed state.

![Auctioneer State Diagram](image)

**Figure 4: Auctioneer State Diagram**

**Auction State Diagram**  An auction starts in an idle state and waits until the start time is reached before sending the start message to the auctioneer and moving into a running state. In the running state the auctioneer watches a timer and waits until the end time is reached, or the timer reaches zero. It resets the timer when it receives a reset message from the auctioneer. When the end time is reached or the timer runs out the auction sends the end message to the auctioneer and moves into a closed state.

![Auction State Diagram](image)

**Figure 5: Auction State Diagram**
4.3.2 Sequence Diagrams

The sequence diagrams show specific scenarios that the system will go through. The objects are located across the top of the diagram and time flow runs from top to bottom of the diagram. There are also lines that connect the objects that display messages that are sent at each particular point in time.

**Register** The Register sequence diagram (Figure 6) shows the sequence of activities involved in registering with the system. It shows how the different classes in the system communicate with each other as a client registers a new account then logs into and out of the system.

![Register Sequence Diagram](image)

**Figure 6: Register Sequence Diagram**
**Create Regular Auction**  The Create Regular Auction sequence diagram (Figure 7) demonstrates the process of creating a regular or reverse auction after a client has logged into the system. The client sends the message `user_create_auction_rqst` to the user class, which then sends `create_auction` message to the Online Brokering System. Finally, Online Brokering System creates the auction.

![Diagram of Create Regular Auction sequence diagram](image)

Figure 7: Create Regular Auction Sequence Diagram
**Place Bid**  The Place Bid sequence diagram, shown in Figure 8, illustrates how a user logs in to the system, joins an auction and places a bid. Before a client places a bid in an auction, it has to send an add_bidder message to the auctioneer class in order to join the auction. After responding to the place_bid message as indicated in this diagram, the auctioneer sends the reset_timer message to the auction class and sends update_bid messages to the users involved in the auction.

![Sequence Diagram](image)

Figure 8: Place Bid Sequence Diagram
5 Model Checking

We were supplied with two state diagrams to model check. It was our job to further analyze these diagrams visually and then to check these models using simulation and verification. We found some problems when first looking at the models and confirmed these problems that we encountered by encoding the diagrams into Promela and then running Xspin. We have included our Promela code in Appendix A.

5.1 Newbid == Maxbid

The Auctioneer will become deadlocked when it receives a newbid that is equal to the current maxbid. This is because the transitions from the Save_Bid state are guarded on newbid being greater or less than maxbid, but not equal to it. We have included an event trace diagram from XSpin (Figure 9) detailing this scenario.

![Diagram](image)

Figure 9: Newbid == Maxbid event trace from XSpin
5.2 Ending an Auction

When the auctioneer receives the auction_time_elapse message it sends the end_auction message to all of the clients. However, a client must be in the Idle state in order to receive the end_auction message. Therefore, if any of the clients are in any state other than Idle, they will not receive the end_all message and will never enter the terminal state. We have included an event trace diagram from XSpin (Figure 10) detailing this scenario.

![Event Trace Diagram]

Figure 10: End auction event trace from XSpin
A Promela Code

/* states of clients */
mtypes = { HI_BID, IDLE, USER_BID, GET_BID, SENT_BID};

/* messages sent to clients */
mtypes = { high_bid, user_bid_request, bid_ok, time_out, ack, end_auction };

/* states of auctioneer */
mtypes = { WAIT, CHECK_OK, SAVE_BID, HIGH_BID, END_ALL };,

/* messages sent to auctioneer */
mtypes = { bidrqst, bidmsg, auction_time_elapse };,

chan clientevent = [0] of {mtypes};
chan auctioneerevent = [0] of {mtypes};

mtypes clientstate;
mtypes auctioneerstate;

int newbid;
int maxbid = 100;

init
{
    clientevent!user_bid_request;

    /* Error 1: newbid == maxbid */
    /*
    newbid = 100;
    */

    /* Error 2: End auction */
    /*
    newbid = 110;
    if
        ::auctioneerevent!auction_time_elapse;
        ::clientevent!timeout;
    fi
    */
}

active proctype client()
{
    clientstate = IDLE;
do
        :: clientstate=IDLE ->

if ::clientevent?user_bid_request ->
    clientstate = USER_BID;
::clientevent?high_bid->
    clientstate = HI_BID;
::clientevent?end_auction->
    break;
fi;

:: clientstate==HI_BID ->
    printf("Display high bid") ->
    clientstate = IDLE;

:: clientstate=USER_BID->
auctioneer_event!bidrqst->
clientevent?bid_ok ->
    clientstate = GET_BID;

:: clientstate == GET_BID ->
    printf("Getting bid amount") ->
auctioneer_event!bidmsg->
    clientstate = SENT_BID;

:: clientstate == SENT_BID ->
    if :: clientevent?time_out->clientstate = USER_BID;
    :: clientevent?ack->clientstate = IDLE;
    fi;
    od
}

active proctype auctioneer()
{
    auctioneerstate = WAIT;
    do
        :: auctioneerstate=WAIT ->
            if :: auctioneer_event?auction_time_elapse ->
                auctioneerstate = END_ALL;
            :: auctioneer_event?bidrqst ->
                clientevent?bid_ok ->
                    auctioneerstate = CHECK_OK;
                fi;

            :: auctioneerstate=CHECK_OK ->
                if :: auctioneer_event?auction_time_elapse ->
auctioneerstate = END_ALL;
:: auctioneer?bidmsg ->
auctioneerstate = SAVE_BID;
fi;
:: auctioneerstate=SAVE_BID ->
if
:: auctioneer?auction_time_elapse ->
auctioneerstate = END_ALL;
:: clientevent!ack ->
if
:: newbid > maxbid -> auctioneerstate = HIGH_BID;
:: newbid < maxbid -> auctioneerstate = WAIT;
fi;
fi;
:: auctioneerstate=HIGH_BID ->
printf("maxbid = newbid") ->
maxbid = newbid ->
clientevent!high_bid->
if
:: auctioneer?auction_time_elapse ->
auctioneerstate = END_ALL;
:: auctioneerstate = WAIT;
fi;
:: auctioneerstate=END_ALL ->
clientevent!end_auction ->
break;
od
}