Visual Specification of Layered Bidding Strategies for Autonomous Bidding Agents

Benjamin J. Ford, Haiping Xu, Christopher K. Bates
Computer and Information Science Department
University of Massachusetts Dartmouth, North Dartmouth, MA 02747, USA
Email: {u_bford, hxu, u_cbates}@umassd.edu

Sol M. Shatz
Computer Science Department
University of Illinois at Chicago, Chicago, IL 60607, USA
Email: shatz@uic.edu

Abstract—**In an agent-based online auction system, a bidding agent can automatically place bids on behalf of a human user according to a user-specified bidding strategy. Current implementations of bidding agents only support a set of simple predefined bidding strategies. In this paper, we introduce a formal bidding strategy model that supports specification of complex bidding strategies for autonomous bidding agents. The formal model is defined as a layered bidding strategy model (LBSM), which can be represented using notations adapted from UML activity diagrams. For real-time and efficient reasoning, the formal model is converted into a rule-based bidding strategy model (RBSM) represented in bidding strategy language (BSL), which can be directly executed by a reasoning module of an autonomous bidding agent. We present an algorithm for converting an LBSM to a rule-based bidding strategy model, and an algorithm to drive the reasoning engine. Finally, we develop a prototype agent-based online auction system using JADE, and demonstrate how layered bidding strategies can be precisely specified, and how our approach may support analysis of impacts on bidding histories by using different bidding strategies in agent-based online auctions.**

Index Terms—**online auction; software agent; bidding strategy; UML diagram; rule-based model; shill bidder**

I. INTRODUCTION

Online auction houses, such as eBay, have seen an increasing amount of transactions since their debut. As the number of transactions increases, researchers have been investigating the mechanisms and benefits of automating online auction activities. One major form of such automation is agent-based online auctions, which are Internet auctions running partially or entirely through the use of software agents, where software agents can act on behalf of human users, such as buyers, sellers, and auction house administrators [1-3].

In an agent-based online auction system, a bidding agent can automatically place bids on behalf of a human user according to a user-specified bidding strategy [4-6]. A bidding strategy consists of a set of bidding activities and conditions. During an online auction, when certain conditions become true, appropriate bidding activities (e.g., increasing the bid amount or placing a bid) can be automatically performed by the bidding agent. While there have been previous efforts on designing optimal bidding strategies [7-9], work on specifying bidding strategies for bidding agents is more rare. Current implementations of bidding agents only support a set of simple predefined bidding strategies [10-12]. One other strategy specification framework utilizes a logic-based approach [13]; however, that approach lacks the flexibility necessary for specifying large and complex strategies. In order to support user-specified bidding strategies for autonomous bidding agents, there is a pressing need for a feasible way for allowing users to specify bidding strategies that effectively represent the user's bidding plans.

In this paper, we introduce a model-based approach that supports specification of complex and layered bidding strategies for autonomous bidding agents (we use the terminologies of *bidding agent* and *autonomous bidding agent* interchangeably in the rest of this paper). Our approach divides a complex strategy into various modular layers. Simple strategies at lower layers can be assimilated into a larger and more complex strategy at a higher layer. For real-time and efficient reasoning, the formal model is converted into a rule-based bidding strategy model represented in bidding strategy language (BSL). Thus the rule-based strategy model can be directly executed by a reasoning module of a bidding agent using a reasoning engine.

This work extends our previous research on specification of flexible and complex bidding strategies in agent-based online auctions [14]. In this paper, we further provide formal definitions of our layered bidding strategy model, present the interface of a visual strategy builder (VSB) that supports visual specification of layered bidding strategies for autonomous bidding agents, and analyze new experimental results generated using our approach. Since our approach adapts notations from UML activity diagrams [15-16] for representing bidding

Manuscript received August 14, 2009; revised October 9, 2009; accepted October 11, 2009.

Corresponding author: Haiping Xu, Email: hxu@umassd.edu

doi:10.4304/jcp.5.6.940-950

strategies, VSB provides users a familiar visual method for specifying flexible and complex bidding strategies. In addition, VSB supports real-time modification of bidding strategies by users. When a user modifies a bidding strategy for a bidding agent at runtime, the rule-based bidding strategy model can be dynamically updated. In this case, all subsequent bidding activities of the bidding agent will be based on the updated bidding strategy model. In addition, our approach is relevant to our current research on trustworthy agent-based online auctions [3], where auction frauds, especially shilling behaviors [17-21] can be automatically detected. Note that a shilling behavior is a type of auction fraud, where a shill bidder can easily disguise himself as a legitimate user in order to drive up the bidding price [22].

