A Pattern Language for Online Auctions Management *

Reginaldo Ré†
Rosana T. V. Braga‡
Paulo Cesar Masiero§
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - Brazil
{rr, rtvb, masiero}@icmc.sc.usp.br

Abstract

Online auctions are gaining popularity as they provide an excellent means of selling goods. Their advantages include: reducing transaction costs, selling goods that are difficult to sell through the ordinary market channels, and increasing the circle of potential customers. This paper presents a pattern language for the Web auction application domain. The pattern language has ten analysis patterns that offer solutions to support all the basic functionalities of this domain. Its development was based on several existing Internet auction systems and its application can guide the developer during analysis of such systems.

Keywords: pattern languages, software patterns, online auctions.

1 Introduction

Electronic commerce has been expanding recently, gaining more and more sellers and buyers. Web-based Information Systems are making a revolution in the electronic commerce. Online auctions, which are a type of electronic commerce, are rapidly gaining popularity [10, 11, 18]. Online auction systems are classified based on different aspects such as: bid definition, specification of auction rules, and definition of transactions. Several reasons motivate their use, among which are the ability to increase the number of potential customers, the low prices for the consumers, the low transaction costs, and the short period of time in which they can be done [9, 13].

The Pattern Language for Business Resource Management deals with trading, location and maintenance of business resources [2]. The Pattern Language for Online Auctions Management was developed in this context, to help the development of systems concerned with trade management through auctions on the Web. The Pattern Language for Online Auctions Management development was based on three existing Internet auction systems [15]. The systems studied were eBay [1], iBazar[2], and Arremate.com[3]. We have done the reverse engineering of these systems based on their user interfaces, as we did not have access to their documentation or source code. Then we have developed models to express their functionality, using our own knowledge about auction systems, as well as information systems theory and existing references about auction systems [13, 18]. The patterns were abstracted from these models, considering the common functionality among the three systems. They capture the functional aspects involved in an online auction system, rather than design and implementation issues. The examples provided in this paper were also extracted from these models.

The structure and the pattern language examples are expressed using UML. The language suggests several basic attributes and methods that may be necessary during instantiation. Canonic methods are omitted because they would increase complexity without adding effective gains to the class model [4, 14]. We use the "operation" concept proposed by Larman [14]. The system output operations are denoted by the "!" prefix and the system input operations by the "?" prefix. When the call message is sent to a collection of objects, instead of to a single instance, we use the "#" prefix.

2 Pattern Language Overview

Figure 1 shows the relationship among the language patterns. The ten patterns can be split in two main categories. The first category - Required Patterns - has patterns to address the essential requirements so that a resource can be auctioned. These required patterns must be applied to all auction systems.

†Financial support from Fapesp Process n. 00/06446-0.
‡Financial support from Fapesp Process n. 98/13588-4.
§Financial support from Fapesp/CNPQ.

[1] www.ebay.com
[2] www.ibazar.com.br
[3] www.arremate.com.br

The second category - Optional Patterns - concerns patterns to address some desirable but not strictly necessary features of auction systems. Figure 1 presents an overview of the sequence of application for each pattern. Table 1 presents a summary of the pattern language and can be used for quick reference.

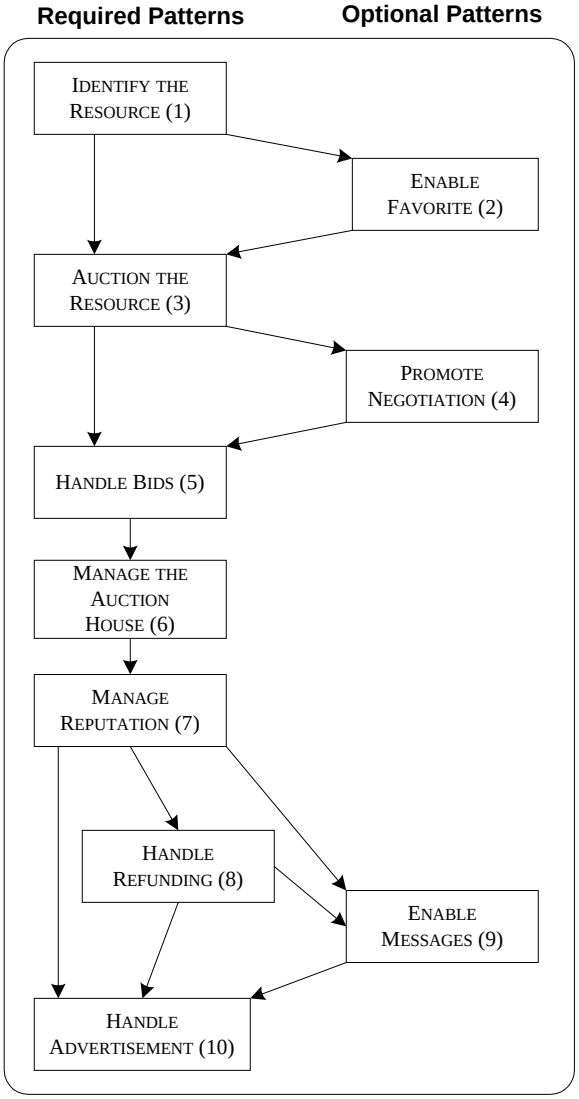

Figure 1. Relationship among the patterns that compose the pattern language

Table 1. Summary of the pattern language

Pattern	Problem	Solution
IDENTIFY THE RESOURCE (1)	How do you represent the business resources auctioned by the system?	section 3.1, page 2
ENABLE FAVORITE (2)	How does your application allow that resource categories of more interest be established for the customers?	section 3.2, page 4
AUCTION THE RESOURCE (3)	How do you handle the different types of resource auctions performed by your application?	section 3.3, page 4
PROMOTE NEGOTIATION (4)	How does your application support the negotiation among auction trading parties?	section 3.4, page 7
HANDLE BIDS (5)	How does your application deal with the different types of bids related to the several types of auction?	section 3.5, page 8
MANAGE THE AUCTION HOUSE (6)	How does your system manage the rules followed by the auction house involved in the auctioning process?	section 3.6, page 10
MANAGE REPUTATION (7)	How can your application provide subsidies for the parties to evaluate each other trustability?	section 3.7, page 12
HANDLE REFUNDING (8)	How can your application provide ways to refund fees that were unduely charged?	section 3.8, page 13
ENABLE MESSAGES (9)	How does your application manage the messages sent to customers?	section 3.9, page 14
HANDLE ADVERTISEMENT (10)	How does your application manage auction advertising?	section 3.10, page 15

3 The Pattern Language for Online Auctions Management

3.1 Identify the Resource

Context

Your business system deals with a specific commercial transaction type: auctions. These transactions refer to, for example: products or services trading, like airline tickets, hotel stays, travel packages, cds, books, art items, and collectible items, which can be called resources. These business resources are managed by specific systems that allow trade among customers, which can be persons or organizations. Auction systems support trade among different types of consumers and for a wide variety of resources. In some cases auctions are the best way for trading a resource, such as: when the seller does not know the real resource value, for perishable resources, art items, collectible items, and overstocked products.

Problem

How do you represent the business resources auctioned by the system?

Forces

- Business resources usually have common attributes or qualities. Keeping information about each particular resource is important for the organization that manages these resources.

- Business resources must be classified into several categories. For example, in an auction house for computer products, resources can be firstly grouped by their nature (*hardware* or *software*), in a second level by their type (input peripherals, CPUs or *notebooks*) and, in a third level, by brand or manufacturer. This grouping is necessary for easily finding the desired resources. This justifies the separation in two or more classes. The space necessary to keep these attributes for each resource and the redundance obtained are undesirable. However, the separation may increase the processing time, as a class will need to reference the other, and this reference needs to be cared of by the system. This must be considered when optimizing system performance.

Structure

Figure 2 shows the class diagram for the IDENTIFY THE RESOURCE pattern.

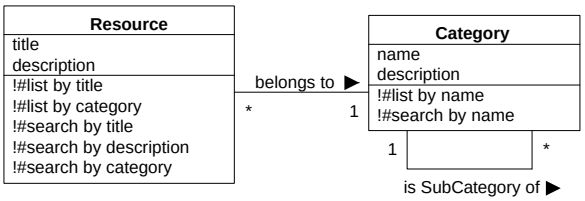

Figure 2. IDENTIFY THE RESOURCE **pattern**

Participants

- *Resource*: represents the business resource managed by the application, defining all its important features. Other attributes may be added according to the particular resource instance, using, for example, the Properties pattern [5].

- *Category*: represents the categories by which resources are grouped, allowing lists of resources by category and search operations by category. Usually, at most three grouping levels are allowed.

