

Realtime Commercial Bidding System
Proposed Client - Auctioneer Interaction
Blackbit Trading, Inc.
Mason, MI

January 17, 2002

Our customer has gotten so enthused with your use of UML diagrams and OO analysis that she has taken stab at the client-auctioneer interaction in the form of class and state diagrams. What your team needs to do is use model checking to explore the behavior of these two proposed classes. No doubt there are problems with the state diagrams. The two classes run concurrently, and you should explore what happens when there is more than one client. Use at least one LTL temporal logic check. You'll be looking for problems like deadlock, dropped bids, or just plain bugs. Even if you find a problem by visual inspection, verify that it is a problem with a model checking run. A few checks are sufficient (two or three), however, the more subtle the problems, the better. You do not need to exhaustively debug the diagrams, nor do you need to fix them.

Here is what the customer sent:

This document describes a state diagram for an Auction client and an Auctioneer. The client accepts events from the following sources:

Event	Source	Action
user_bid_request	Client interface screen	Signals client that a bid is ready
bidok	Auctioneer	Response to bidrqst, means we can send bid.
ack	Auctioneer	Previous message received
timeout	System	Pre-determined time limit has elapsed
high_bid(amt)	Auctioneer	"bid" is the new system-wide high bid
end_auction	Auctioneer	The auction is over

The Auctioneer accepts events from the following sources:

Event	Source	Action
bidrqst	Client	Check to see if it's ok to accept bid from client
bidmsg(newbid)	Client	Process new bid "newbid"
auction_time_elapse	Auction Site	Stop all clients

Normally the user clicks "enter bid" on the client interface screen when she wants to enter a new bid. This generates the "user_bid_request" event, which signals the client to ask the Auctioneer if it can send a new bid. When the Auctioneer replies "OK", the bid is sent, and an "ack" is awaited. If a sufficiently long time passes, the system generates a "timeout" event within the client.

When the client receives a "high_bid(bid)" event, some client has achieved a new high bid (maybe this client), and the high bid should be displayed to the user. When the client receives a "end_auction" event, the client should shut down in an orderly manner.

The Auctioneer waits for "bidrqst" messages from clients. When it gets one, it checks to make sure the client is qualified to bid (detail not shown in these diagrams). If qualified, the Auctioneer replies with permission to send the bid (bidok message), and waits for the bid. When the bid is received, it is checked for a new maximum bid, and if the maximum, the Auctioneer notifies all clients that there is a new high bid.

When the event "auction_time_elapse" occurs, the Auctioneer must tell all clients to shutdown, then exit itself.

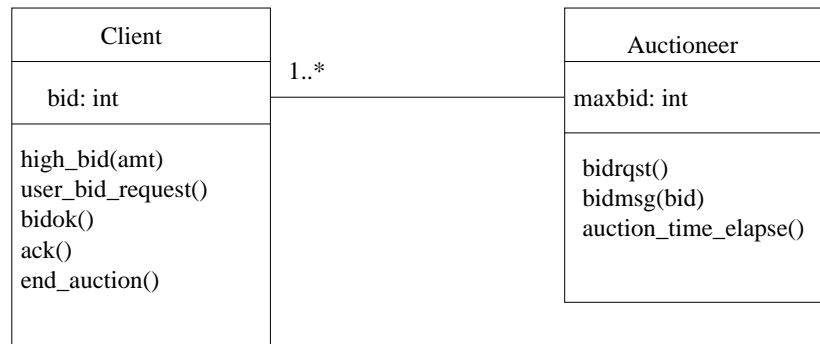


Figure 1: The abbreviated class diagram

An abbreviated class diagram for these two classes is shown in Figure 1. Figure 2 shows the client state diagram and Figure 3 shows the auctioneer state diagram.