The rest of this paper is organized as follows. In Section II, we describe related work and highlight the relationships to our research. In Section III, we first present an overview of agent-based online auction systems, and then describe a bidding agent architecture that supports specification of layered bidding strategies. In Section IV, we give a detailed description of a layered bidding strategy model (LBSM) and illustrate its basic ideas using simple examples. To generalize our ideas, we provide some key formal definitions for our layered model. In Section V, we discuss about a rule-based bidding strategy model (RBSM), and present an algorithm for converting an LBSM to an RBSM, and an algorithm to drive the reasoning engine. In Section VI, we give a brief description of the visual strategy builder interface for specification of LBSM, and then provide a case study to show how our approach may support analysis of impacts on bidding histories by using different bidding strategies in agent-based online auctions. In Section VII, we provide conclusions and our future work.

II. Related Work

Previous related work includes research on designing good bidding strategies for agent-based online auctions, and work on formal specification of bidding strategies. Park, et al. develop an adaptive agent bidding strategy, called the *p*-strategy, based on stochastic modeling for dynamic, evolving multi-agent auctions [7]. The *p*-strategy considers the dynamics and resulting uncertainties of an auction process using stochastic modeling, which can adaptively decide when the model should be used. Ma and Leung present the design and analysis of a new strategy for buyer and seller agents participating in agent-based continuous double auctions (CDA) [8]. The proposed strategy employs heuristic rules and reasoning mechanisms based on a two-level adaptive bid-determination method, which allows bidding agents to dynamically adjust their behaviors in response to changes in the supply-demand relation of the market. Although the above proposed bidding strategies may provide chances for a user to win auctions, they are either difficult to use by inexperienced and ordinary users, or they must be predefined as bidding strategies for bidding agents. In the latter case, users are typically not allowed to modify or improve the bidding strategy to meet their personal preferences and needs. In contrast, our approach explicitly provides users the mechanisms to adopt an existing bidding strategy, design their own strategies, and compose available strategies into a more complex one. With such mechanisms, a bidding agent can truly place bids on behalf of a human user to meet the user's bidding requirements.

Very little work has been done on formal specification of bidding strategies. Gimenez-Funes, et al. introduce both a formal and pragmatic approach for the design of bidding strategies with useful heuristic guidelines for buyer agents [23]. The proposed approach utilizes global and individual probabilistic information such that the resulting bidding strategy can balance the agent's short-term and long-term benefits. Other research has described how defeasible logic can be utilized to specify negotiation strategies [24, 13]. Defeasible logic – although it is formal and allows users to specify rules based on uncertainty – has an inherently large learning curve due to its mathematical foundations [25]. Efforts have been made to counter this disadvantage by utilizing digraphs [25], but that representation still requires users to learn a notation that is not widely used.

Additional work that is related to our proposed approach includes specification of bidding strategies with heuristics and fuzzy logic. Anthony and Jennings propose a heuristic decision making framework for autonomous agents to bid across multiple auctions with varying protocols [26]. The framework allows an agent to adopt varying tactics and strategies that is consistent with the user's preferences. He, et al. present a novel heuristic bidding algorithm for software agents to obtain multiple goods from overlapping auctions [27]. The algorithm uses neurofuzzy techniques to predict the expected closing prices of auctions and to adapt the agent's bidding strategy to reflect the type of environment in which it is situated.

Unlike the above approaches, our formal bidding model adopts some notations from UML activity diagrams – a popular standardized notation – to explicitly display strategy transitions and action transitions as a workflow of activities. Such a representation can support an easy-to-use interface for users to graphically specify bidding strategies. As a result, it is expected that with our approach, it will be significantly easier for users to learn how to specify strategies, while still allowing them to specify complex and flexible bidding strategies. For example, one such bidding strategy may be based on adapting to other bidding agents' bidding behaviors, possibly by examining increases in bidding increment or bidding frequency for a given period of time.

III. Bidding agent architecture

Figure 1 presents an overview of a general agent-based online auction system. There is a central auction house that consists of various auction agents, each of which manages an auction in progress. The bidding agents, which represent human users, can search for auction agents through a Directory Facilitator (DF) agent. A user who wants to bid automatically on a particular auction must provide its bidding agent with a bidding strategy. The bidding agent then communicates with the corresponding auction agent to query for related information, such as the current highest bid and the number of active bidders in that auction. Based on the available information, the bidding agent makes decisions, and may place bids by sending bid requests to the corresponding auction agent in the auction house.