Example

Figure 3 shows an example of the IDENTIFY THE RESOURCE pattern used by iBazar. It is a straight application of the pattern, having only one more attribute (`photo`). The [0..1] suffix means that an auctioned product can or cannot have a photo. Arremate.com and eBay have a different type of categorization, shown in the Variants section.

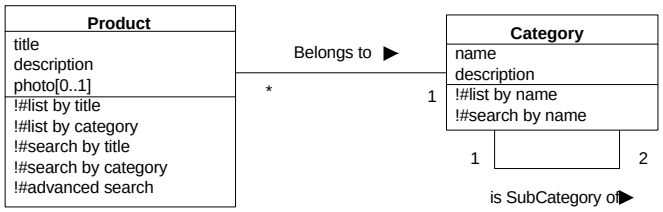

Figure 3. IDENTIFY THE RESOURCE **pattern example**

Variants

This pattern is a variation of the IDENTIFY THE RESOURCE pattern proposed by Braga et al. [2]. The originally proposed pattern presents fewer attributes and other operations. When several grouping levels occur we have nested type objects [12].

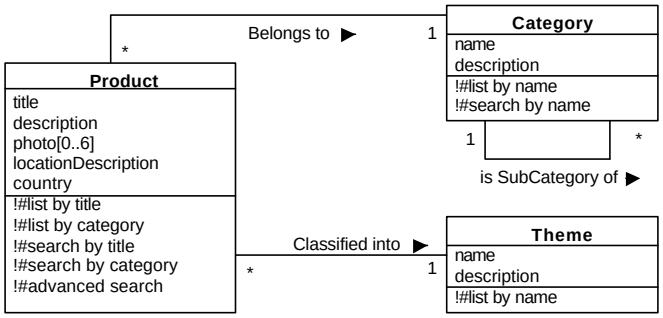

Figure 4. Example of a variant of the IDENTIFY THE RESOURCE **pattern**

To improve categorization, resources may be classified by more than one group, resulting in greater flexibility. Johnson and Woolf [12] call this variant "multiple type objects", similarly to the "nested type objects" presented in our main pattern. A new class is created for each desirable category, linked to the `Resource` class. An example of this pattern is found in eBay. `Product` has two categorization type classes, `Category` and `Theme`, as shown in Figure 4. The `Theme` class groups resources

that have some common characteristics, for example collectible items, football, or cooking, but this categorization is independent from the `Category`. `Products` of different `categories` may belong to the same `Theme`, as for example, a kitchen mixer can be included in the "electrical appliance" `Category` and, at the same time, can be included in the cooking `Theme`. So, this could not be solved by our standard pattern, which allows only hierarchal subcategories.

Related Patterns

This pattern is also an application of the ACCOUNTABILITY pattern [6].

Following Patterns

After using the IDENTIFY THE RESOURCE pattern, try to use the ENABLE FAVORITE. If it is not applicable then use the AUCTION THE RESOURCE pattern.

3.2 Enable Favorite

Context

Your application deals with resources that have already been identified and categorized. In some systems it can be allowed for customers to group some resource categories, because they are the most interesting to them. This option may help to define a consumer purchase profile. Thus, by observing this consumer purchase profile we can send them offers as a way of fostering auctions.

Problem

How does your application allow that resource categories of more interest be established for the customers?

Forces

- It is desirable to offer customers an easy access to interesting resources.
- To store information about customers profile is a good strategy for addressing advertisements of specific resource types. This type of strategy, besides being important for the organization that owns the system, is convenient to customers, giving them only the information about resources that have more chance of being of interest.
- It is important to balance benefits versus drawbacks of offering many offers to the customers, as they may be annoyed by receiving these constant offers. Advertisement should attract customers to frequently visit the auction system site.

Structure

Figure 5 shows the class diagram for the ENABLE FAVORITE pattern.

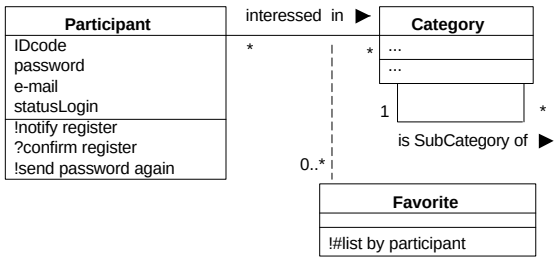

Figure 5. ENABLE FAVORITE **pattern**

Participants

- *Participant*: Represents the party - organization or person - who intends to auction or acquire a resource.
- *Category*: As described in the IDENTIFY THE RESOURCE pattern.
- *Favorite*: Represents the resource categories selected by the participant as interesting.

Example

Figure 6 presents an example of the ENABLE FAVORITE pattern used at the eBay online auction site. The *Favorite* class stores the most interesting resource categories for the customer. eBay states that only four preferred categories may be stored. Both Arremate.com and iBazar do not have the concept of preferred customer categories.

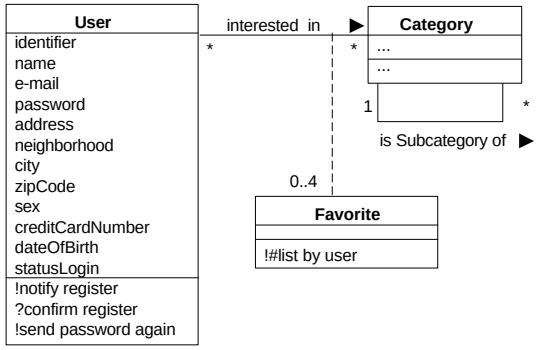

Figure 6. ENABLE FAVORITE **pattern example**

Following patterns

After using the ENABLE FAVORITE pattern, use the AUCTION THE RESOURCE pattern.

3.3 Auction the Resource

Context

Your application deals with resources that have already been identified and categorized. The resource auction may

be considered as a property transference, in which a resource owned by a party becomes owned by another party. When a trade is done through an auction, the resource is put on sale by a trading party and several other parties try to buy it for the lowest possible price. There are several types of auctions that provide various options for the trading parties, each with its own rules to define which of the buying parties will be the winner.

Problem

How do you handle the different types of resource auctions performed by your application ?

Forces

- Information about the participants must be stored, both to supply the information needed for the trading process and for the system functioning.
- It is important that several auction types be available, observing those that are more appropriate to certain types of resource, the quantity of auctioned resources, efficiency, and restrictions imposed by certain auction types, considering the environment in which the trade occurs: the Internet.
- In some cases the source party may want to change information about the auction or even the auction type. It is necessary to establish rules for changing this information so that participants are not harmed.
- Rules for auction cancelation must also be established so that the participants are not harmed.

Structure

Figure 7 shows the class diagram for the AUCTION THE RESOURCE pattern.

Participants

- *Participant*: Represents the party - organization or person - who intends to auction or acquire a resource. It has two specialized classes: *Source Party* and *Destination Party* (see below). Notice that the same participant can play both roles in different auctions. This is guaranteed by the use of the Roles pattern [3, 7]. The `statusLogin` attribute indicates whether the participant has supplied its `IDcode` and `password` and, therefore, can effectively participate in the auctions. It is important to notice that passwords need special treatment during design, through a security policy [19], but we are not considering such issues in this pattern language. This class has some basic attributes, but other attributes may be added, depending on the particular instance of the participant.
- *Participant Role*: Represents the role played by the participant in a specific auction, which can be *Source Party* or *Destination Party* (see below).