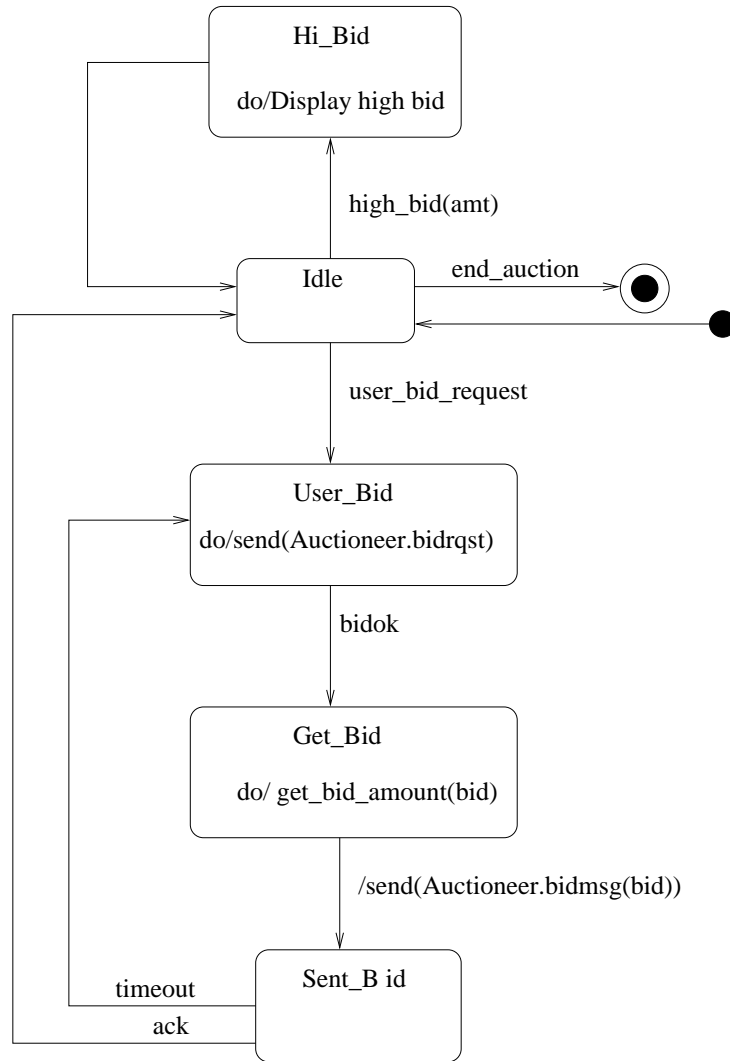


Figure 2: The client state diagram

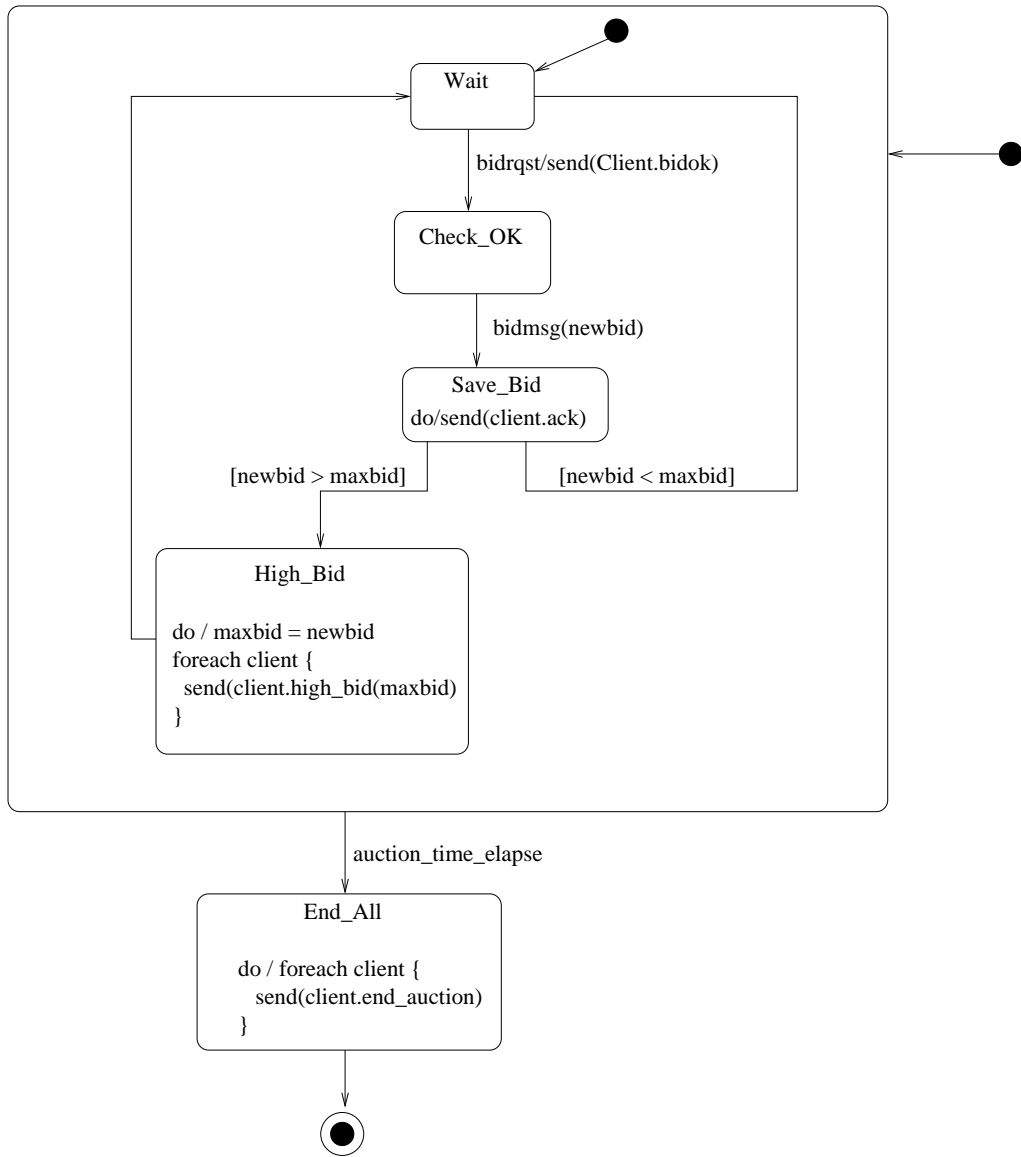


Figure 3: The Auctioneer state diagram