Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Elisabetta Di Nitto, Matteo Rossi and Damian Tamburri

20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Software Engineering 2 – Written Exam 2 (WE2)

February 14th, 2022

Last Name

First Name

Id number (Matricola)

Notes

1. Remember to write your name and Id number (matricola) on each piece of paper that you hand in.
2. You may use a pencil.
3. Incomprehensible handwriting is equivalent to not providing an answer.
4. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden, with the exception of an ebook reader.
5. The exam is composed of 2 parts, one focusing on requirements, and one focusing on design. Read carefully all points in the text!
6. **Total available time for WE2: 1h and 30 mins**

System Description: AuctionManager

We have been tasked with developing a system, **AuctionManager**, for the management of auctions of items (e.g., paintings, pieces of art, memorabilia, etc.).

Each auction involves a batch of items, which are bid upon one by one. For each successive item, the auction director opens the bidding; then, participants in the auction can bid on the item; if no new bid arrives within 4 minutes since the last one, the bidding closes, and the highest bid wins the right to buy the item (notice that each new bid must be higher than the previous one).

AuctionManager must handle two types of participants to the auction: people who bid remotely (who do not physically attend the auction), and people who bid in person, by physically attending the auction. In the case of people who bid in person, the bidder raises a panel with an id of the bidder and the amount bid; in this case the bidding is tracked by an auction assistant (possibly the auction director her/himself), who records the bid in the application. Whenever a new bid arrives (either in person, or remotely), within 1 second the new highest bid is shown both to remote bidders and on a panel, physically, in the auction room.

The system allows auction organizers to set up the auction, by inserting/modifying the information about the auctioned items and the order in which the items are going to be shown in the auction.

The system also allows third-party applications to access the information about future and past auctions, to allow interested stakeholders to build statistics regarding auctions (e.g., determine which items are most likely to draw interests from bidders), and to advertise auctions.

Part 1 Requirements (7 points)

RASD_Q1 (2 points)

With reference to the Jackson-Zave distinction between the world and the machine, identify the relevant world phenomena for *AuctionManager*, including the ones shared with the machine, providing a short description if necessary. For shared phenomena specify whether they are controlled by the world or the machine. Focus on phenomena that are relevant for the management of auctions and bids.

RASD_Q2 (3 points)

Describe through a UML Class Diagram the main elements of the *AuctionManager* domain.

RASD_Q3 (2 points)

Consider the following goal of the *AuctionManager* system:

G1: The participants to the auction always have updated information about the bids of the current auctioned item.

Define in natural language suitable domain assumptions and requirements to guarantee that the *AuctionManager* system fulfills goal *G1*.

Part 2 Design (7 points)

DD_Q1+Q2 (2+3 points)

Assuming you need to implement system *AuctionManager* analyzed above:

1. Identify the most relevant components and interfaces and describe them through a UML Component diagram.
2. Provide a brief description of each identified component.
3. For each component, list the operations it provides through its interfaces.

In defining the components and the operations, focus on those that are relevant to set up and manage auctions.

NB: For each operation, you do not need to precisely specify its parameters; however, you should give each operation a meaningful enough name to understand what it does; you can also briefly describe what information operations use/produce.

DD_Q3 (2 points)

Describe through a UML Sequence Diagram the interaction that occurs between the system components when an in-person participant to the auction makes a new bid for the current item and the in-room panel is updated with the new information (do not consider the update of the information to remote participants).