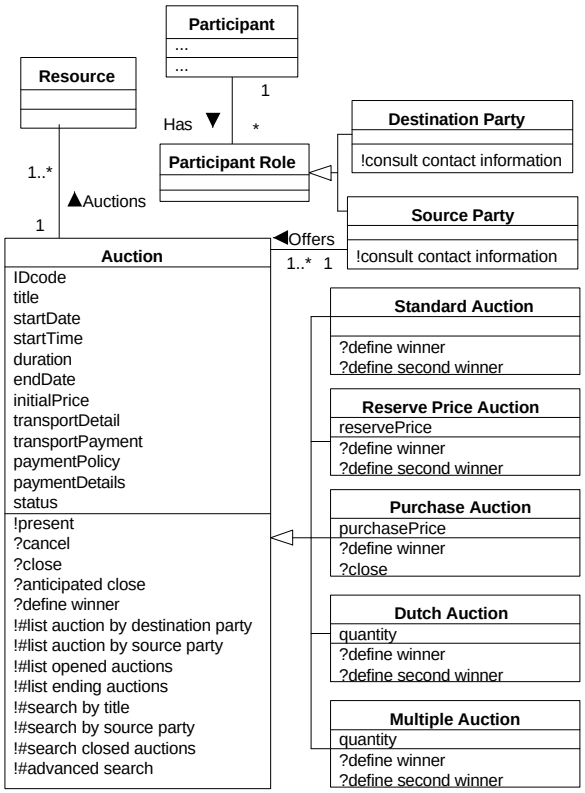

Figure 7. AUCTION THE RESOURCE **pattern**

- *Source Party*: Represents the party that owns the resource before the transaction is concluded.
- *Destination Party*: Represents the party that bids and that may become the resource owner after the auction is closed.
- *Resource*: As described in the IDENTIFY THE RESOURCE pattern.
- *Auction*: Abstract class that represents the details of the resource auction process. This class has all the basic details for an auction to be performed, independently of its type. This class controls the implementation of the change rules, i.e., what information can be changed. There are other attributes to guarantee these rules, such as: how many times information about the auction can be changed, what is the period in which an auction can be changed, and if an auction can be changed when there are no bids. The attributes used as parameters for the auction to work are stored on the `Auction House` class in the MANAGE THE AUCTION HOUSE pattern. Depending on the details and functionalities desired for a particular auction instance, other attributes may be added to this class and to the `Auction House` class. This be-

havior can be implemented using the Strategy pattern [8].

- *Standard Auction*: This is the most common auction type, in which several destination parties bid to acquire a resource offered by a source party. The highest bid wins the auction. After closing the auction and defining the winner, contact data become available for consultation (`consult contact information`). The `Auction` class has an operation (`define second winner`) that allows the source party to conclude the transaction with the second winner, in case any problem occurs with the first winner (`define winner`). The second winner is the source party that offered the second highest bid. This auction type does not allow that several resource items be auctioned.

- *Reserve Price Auction*: In this auction type the source party defines, besides the initial price, a reserve price that is not visible to other auction participants. Bids that are lower than the reserve price are accepted, but the source party does not need to trade the resource if the auction is closed with a value below the reserve price. In this case the auction will be closed without a winner and auction participants will not receive contact information. In order to have a winner, a reserve price auction must have a final bid that meets or exceeds the reserve price. The reserve price can be decreased by the source party during the auction, but cannot be increased.

- *Purchase Auction*: In this auction type the transaction can be concluded when a certain resource purchase price is met, i.e., the auction is automatically closed (`close`). So, the winner is the bidder who bids a value that is equal or greater than the purchase price. The auction type represented by this class does not allow multiple items of the same resource to be auctioned. This is an optional class, as it provides a functionality that can or cannot be desired.

- *Dutch Auction*: This auction type is used for the auction of multiple items of the same resource. The source party establishes an initial price and the several destination parties bid and supply the desired quantity of the resource. Multiple bids of multiple bidders with distinct quantities and prices are allowed. The rules to define the winners are:

  – the winner is the bidder that bids the highest price. The winners will pay the value corresponding to the least bid value among the winners. This means that everybody will pay the same value;

  – if a tie occurs among the winners (the bid value is the same, but with different item quantities), the winner is defined by the bid order. The losers can buy the remaining items. The value to be paid by all the winners is the value corresponding to the least bid value among them.

  This is an optional class, because Multiple¸Auctions can also be used for auctioning multiple items.

- *Multiple Auction*: This auction type works similarly to the standard auction. However, it admits the trade of multiple items of the same resource. So, several destination parties may acquire a certain quantity of resource items. All bids are classified by price, followed by quantity, and at last by time, i.e., higher value bids overcome lower value bids. If bids with the same value occur, the winner is the bidder that bids for more items. If there are bids with the same value and quantity, the winner is the first to bid. This is an optional class, because some auction systems use `Dutch Auction` for trading multiple items.

Example

Figure 8 presents an example of the AUCTION THE RESOURCE pattern adopted by the online auction site Arremate.com, which uses `Multiple Auction` for both single and multiple product items. Arremate.com also uses two other auction types: `Reserve Price Auction` and `Winner Auction`, which is an instantiation of the `Purchase Auction`. eBay uses `Dutch Auction` to trade multiple product items and `Standard Auction` to trade only one item. It also provides `Reserve Price Auction` and `Purchase Auction`. iBazar has only `Standard Auction` and `Reserve Price Auction`.

Variants

To make the pattern language useful to different auction types, new classes representing the auction rules may be added as specializations of the `Auction` class. These new auction types may need new attributes and methods.

In some systems `Standard Auction` is replaced by `Multiple Auction`. In this case `Multiple Auction` is used both for auctions of a single item and for several items.

To protect themselves against possible misuse of the system by customers, some auction systems require an identification document, for example, idCard or driver's license. On the other hand, some auction systems require the customer credit card to register, while others do not require specific information about the user.

Some auction systems allow a party to cancel the auction, while in others it is not allowed. There are rules for cancelling an auction, as for example: it can be cancelled

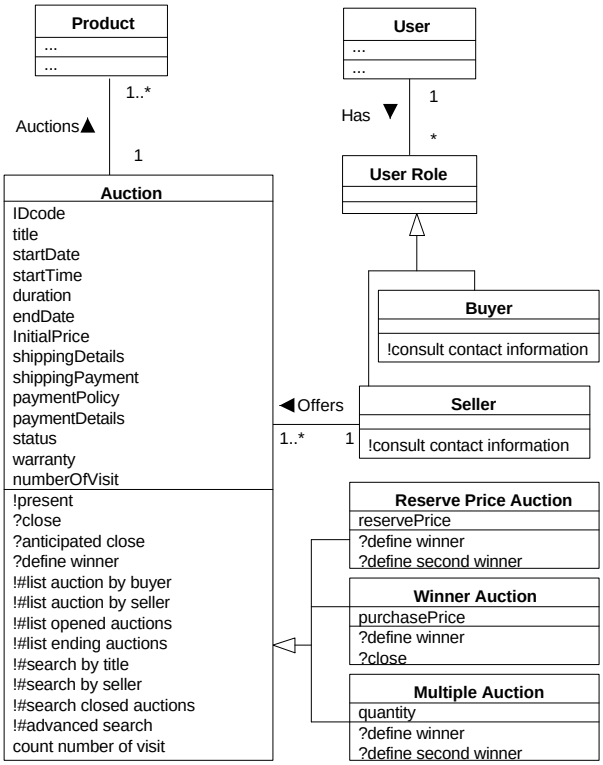

Figure 8. AUCTION THE RESOURCE **pattern example**

only during a pre-established period of time after its inclusion; it can be cancelled due to pre-established reasons; or simply because the source party wants to. Similarly, rules for changing auction data are necessary. Besides the examples mentioned in the cancellation case, some systems only allow the source party to add information, and not to remove or change information previously included. The parameters for cancelling and changing rules are stored in the `Auction House` class of the MANAGE THE AUCTION HOUSE pattern.

Related Patterns

This pattern is an application of patterns ASSOCIATION-OBJECT [1] and TIME ASSOCIATION [3]. It is also a combined application of patterns SPECIFIC ITEM-TRANSACTION and PARTICIPANT-TRANSACTION [4].

Following Patterns

After applying the AUCTION THE RESOURCE pattern try to use the PROMOTE NEGOTIATION pattern. If it is not applicable, then use the HANDLE BIDS pattern.

3.4 Promote Negotiation

Context

Your application deals with resources that have been identified, categorized, and are being traded by auction types already defined. It is often necessary for customers to clarify some doubts about these resources or about the auction in which they want to bid. Questions are answered directly by the source party. This type of communication among parties is similar to conventional trade and provides a direct channel among auction participating parties, causing a competitive environment that is beneficial to the auction.

Problem

How does your application support the negotiation among auction trading parties?

Forces

- Establishing rules for disclosing questions and corresponding answers is useful to clarify details about resources being auctioned. Hence, if both questions and answers are logged in the system, all users can benefit from the answers, not only the question authors.

- Using discussion lists for communication among parties helps to solve doubts, but there is not an explicit correspondence between questions and a specific auction. So, to retrieve questions/answers from a particular auction, it would be necessary to read several messages of the discussion list.

- Establishing rules to encourage the source party to answer questions is important to improve the chance of more bids. More details about the resource and the auction enhance trading chances. Besides that, this can show the real interest from the source party in trading the resource. Incentives can be given through fee discounts or other types of auction promotions.

Structure

Figure 9 shows the class diagram for the PROMOTE NEGOTIATION pattern.

Participants

- *Participant*: As described in the AUCTION THE RESOURCE pattern.

- *Auction*: As described in the AUCTION THE RESOURCE pattern.