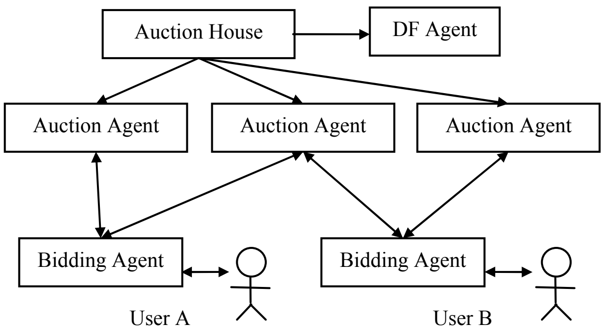

Figure 1. Agent-based online auction system

A bidding agent consists of a bidding agent interface and a reasoning module. The bidding agent interface is responsible for communicating with the DF agent, auction agents and human users. The reasoning module is used to make decisions for choosing the next bidding activity according to user-specified bidding strategies.

Figure 2 describes the bidding agent architecture. A user can specify a layered bidding strategy model (LBSM) through the bidding agent interface. The LBSM is represented as a visual strategy model that is internally stored as an XML file. Once a strategy has been defined, it is converted into a rule-based bidding strategy model (RBSM) consisting of a set of production rules. The production rules can be directly executed by the reasoning module for decision making. Based on the current state of the auction, the reasoning module determines the next bidding action and sends it to the bidding agent interface for further processing. For example, if the next action is to place a bid, the bidding agent makes a bid request to the corresponding auction agent through the bidding agent interface.

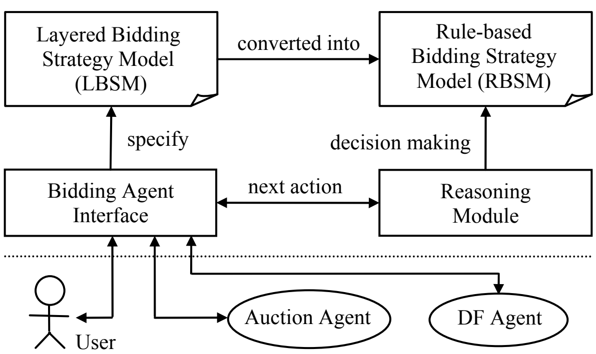

Figure 2. Bidding agent architecture

IV. LAYERED BIDDING STRATEGY MODEL

In our model-based approach for bidding agents, we utilize a layered architecture to specify bidding strategies. A layered architecture allows specification of bidding strategies at different levels of complexity. Figure 3 illustrates the general architecture of our layered bidding strategy model (LBSM).

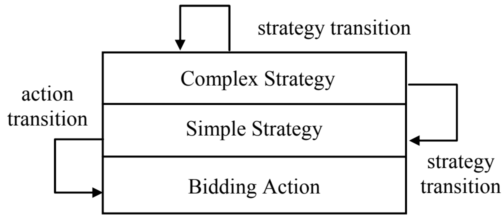

Figure 3. Layered bidding strategy model (LBSM)

An LBSM consists of three layers, namely the *complex-strategy* layer, the *simple-strategy* layer, and the *bidding-action* layer. The complex-strategy layer defines complex strategies using simple strategies from the simple-strategy layer as well as complex strategies from the same layer. The functionality of switching between strategies in a complex strategy is defined by *strategy transitions*. The simple-strategy layer defines simple strategies using bidding actions defined in the bidding-action layer. The functionality of switching from a bidding action in a simply strategy to another bidding action is defined by *action transitions*. The bidding actions layer defines the atomic bidding actions available to a bidding agent. Examples of such actions include placing a bid, changing bid limit, and random pausing for a specified range of time.