Solutions

RASD_Q1

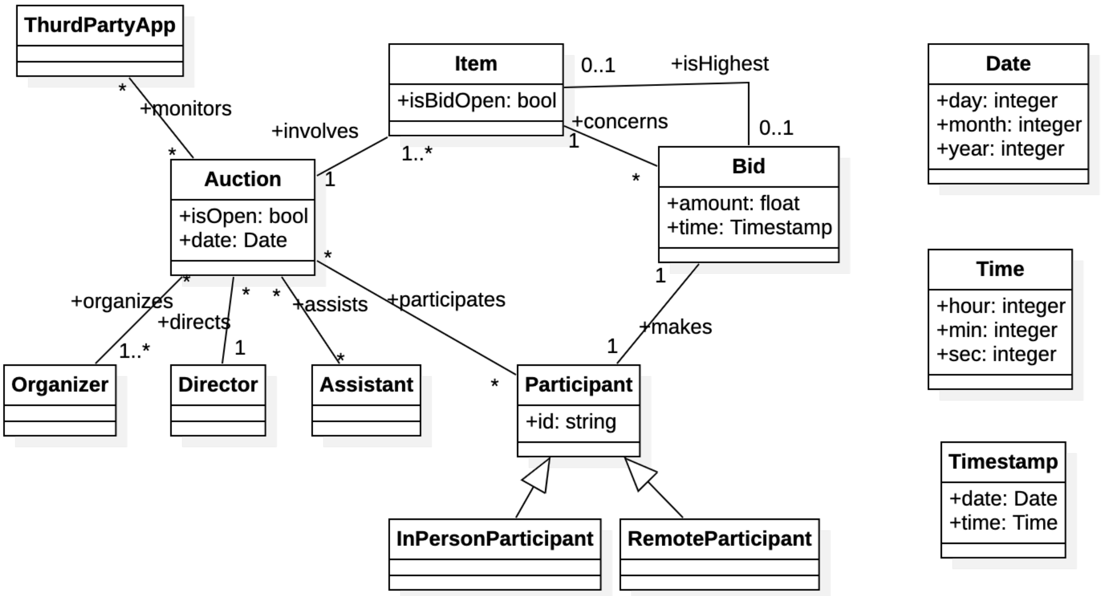
World phenomena:
Participant takes auction in person
Participant decides to take auction from remote
Auction assistant shows item in the room
In-person participant bids on item
Auction assistant sees the bid in the room
Third-party application builds statistics regarding auctions
Third-party application advertises auction

Shared phenomena, world-controlled
Auction director opens bidding on item
Remote auction participant bids on item
Auction assistant inputs bid from auction room
Auction organizer creates auction
Auction organizer adds item to auction
Auction organizer modifies item in auction
Auction organizer deletes item from auction
Third-party application retrieves information of auction

Shared phenomena machine-controlled
System selects the next item to be auctioned
System closes bidding on item after timeout expires
System shows new highest bid to remote bidder
System shows new highest bid on panel in auction room

RASD_Q2

A possible domain model for the *AuctionManager* system is the following:



The relationship among *Organizer*, *Director*, *Assistant* is left unspecified in this model. *Director* and *Assistant* could specialize *Organizers*, but other relationships are also possible and the text does not define them explicitly.

RASD_Q3

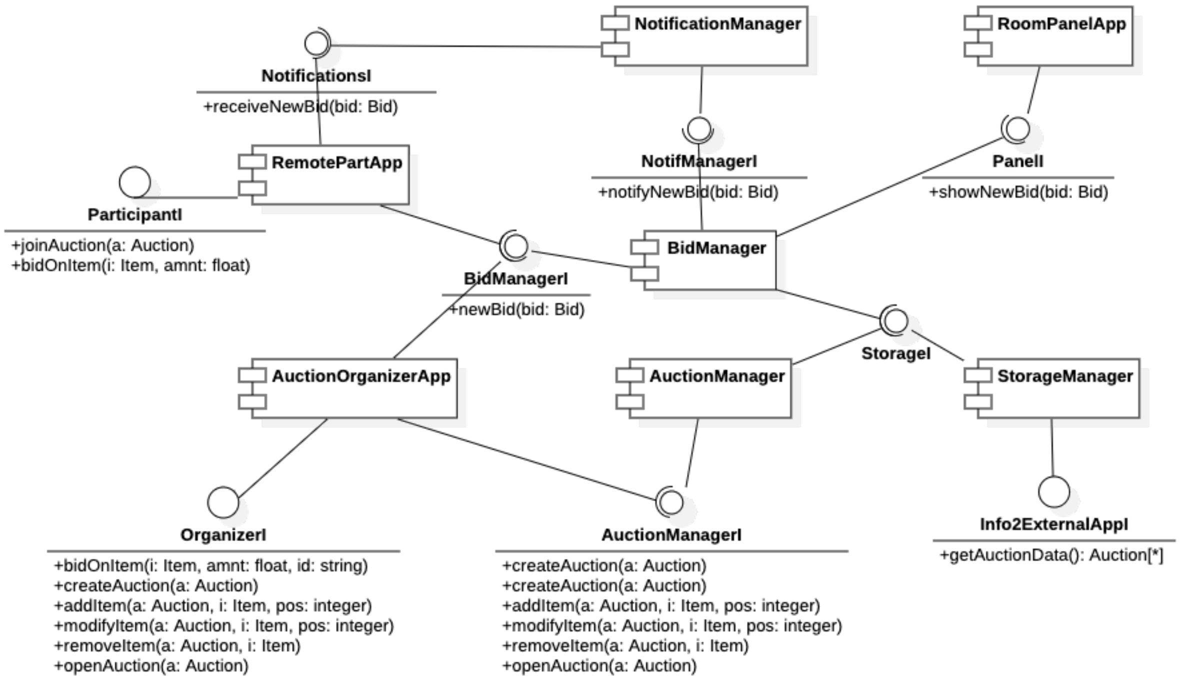
Requirements
R1: The system shall allow remote bidders to make bid for current item (the bid includes the amount bid on the item).
R2: The system shall allow assistants to input the information about a bid coming from the room.
R3: When a new bid (either remote or in-person) is recorded, within 0.5 seconds the system displays the new bid on a panel in the room.
R4: When a new bid (either remote or in-person) is recorded, within 0.5 seconds the system sends a notification to all remote participants about the new bid.
R5: When a notification of a new bid is received by the remote participant, the system displays it for the user.
R6: The system accepts a new bid only if it is higher than the current highest one.
R7: The system allows the auction director to open the bidding on a new item.
R8: If no new bid arrives within 4 minutes from the last one, bidding on the current item closes.