- *Question*: Represents the questions asked by the `Participant` to a `Source Party`, about some `Auction`. These questions refer to auction details or doubts from the destination party.

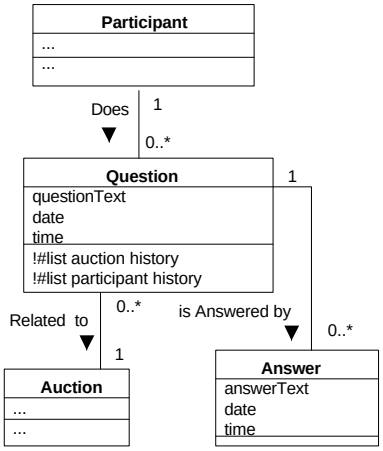

Figure 9. Structure of the PROMOTE NEGOTIATION **pattern**

- *Answer*: Represents the answers given by the source party to the questions asked by the destination parties.

Example

Figure 10 presents an example of the PROMOTE NEGOTIATION pattern used by the auction web site Arremate.com. Both eBay and iBazar use the same solution to offer a communication channel among trading parties.

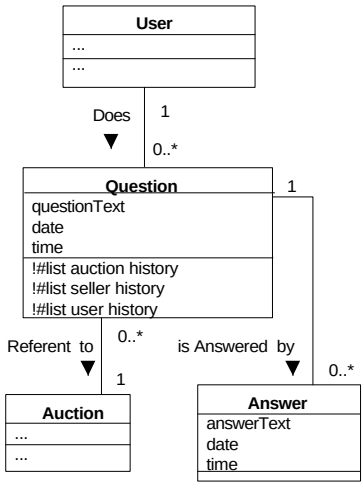

Figure 10. Example of the PROMOTE NEGOTIATION **pattern**

Following patterns

After applying the PROMOTE NEGOTIATION pattern, use the HANDLE BIDS pattern.

3.5 Handle Bids

Context

Your application deals with various types of resource auction that have already been established, providing an infrastructure for the trade. In an auction, resources are acquired through bids. These bids, together with some auction closing rules according to each auction type, will define the auction winner or winners. Bids follow rules that depend on the auction type, such as the quantity and increment value of the bids. Destination parties have their own list of preferred auctions. Auctions that belong to this list are more easily accessed and can be monitored more closely.

Problem

How does your application deal with the different types of bids related to the several types of auction?

Forces

- Storing bid information is important to define the auction winner(s). If the auction type allows multiple winners then data about the corresponding bids will be used for tie breaking.
- It must be ensured that the rules imposed by the auction type are applied to bids. Auctions with special bid types may exist.
- It is necessary to establish rules for bid cancellation, so that the auction participants are not harmed. These rules must cover bid cancellation both by the source party and by the destination party.

Structure

Figure 11 shows the class diagram for the HANDLE BIDS pattern.

Participants

- *Participant*: As described in the AUCTION THE RESOURCE pattern.
- *Participant Role*: As described in the AUCTION THE RESOURCE pattern.
- *Destination Party*: As described in the AUCTION THE RESOURCE pattern.
- *Auction*: As described in the AUCTION THE RESOURCE pattern.
- *Bid*: Represents the bid details obtained during the auction. This class has basic attributes, but new attributes may be added according to the auction system specific functionality. This can be done using, for example, the Properties pattern [5].

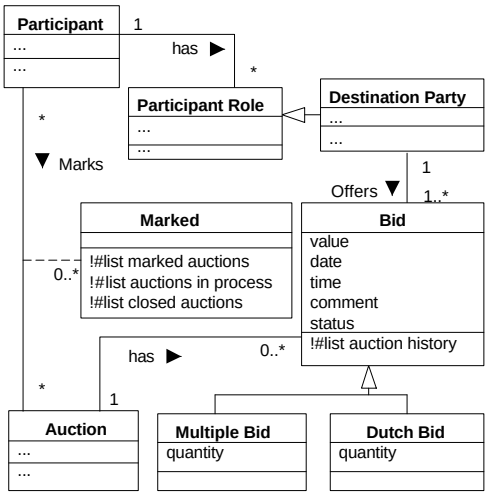

Figure 11. HANDLE BIDS **pattern**

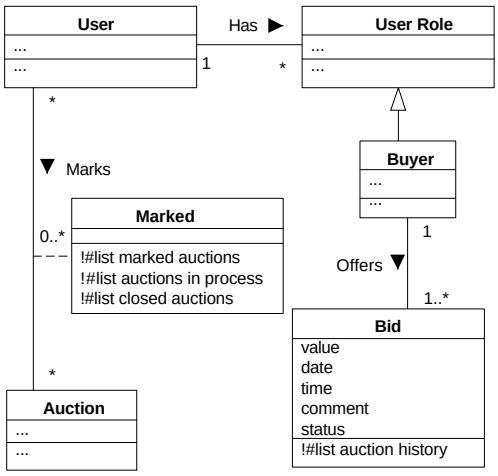

Figure 12. Example of the HANDLE BIDS **pattern**

- *Dutch Bid*: Represents bids for the `Dutch Auction`. So, this is an optional class that needs to be used only when `Dutch Auction` is used.
- *Multiple Bids*: Represents bids for the `Multiple Auction`. So, this is an optional class that needs to be used only when `Multiple Auction` is used.
- *Marked*: Represents the auctions marked by the destination parties so that they can better observe bids and, consequently, more rapidly access the auctions. So, auctions of more interest for the destination party can be found more easily.

Example

Figure 12 presents an example of the HANDLE BIDS pattern, used by the online auction site iBazar. Marking auction facilities are also offered by Arremate.com and eBay. iBazar does not allow bid cancellation. iBazar has several auction types but none of them accepts trade of multiple items, although they admit reserve value auctions. So, the `Bid` class, which stores data for normal bids, is enough to take care of auctions with only one resource item. Aremate.com works similarly to iBazar, except that it allows `Multiple Auction` (auction of several resource items). So, it is necessary to use the `Multiple Bids` class. Additionally, Arremate.com implements increment rules and offers an automatic bid service (see more details in the Variants section). eBay also implements increment rules and automatic bids, as well as bid cancellation and multiple resource items auctions (through `Dutch Auction`). eBay allows also that only selected destination parties are able to bid (details in the Variants sections).

Variants

As mentioned in the AUCTION THE RESOURCE pattern, several auction types can be added, together with the corresponding bid types to support them. If an auction type is added through specialization of the `Auction` class and it needs a specific bid type, this bid type must be added through specialization of the `Bid` class.

Besides the variants that resulted from the possible bid types, there are other variants of this pattern. Figure 13 shows the `Proxy Bidder` class, which represents the mechanism for automatic bidding. So, the application supplies a mechanism through which bids are automatically done according to some parameters established by the destination party to ease the trade. This mechanism is enabled by the destination party through the `enable proxy bidder` operation, which considers the `maximumBidAmount` attribute as an upper limit. When the destination party chooses the automatic bid option, the system is responsible for a new bid whenever another destination party bids a greater value, i.e., the mechanism immediately offers a bid that is superior to the previous bid without, however, exceeding the limit value established by the destination party.

Another variant, also presented in Figure 13, refers to setting up increment rules, provided by the `check bid increment` operation. This option is important when you want the bids to have a value that is proportional to the total resource value. This option affects the automatic bid mechanism, which has also to obey the bid increment rules. So, the automatic bid mechanism will bid a value that is immediately greater than the previous bid, is allowed by the increment rules, and does not exceed the limit value.

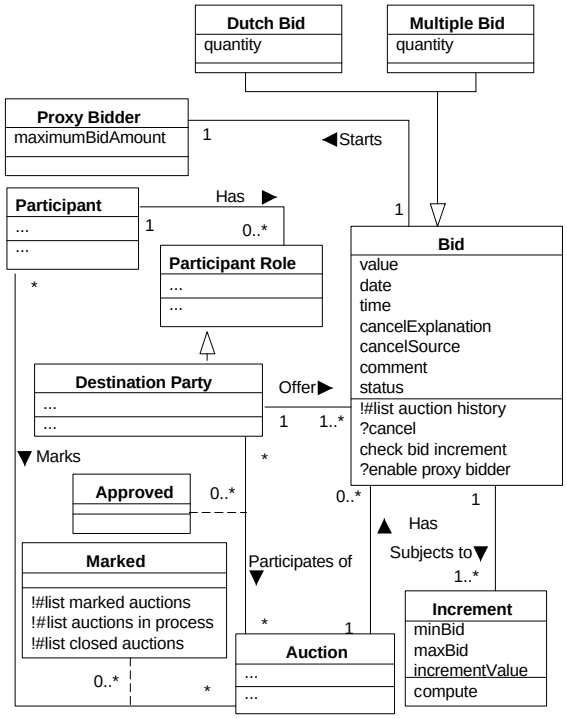

Figure 13. Variant of the HANDLE BIDS **pattern**

An auction may be accessed only by a number of destination parties previously determined by the source party. For example, this option may be chosen if a type of resource has specific types of buyers. The `Approved` class, shown in Figure 13, represents the destination parties able to participate on a certain auction. So, only previously established destination parties can bid in this auction.