Figure 4 shows an example of simple strategy called *S*1 using notations of UML activity diagrams. The initial action is a *ChangeDynamicBidIncrement* action that must be executed first whenever strategy *S*1 is selected to execute. This initial action changes a user's bid increment to \$10. The next action is a *DynamicBidAction* that places a single bid in the amount of the current highest bid plus the current bid increment (10 dollars). The strategy then requires a pause for a random time between 8 minutes (480 seconds) and 16 minutes (960 seconds), followed by a check to see if the transition condition `!highBidder && (highBid + 10 <= bidLimit)` is true or not. This condition is used to check whether the bidder's last bid remains the highest or whether placing another bid may exceed a pre-specified bid limit for this user. In either case, if the transition condition, as specified, evaluates to *true*, the next action will again be *DynamicBidAction*; otherwise, a *PauseBiddingAction* will be taken. This procedure must be repeated until the bid limit is reached or the auction terminates. Since simple strategy *S*1 describes a typical behavior of a bidding agent, we call *S*1 a normal strategy. Figure 5 shows a part of the XML representation for strategy *S*1, which can be automatically converted into an RBSM (we will describe the conversion algorithm in Section V).

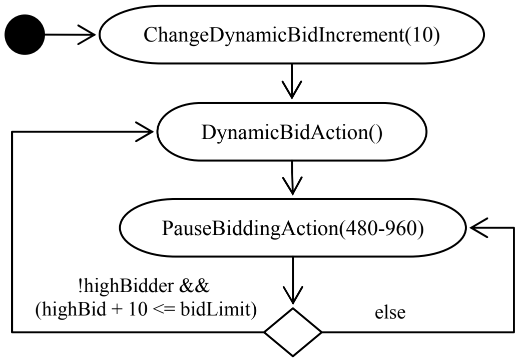

Figure 4. Simple strategy *S*1 (normal strategy)

```
<LBSM><Simple Strategy>
<Strategy id = "S1">
<Actions><Action id = "a1">
    <Initial>true</Initial>
    <Class>ChangeDynamicBidIncrement</Class>
    <Parameter>10</Parameter>
</Action>
<Action id = "a2">
    <Class>DynamicBidAction</Class>
    <Parameter>null</Parameter>
</Action>
<Action id = "a3">
    <Class>PauseBiddingAction</Class>
    <Parameter>480-960</Parameter>
</Action></Actions>
<Transitions><Transition>
    <Start>a1</Start><End>a2</End>
    <Condition>null</Condition>
</Transition>
  ...
</Transitions></Simple Strategy></LBSM>
```

Figure 5. XML representation of simple strategy *S*1

In Table I, we list a few key atomic bidding actions that can be used to specify simple bidding strategies such as strategy *S*1 defined in Figure 4. Note that most bidding actions require a parameter (e.g., *BasicBidAcion*), but some may not (e.g., *DynamicBidAction*).

TABLE I.
ATOMIC BIDDING ACTIONS FOR SIMPLE STRATEGY

Bidding Action	Parameter (type)	Semantic
BasicBidAction	bid amount (long)	Place a bid with a fixed bid amount.
ChangeDynamic BidIncrement	bid increment (long)	Change the bid increment for the following executed DynamicBidActions with no parameter.
DynamicBid Action	none	Place a bid with a dynamic bid amount, which equals to the current bidding price plus a bid increment specified by the latest ChangeDynamicBidAction.
DynamicBid Action	bid increment (long)	Placing a bid with a dynamic bid amount, which equals to the current bidding price plus the bid increment.
ChangeBidLimit Action	bid limit (long)	Change the user's previously specified bid limit.
PauseBidding Action	pause time (long)	Stop bidding for a random pause time (in seconds) with ±15% of the specified pause time.
PauseBidding Action	range of pause time (long-long)	Stop bidding for a random pause time (in seconds) with a range of the pause time specified.

Table II shows the keywords that can be used to specify transition conditions. For example, the keyword *highBid* represents a variable used in a condition, which is set to the current highest bidding price in the auction when that condition is evaluated. Similarly, variable *highBidFrequency* is used in a condition to evaluate the largest number of bids places by a bidder (other than the bidder who evaluates the condition) in the past 20 minutes. Note that the higher *highBidFrequency* is, the more aggressive the corresponding bidder is in the corresponding auction.

TABLE II.
KEYWORDS DEFINED FOR TRANSITION CONDITIONS

Keyword	Type	Semantic
bidIncrement	long	The current bid increment set by the ChangeDynamicBidAction.
bidLimit	long	The current bid limit set either manually by the user or by ChangeBidLimitAction.
highBid	long	The current highest bid for the auction.
bidDifference	long	The difference between the last two bids in the auction.
highBidFreqency	long	The largest number of bids made by a bidder in the past 20 minutes.
numberBidders	long	The number of bidders that have participated in the auction.
timeBetweenBids	long	The elapsed time between the last two bids in the auction (in seconds).
timeSinceLastBid	long	The elapsed time since the last bid in the auction (in seconds).
timeRemaining	long	The remaining time in the auction (in seconds).
timeElapsed	long	The elapsed time since the auction started (in seconds).
auctionDuration	long	The duration of the auction (in seconds).
highBidder	boolean	Set to true if the current agent is the one who placed the current highBid for the auction; otherwise, it is set to false.
else	boolean	Set to true when conditions on all other branches are false.