Domain assumptions
DA1: When in-person bidder makes bid, assistant correctly captures it.
DA2: The latency of the underlying communication network is such that delivery of a bid to remote participants and to the room panel does not take longer than 0.5 seconds.

R1-R2, plus DA1 guarantee that the system has updated information about the current bid. R3 then guarantees that the people attending the auction in person receive the information through the room panel in time. Similarly, R4-R5, combined with DA2, guarantee that the information also reaches remote bidders in a timely fashion. R6, R7, R8 are auxiliary requirements with respect to goal G1, they ensure that bids are correct, and that the bidding is also correctly opened and closed (notice, though, that this does not directly influence the fact that participants have updated information about the bids).
Notice also that the notification transmission mechanism (e.g., delivery through the internet) is something that is not under the control of the application, so we make a reasonable assumption about the delivery time of a notification through the internet, and state the requirement concerning the timing of the notification accordingly.

DD_Q1+Q2

A possible breakdown of the system into components is the following (notice that the types of parameters is only indicative, to give an idea of the kind of information that is handled by each operation):


NotificationManager
RoomPanelApp
NotificationsI
+receiveNewBid(bid: Bid)
NotifManagerI
+notifyNewBid(bid: Bid)
PanelI
+showNewBid(bid: Bid)
RemotePartApp
ParticipantI
+joinAuction(a: Auction)
+bidOnItem(i: Item, amnt: float)
BidManager
BidManagerI
+newBid(bid: Bid)
StorageI
AuctionOrganizerApp
AuctionManager
StorageManager
OrganizerI
+bidOnItem(i: Item, amnt: float, id: string)
+createAuction(a: Auction)
+addItem(a: Auction, i: Item, pos: integer)
+modifyItem(a: Auction, i: Item, pos: integer)
+removeItem(a: Auction, i: Item)
+openAuction(a: Auction)
AuctionManagerI
+createAuction(a: Auction)
+createAuction(a: Auction)
+addItem(a: Auction, i: Item, pos: integer)
+modifyItem(a: Auction, i: Item, pos: integer)
+removeItem(a: Auction, i: Item)
+openAuction(a: Auction)
Info2ExternalAppI
+getAuctionData(): Auction[*]