In some auction systems it is possible to cancel bids, with different rules to do it. There are rules to control who can cancel a bid; if necessary, a limiting time for cancelling; if there are pre-established reasons to cancel; if it is necessary to explain the reasons; etc. New attributes and methods to deal with this functionality must be implemented in the classes `Bid` and `Auction House`.

Following Patterns

After applying the HANDLE BIDS pattern use the MANAGE THE AUCTION HOUSE pattern.

3.6 Manage the Auction House

Context

Your application manages resources to be auctioned by source parties; several destination parties will bid in order to acquire resources. This requires that the auction house establishes the rules for intermediating the trading process. It is responsible for charging or not fees due to system use and resource trade. The auction house determines the general parameters that guide the whole auction trade process.

Problem

How does your system manage the rules followed by the auction house involved in the auctioning process?

Forces

- Storing information about the auction house is important when trading is done through auction.
- Several generic rules about the auction house functioning need to be defined.
- It is necessary to establish how and which fees are to be paid by trading parties.
- Storing different fee types requires more space and processing time, though providing greater flexibility.

Structure

Figure 14 shows the class diagram for the MANAGE THE AUCTION HOUSE pattern.

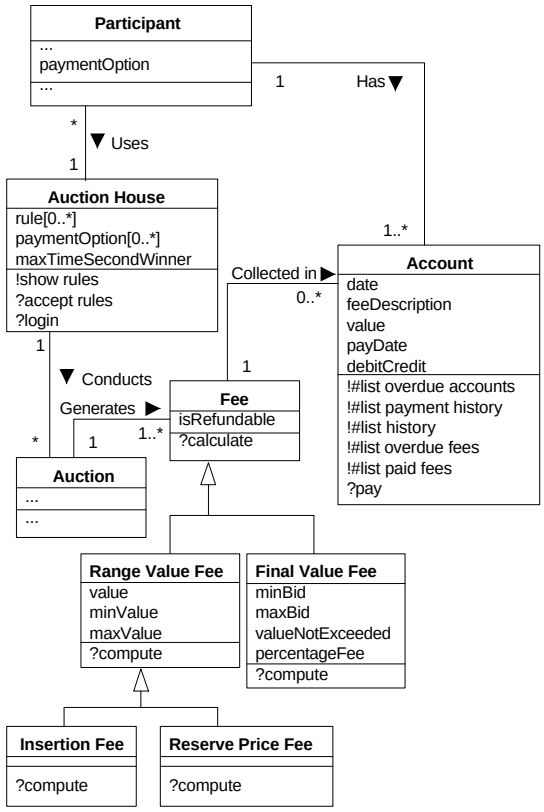

Figure 14. MANAGE THE AUCTION HOUSE **pattern**

Participants

- *Auction*: As described in the AUCTION THE RESOURCE pattern.
- *Auction House*: Represents the system itself, i.e., the auction house that intermediates the transaction. This class stores some parameters that refer to the global work of the auction process as, for example, the due date in which the source party can ask for contact data about the second winner (`maxTimeSecondWinner`). The `rule` attribute stores misbehavior general rules for source party, destination party or auction house. This attribute deals with legal and contractual issues of the service done by the auction house, which must be accepted (`show rules` and `accept rules`) by source and destination parties, so that they can use the auction system.
- *Fee*: This is an abstract class that represents the fees that can be charged by the auction house. The `refundable` attribute indicates whether this fee can be returned to the participant. This class, as well as all its specializations, is optional because fees can or cannot be charged, depending on the application.
- *Final Value Fee*: Represents a fee that is charged based on a percentage (`percentageFee`) of the final bid value, i.e., the bid that closed a successful auction.
- *Range Value Fee*: It is an abstract class representing a fee that is computed based on limit values (`minValue`, `maxValue`) and a fixed value (`value`).
- *Insertion Fee*: Represents a fee that is computed based on limit values of the initial resource price (`initialPrice`). This fee is charged when a resource is inserted in an auction and, usually, cannot be refunded.
- *Reserve Value Fee*: Represents a fee that is computed based on limit values of the resource reserve price (`reservePrice`). This fee is charged only for `Reserve Price Auction` and is usually refundable if the transaction is not effected.
- *Account*: Represents the charged fees and the possible refunds made to a participant. So, the value (`value`) can represent credit or debit for the participant (`debitCredit`). The use of this class is directly linked to the use of the `Fee` class, i.e., it can be applied only if the `Fee` class has also been applied.
- *Participant*: As described in the AUCTION THE RESOURCE pattern.

Example

Figure 15 presents an example of the MANAGE THE AUCTION HOUSE pattern used by the auction web site Arremate.com. Arremate.com has only the success fee, which is a fee computed based on the final value (`Success Fee`). The system implements a rule for the payment of fees in which the customer receives a warning with the value to be paid only when his/her account reaches a limit value. Besides the auction final value fee (`Final Value Fee`), eBay implements other fee types like reserve value fee (`Reserve Price Fee`) and insertion fee `Insertion Fee`. When the customer delays the fee payment over a pre-established period, system utilization is blocked. iBazar does not charge any type of fee.

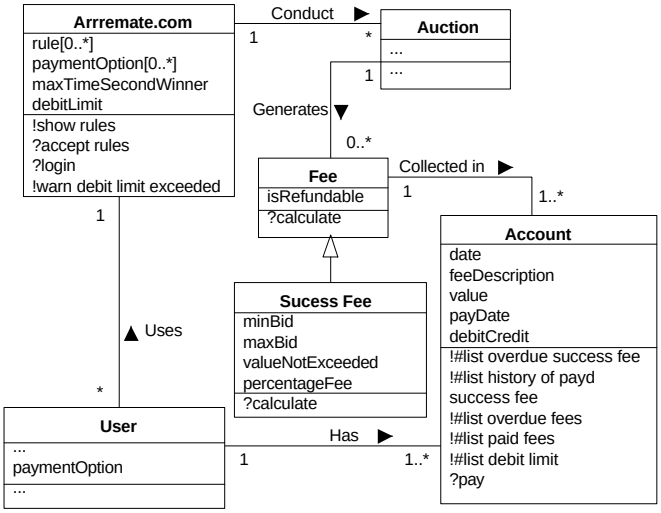

Figure 15. MANAGE THE AUCTION HOUSE **pattern example**

Variants

Some auctions systems have access to organizations that make the resource delivery, payment receipt and payment delivery. Information about these organizations are stored in the `Auction House` class.

Some systems offer fee discounts or free fees, due to marketing strategies established by the organization that owns the system. One example is the free insertion fee if the source party puts the resource on sale again for a variable number of times, during a pre-established period of time, before the resource is sold. Some additional parameters are stored in the `Auction House` class like period of time, number of times, and whether the fee is free or has a discount.

Related Patterns

This pattern applies the PARTICIPANT-TRANSACTION pattern [4].

Following Patterns

After applying the MANAGE THE AUCTION HOUSE pattern use the MANAGE REPUTATIONS pattern.

3.7 Manage Reputation

Context

Your application deals with resource auctions. Resources are offered by a source party and disputed by several destination parties through bids. The transaction occurs when a winner is declared and the auction is closed. After closing the auction, both source and destination parties can give their vote about the transaction conclusion with the other party. These reputation evaluations are necessary to protect the trading parties. So, reputations of all the trading parties are available in the system to be consulted at any time. This is a way to know which are the trading parties with good reputation and, thus, more trustable.

Problem

How can your application provide subsidies for the parties to evaluate each other trustability?

Forces

- Only the parties that have participated in a transaction can give their opinion about the other party and, so, supply reliable information. There are auctions in which multiple destination parties are winners and, so, specific rules must be implemented in this case.
- Keeping secret confidential data is important both to the organization that owns the system and to its customers.
- Providing a defense for both parties and storing the arguments of all the parties involved is essential to protect customers [16].

Structure

Figure 16 shows the class diagram for the MANAGE REPUTATION pattern.

Participants

- *Source Party*: As described in the AUCTION THE RESOURCE pattern.
- *Auction*: As described in the AUCTION THE RESOURCE pattern.
- *Destination Party*: As described in the AUCTION THE RESOURCE pattern.

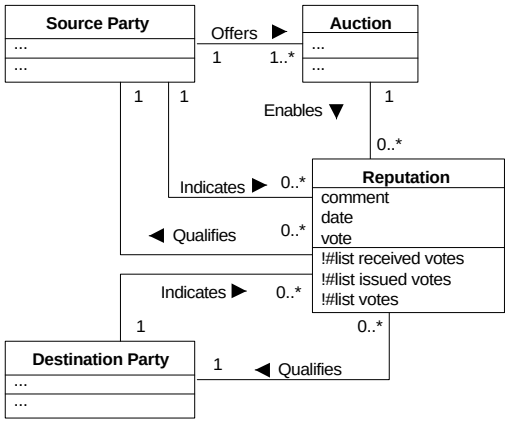

Figure 16. MANAGE REPUTATION **pattern**

- *Reputation*: Represents the vote given by a source party to a destination party or vice versa, evaluating it after the auction closing.

Example

Figure 17 presents an example of the MANAGE REPUTATION pattern, used by the web auction system iBazar. iBazar, eBay and Arremate.com allow the trading parties to generate evaluations. These votes can be viewed by other customers. Only eBay implements an option for making the list public, so that all customers can see any reputation or keep their votes secret to be seen only by who has given them.

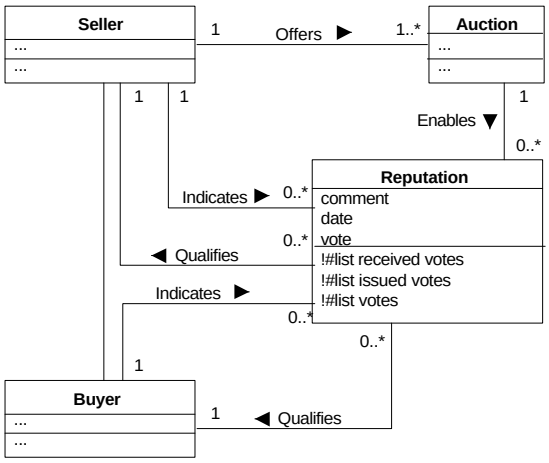

Figure 17. MANAGE REPUTATION **pattern example**

Variants

A variation of the evaluation functioning is to make it public or not, so that any person can see it or only who has given it. This option allows the customer to have the flexibility of disclosing his/her opinion about other traders or leave it secret. Most systems encourage public evaluation, to create a safer trading environment for customers that use online auctions.

In some auction systems the evaluation can have its value changed according to some rules that vary from system to system, as for example: the change can be done only once, or in a pre-established period of time, or at any time. The parameters for these rules, such as how many times changes can be done and the period in which they can be done, must be stored in the `Auction House` class of the MANAGE THE AUCTION HOUSE pattern. Other parameters, as for example whether the vote was already changed, must be stored in the `Reputation` class.

Following Patterns

If, during the application of the MANAGE THE AUCTION HOUSE pattern, the `Fee` class or any of its specializations has been used, then apply the HANDLE REFUNDING pattern. Otherwise try to use the ENABLE MESSAGES pattern. If it is not applicable, use the HANDLE ADVERTISEMENT pattern.

3.8 Handle Refunding

Context

Your application deals with resource auctions, in which fees are charged by the auction house. A number of these fees can be refunded in some cases, for example when a transaction is not effected due to the destination party giving up. In these cases a way to return the fees already paid by the destination party must be offered. On the other hand, the accused party has the right of defense for the refund request. So, the auction house can refund the eventual fees or not. Her decision is based on the refund request and the corresponding appeal request. After the case is judged, the auction house indicates whether the refund request proceeds, returning the fees to the source party and giving a negative evaluation vote to the destination party. Otherwise, the fees are not refunded and the source party receives a negative evaluation vote.

Problem

How can your application provide ways to refund fees that were unduely charged?

Forces

- Storing data about refund requests from one of the parties and the corresponding appeal request from the other party is important for the auction house decision making.
- It is necessary to refund fees according to each auction type and business policy.
- Allowing a reputation evaluation of the parties is essential to protect competent auctioneers.

Structure

Figure 18 shows the class diagram for the HANDLE REFUNDING pattern.

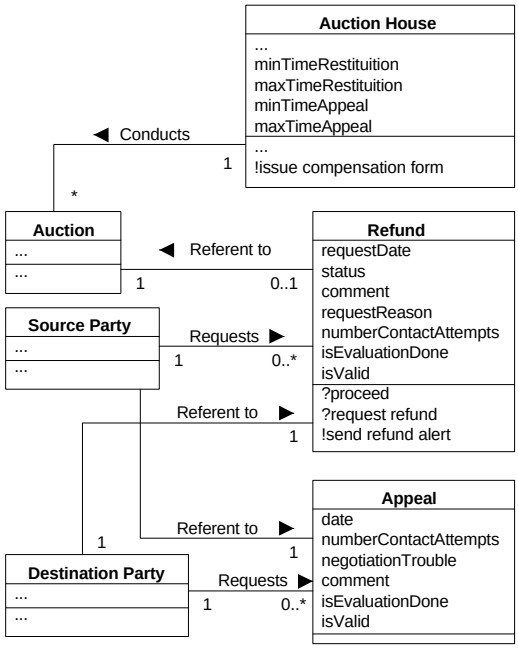

Figure 18. HANDLE REFUNDING **pattern**

Participants

- *Auction*: As described in the AUCTION THE RESOURCE pattern.
- *Source Party*: As described in the AUCTION THE RESOURCE pattern.
- *Destination Party*: As described in the AUCTION THE RESOURCE pattern.
- *Refund*: Represents the refund request, made by a source party, for the fees concerning a resource auction whose transaction has not been effected. This request can be done when, after closing an auction and defining its winner, the trade is not concluded. The source party can, then, ask for the fees refunding. The `isEvaluationDone` attribute indicates whether the source party has given an evaluation vote about the destination party reputation.

- *Appeal*: Represents the appeal requested from a destination party, referent to a refund request done by a source party. When a refund request occurs, the accused party has the right of defense, supplying information about the reasons for having not effected the trade. The `isEvaluationDone` attribute indicates whether the destination party has given an evaluation vote about the source party reputation.

- *Auction House*: As described in the MANAGE THE AUCTION HOUSE pattern.

Example

Figure 19 presents an example of the HANDLE REFUNDING pattern, used by the auction online site Arremate.com. When seller and buyer cannot settle an agreement, the seller has a minimum and maximum due date to ask for reimbursement of fees already paid for a trade that was not concluded. The system issues a warning to the buyer about the seller complaint, and the buyer has a due date to answer, filling in an appeal form. The auction house judges the case and decides whether the reimbursement is fair. In the positive case, the buyer receives a negative vote from the auction house. Otherwise, the seller receives a negative vote to his reputation. eBay uses almost the same procedure. It accepts the refund request, waits for the destination party to answer, and a possible understanding between parties. Otherwise a judgement is done and a decision is issued, similarly to Arremate.com procedure. iBazar, which is a free service, does not need to handle refunding.

Related Patterns

This pattern applies patterns ASSOCIATION-OBJECT [1] or TIME ASSOCIATION [3], TRANSACTION-SUBSEQUENT TRANSACTION and PARTICIPANT-TRANSACTION [4].

Following Patterns

After using the HANDLE REFUNDING pattern, try to use the ENABLE MESSAGES pattern. In case it is not applicable, use the HANDLE ADVERTISEMENT pattern.

3.9 Enable Messages

Context

Your application deals with resource trade through auctions. In this type of trade several events occur as, for example, the auction starting, the bids, the auction closing and the winner definition. Many of these events must be reported to customers using messages, so that they can monitor all the auction phases. Besides the messages that are directly related to auctions, messages with other purposes like advertisement, changes in the auctions, and promotions, can exist.

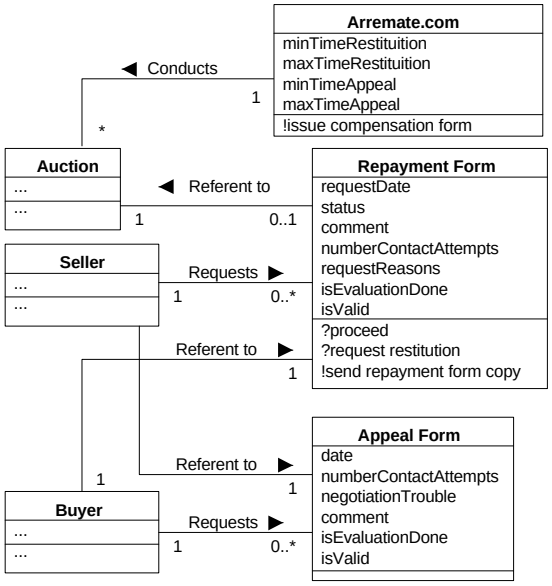

Figure 19. HANDLE REFUNDING **pattern example**

Problem

How does your application manage the messages sent to customers?