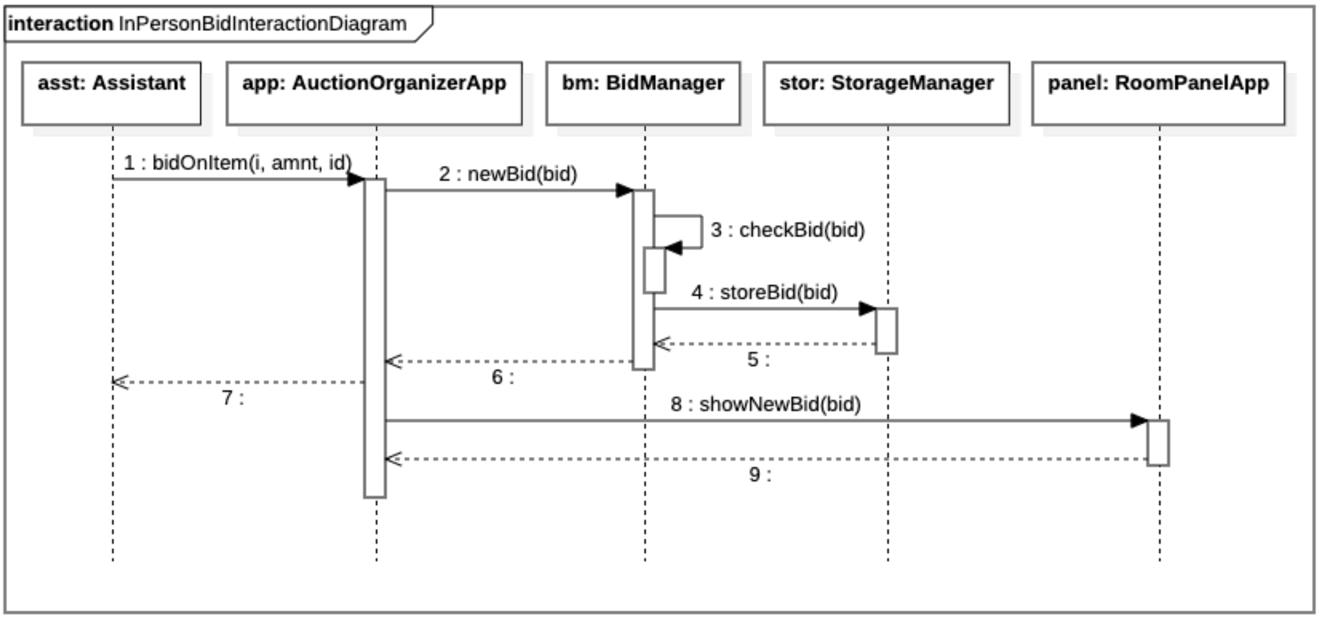
Clearly, the remote participants must have their own application (*RemotePartApp*), which one can imagine will be completely different from the one of auction organizers (it most probably will run on different devices). Auction organizers have their own application (*AuctionOrganizerApp*). In this case, one could choose whether to have a single application for organizers, directors, and assistants, or separate ones; the proposed solution follows the first approach (a single application for all). In addition, a separate module (*RoomPanelApp*) handles the displaying of bids on the panel in the room.

Module *AuctionManager* handles the configuration of the auctions by the organizers, whereas *BidManager* handles the bids that come from both remote participants and from the auction room (in which case they are inserted by the auction assistants/organizers).

NotificationManager and *StorageManager* provide services to the other components. In particular, the *NotificationManager* module handles notifications sent to remote participants. The *StorageManager* module, instead, provides other modules with access to the storage of the relevant data (auction information, items, etc.). The *StorageManager* module provides two interfaces, *StorageI* and *Info2ExternalAppI*. Interface *StorageI* provides the internal modules with functions to create, modify, delete the data (these functions are not detailed in the diagram for simplicity, as they are quite standard and depend on the data stored in the storage). Interface *Info2ExternalAppI*, instead, allows third parties to access the data concerning the auctions (this interface is only sketched in the diagram; its main function is clearly providing data to third parties, but one can imagine that it will also include functions and mechanisms to authenticate applications, etc.).

The lists of operations are directly provided in the diagram (except for the *StorageI*, as explained above).

DD_Q3


interaction InPersonBidInteractionDiagram
asst: Assistant
app: AuctionOrganizerApp
bm: BidManager
stor: StorageManager
panel: RoomPanelApp
1 : bidOnItem(i, amnt, id)
2 : newBid(bid)
3 : checkBid(bid)
4 : storeBid(bid)
5 :
6 :
7 :
8 : showNewBid(bid)
9 :