Forces

- Storing messages sent to customers is important for providing them the auction events history. This history can be an important source of information to support decision making regarding the auctions in which the customer is participating.

- It is desirable to let customers set the rules for messages receiving, so that they can determine which messages are considered important.

- Messages become a direct channel with customers and are useful for the auction house both as a simple communication channel and as a way to propagate merchandising. So, messages can be a marketing strategy tool.

Structure

Figure 20 shows the class diagram for the ENABLE MESSAGES pattern.

Participants

- *Participant*: Represents the party - organization or person - that intends to auction a resource (source) and the parties interested in acquiring a resource (destination). This class stores the options selected

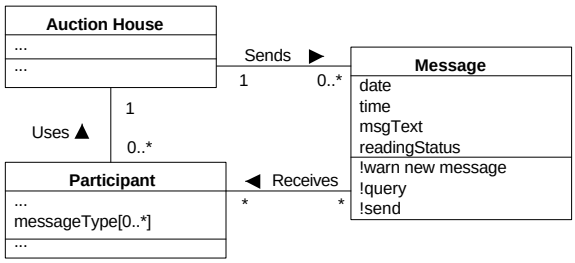

Figure 20. ENABLE MESSAGES **pattern**

by the participants regarding which messages they want to receive (`messageType`). These options refer to messages about events occurred in the auction, for example, the auction starting or closing, the auction winner, the bids sent, the greatest bid sent by other participant, etc.

- *Auction House*: As described in the MANAGE THE AUCTION HOUSE pattern.
- *Message*: Represents the messages stored and that are sent to the participant. These messages are available to be consulted at any time, both through the system or by email.

Example

Figure 21 presents an example of the ENABLE MESSAGES pattern, used by the online auction site iBazar. It does not offer the option for configuring the message types, i.e., customers cannot define which messages they want to receive or not. All messages are stored and can be removed only when the customer wants. The customer is warned about new messages whenever he or she executes the system login procedure. The option for configuring the message type is available both in eBay and Arremate.com. Arremate.com does not store the customer message history, as messages are only sent by email.

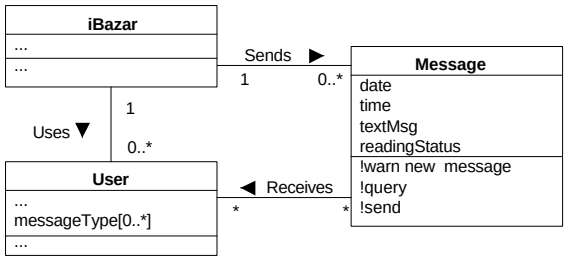

Figure 21. Example of the ENABLE MESSAGES **pattern**

Related Patterns

This pattern is an application of patterns ASSOCIATION-OBJECT [1] and TIME ASSOCIATION [3].

Following Patterns

After applying the ENABLE MESSAGES pattern, use the HANDLE ADVERTISEMENT pattern.

3.10 Handle Advertising

Context

Your application deals with resource auctions. The auction house offers some ways for the resources to be announced. The purpose is to promote and announce the auction to attract destination parties that are possibly interested in acquiring the resource. The destination party defines, among the options offered by the auction house, the advertisement type that better suits the resource to be sold. So, the auctions are announced using different emphasis and presentation features, depending on the advertisement type chosen.

Problem

How does your application manage auction advertising?

Forces

- Establishing advertisement rules that categorize the several auction types is important to promote the resource trade and the auction house. The more attractive are the auction announcements, the greater is the chance to attract new customers and, consequently, increase the number of trades.
- It is important to have an advertisement policy for auctions that have greater impact on customers, establishing an advertisement strategy that promotes other auctions.

Structure

Figure 22 shows the class diagram for the HANDLE ADVERTISING pattern.

Participants

- *Auction House*: As described in the MANAGE THE AUCTION HOUSE pattern.
- *Auction*: As described in the AUCTION THE RESOURCE pattern.
- *Advertisement*: Represents the way auctions are announced. This class stores the advertisement rules, as well as some data to effectively propagate the advertisements. Examples of rules are: the auction must have all its questions answered, the customer cannot have a negative evaluation, the auction must have a special type of promotion, the auction must be of a certain type (reserve price, Dutch, etc) or simply be

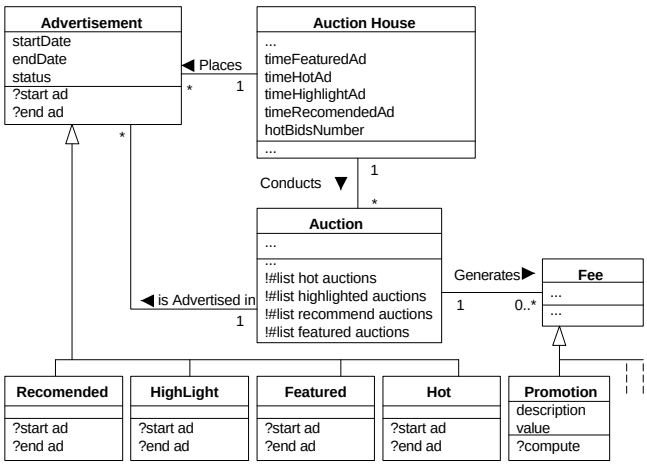

Figure 22. HANDLE ADVERTISING **pattern**

chosen because it is a resource that calls customers attention. These rules may be implemented for all specializations of this class.

- *Recommended*: Represents the auctions chosen by the system, among which are candidates established by business rules of the auction house, to be announced in the area of greatest importance of the site pages.
- *Highlight*: Represents the auctions chosen by the system to be announced in a highlighted area of the site pages.
- *Featured*: Represents the auctions that have a high visibility level in the site pages. They can be announced in the main page, in the search pages, or in special areas that ease the browsing among categories.
- *Hot*: Represents the auctions that have a good number of bids, so that they call the attention of the destination parties. The number of bids necessary for an auction to be classified as Hot is represented by the `hotBidsNumber` of the `Auction House` class. The Hot auctions may be placed in a highlighted area of the site main page.
- *Promotion*: Represents several simple forms for an auction to be presented (with boldface, with a picture beside the listing, in a highlighted position within a listing, etc). These forms are directly related to fees. So, this class represents not only the simplified types of advertisement, but also the service fees corresponding to these types. These promotion types can be used as a strategy for more elaborated advertisements (`Featured`, `Highlight`, `Recommended`). For example, an auction can only be announced as `Featured` if it has a certain promotion type.
- *Fee*: As described in the MANAGE THE AUCTION HOUSE pattern.

Example

Figure 23 presents an example of the HANDLE ADVERTISING pattern, used by the online auction site eBay. eBay has two special types of advertisement, Featured and Hot. The Featured type requires the auction to have a special promotion type ("Featured in the main page"). The auction house determines the auction advertisement period of time and the order in which auctions are announced. Arremate.com has three advertisement types and implements its own advertisement rules. However, it works similarly to eBay. iBazar does not distinguish among auction advertisements, using the `Advertisement` class to propagate the announcements for all the auctions. The advertisements work similarly to eBay and Arremate.com.

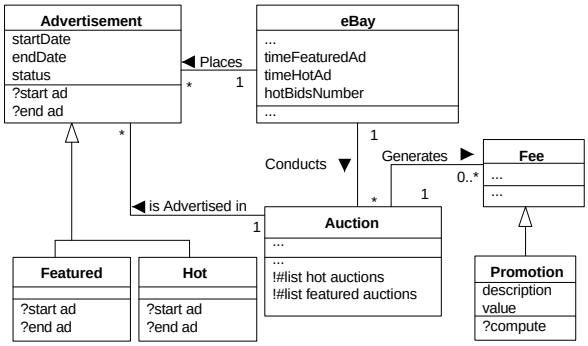

Figure 23. HANDLE ADVERTISING **pattern example**

Related Patterns

This pattern is an application of the patterns ASSOCIATION-OBJECT [1] and TIME ASSOCIATION [3].

Following Patterns

This is the end of your system analysis.

4 Example of the Pattern Language Application

Figure 24 shows the class model resulting from an application of the Pattern Language for Online Auctions Management. The tags show the role played by the class it points to. The tags contain labels like "P#n: role", where "n" is the pattern number and "role" is the role played by the class in that pattern [17]. It was applied to fulfill the requirements that refer to the auction process of the online

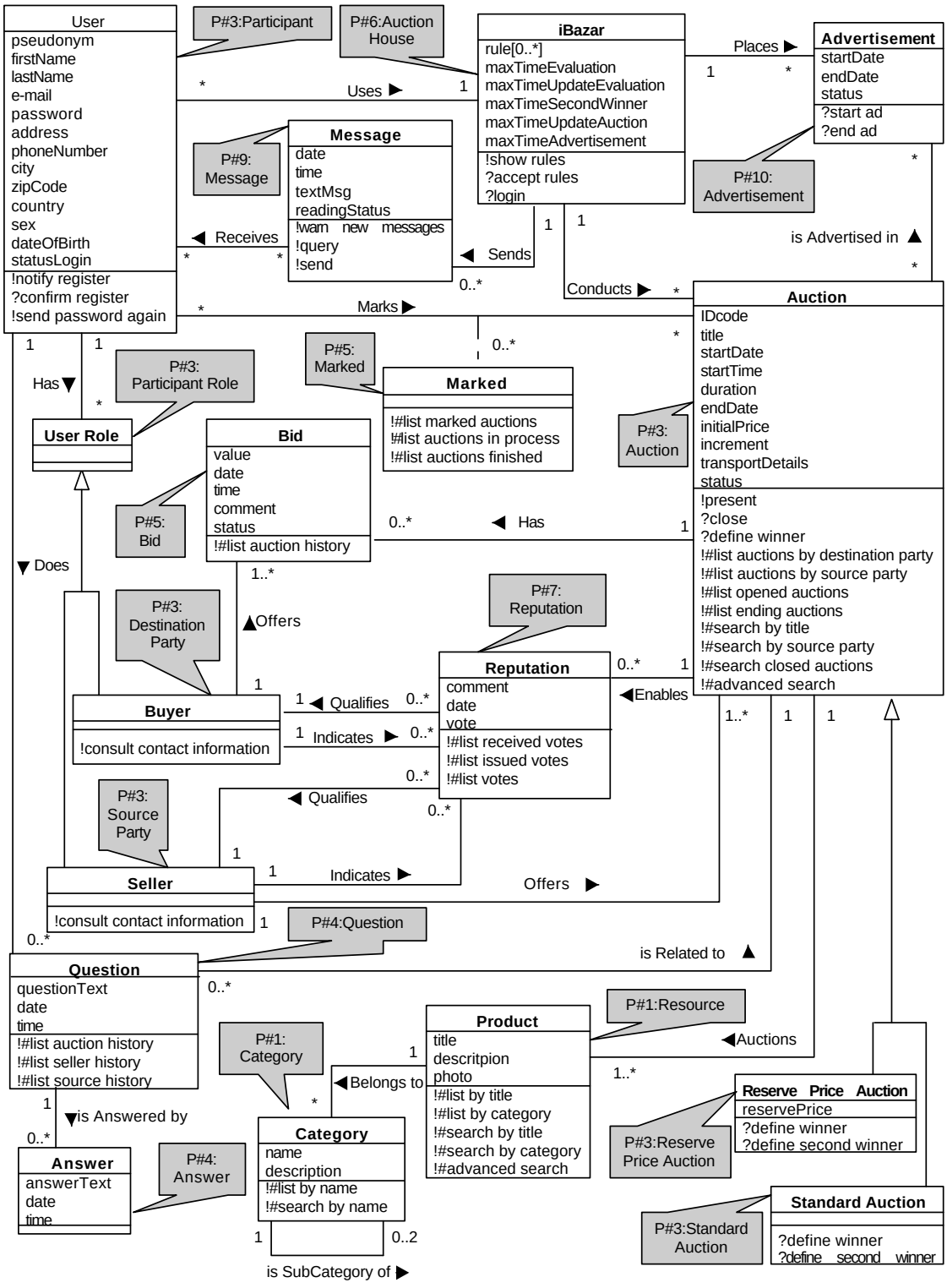

Figure 24. Example of the Pattern Language for Online Auctions Management Application: iBazar

auction site iBazar [15]. To attend the iBazar functionality, as captured from the reverse engineering of its interface, patterns (1) IDENTIFY THE RESOURCE, (3) AUCTION THE RESOURCE, (4) PROMOTE NEGOTIATION, (5) HANDLE BIDS, (6) MANAGE THE AUCTION HOUSE, (7) MANAGE REPUTATION, (9) ENABLE MESSAGES and

(10) HANDLE ADVERTISEMENT were applied. iBazar does not have the concept of resource categories preferred by the customer and, therefore, the ENABLE FAVORITE pattern was not applied. The HANDLE REFUNDING pattern was also not applied, because iBazar is free and, so, does not need to refund fees. It is interesting to notice that iBazar does not offer multiple resource items auction, limiting the auction to some resource types.

5 Concluding Remarks

The proposed pattern language addresses system development for several types of online auction systems to help software engineers during the analysis of such systems. A framework is being developed to support this pattern language, using Smalltalk. It will provide the classes for each pattern and a Web interface that can be customized to particular online auction applications. Future work includes extending this pattern language to cover other types of auctions, as for example reverse auctions.

6 Acknowledgements

We are grateful to our shepherd Federico Balaguer for his valuable comments that contributed to the present version of our pattern language. We thank FAPESP for the financial support to this research.

References

[1] Boyd, L. (1998). *Business Patterns of Association Objects*, pages 395–408. Pattern Languages of Program Design 3, Martin, R.C., Riehle, D. , Buschmann, F. – Addison-Wesley, Reading – MA, USA.

[2] Braga, R. T. V., Germano, F. S. R., and Masiero, P. C. (1999). A pattern language for business resource management. In *6th Pattern Languages of Programs Conference (PLoP'99)*, Monticello – IL, USA.

[3] Coad, P. (1992). Object-oriented patterns. *Communications of the ACM*, 35(9):152–159.

[4] Coad, P., North, D., and Mayfield, M. (1997). *Object Models: Strategies, Patterns and Applications*. Prentice-Hall, Upper Saddle River – NJ, USA, 2 edition.

[5] Foote, B. and Yoder, J. (1998). Metadata and active object-models. PLoP '98 and EuroPLoP '98 Conference, Technical Report #wucs-98-25, Dept. of Computer Science, Washington University Department of Computer Science, September 1998.

[6] Fowler, M. (1997a). *Analysis Patterns*. Addison-Wesley, Menlo Park – CA, USA.

[7] Fowler, M. (1997b). Dealing with roles. Proceedings of the 4 th Annual Conference on the Pattern Languages of Programs, Monticello, Illinois, USA, September 2-5, 1997. Available for download at: http://www.martinfowler.com/articles.html.

[8] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Massachusetts.

[9] Heck, E. and Vervest, P. (1998). How should CIOs deal with web-basead auctions? *Communications of the ACM*, 41(7):99–100.

[10] Huhns, M. N. and Vidal, J. M. (1999). Online auctions. *IEEE Internet Computing*, 3(3):103–105.

[11] Isakowitz, T., Bieber, M., and Vitali, F. (1998). Web information systems. *Communications of the ACM*, 41(7):78–80.

[12] Johnson, R. and Woolf, B. (1998). *Type-Object*, pages 47–65. Pattern Languages of Program Design 3, Martin, R.C., Riehle, D. , Buschmann, F. – Addison-Wesley, Reading – MA, USA.

[13] Kumar, M. and Feldman, S. T. (1998). Internet auctions. In *3rd USENIX Workshop on Electronic Commerce*, Boston – MA, USA.

[14] Larman, C. (1997). *Applying UML and Patterns: An Introduction to Object-Oriented Analisys and Design*. Prentice-Hall, Upper Saddle River – NJ, USA.

[15] Ré, R. and Masiero, P. C. (2001). Requirements specification for online auctions: Arremate.com, iBazar and eBay. ICMC/USP – Sao Carlos, SP, Brazil. Working Document (in Portuguese).

[16] Resnick, P., Zeckhauser, R., Friedman, E., and Kuwabara, K. (2000). Reputations systems. *Communications of the ACM*, 43(12):15–48.

[17] Vlissides, J. (1998). Pattern hatching. design patterns applied.

[18] Wellman, M. P. and Wurman, P. R. (1998). Real time issues auctions. In *First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, Denver, CO – USA.

[19] Yoder, J. and Barcalow, J. (1997). Architectural patterns for enabling application security. In Proceedings of the 4th Conference on Patterns Language of Programming (PLoP'97), 1997. 2.