Figure 6 and 7 illustrate two additional simple bidding strategies. Strategy *S*2 defined in Figure 6 looks similar to strategy *S*1 (defined in Figure 4); however, in *S*2, after each dynamic bid action, there is a random pause of $60\pm 15\%$ seconds, and then the strategy checks if the elapsed time since the last bid in the auction exceeds 180 seconds (3 minutes). If this is true, and also if the bidding agent using this strategy is not a `highBidder` and placing a new bid will not exceed the bid limit, a dynamic bid is placed; otherwise, the strategy again pauses for $60\pm 15\%$ seconds. In other words, a bidding agent using *S*2 attempts to avoid competing with others by placing bids only when no other bidder has placed a bid in the past 3 minutes; we call such a strategy a *cautious* one. Note that we implement the pause time as a random time, so the bidding strategy adopted by a bidding agent will not be easily detected by other bidders.

Strategy *S*3 defined in Figure 7 also looks similar to *S*1, but its resulting bidding behavior is quite different from that of *S*1. In *S*3, after each dynamic bid action, it pauses for $240\pm 15\%$ seconds (or around 4 minutes) rather than a random time between 8 and 16 minutes as in

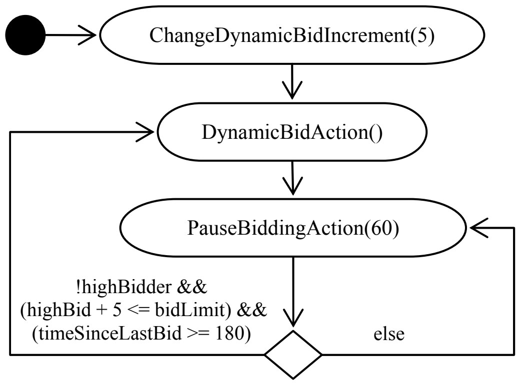

Figure 6. Simple strategy *S*2 (cautious strategy)

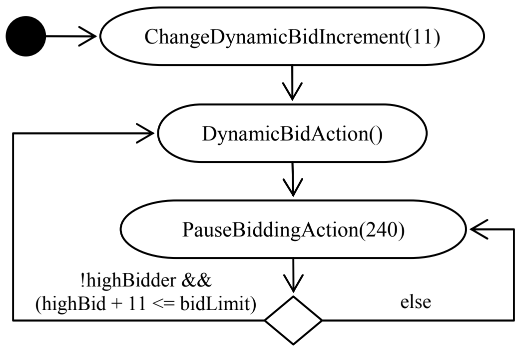

Figure 7. Simple strategy *S*3 (aggressive strategy)

*S*1 before it places another dynamic bid. Thus, *S*3 is more aggressive than *S*1, and we call it an *aggressive* strategy.

A complex strategy consists of a set of strategies, strategy transitions and an initial strategy. Note that a strategy in a complex strategy can either be a simple or complex one. The initial strategy is the one that is first executed when the complex strategy is selected to execute. Figure 8 shows a complex strategy *C*1 that contains two simple strategies *S*2 and *S*3, where *S*2 is the initial strategy of *C*1. When *C*1 is selected to execute, the initial action in *S*2 will be executed first. While the time remaining in the auction is greater than 720 seconds, *S*2 is executed continuously. Once the time remaining is less than or equal to 720 seconds, a strategy transition must be made to invoke simple strategy *S*3. Similarly, in this case, the initial action of *S*3 will be selected as the next action. Note that although we illustrate only two simple strategies in *C*1, a complex strategy may also contain other complex strategies as components. Figure 9 shows a part of the XML representation for complex strategy *C*1.

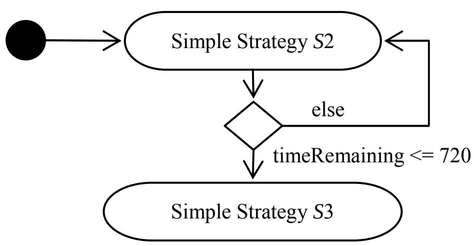

Figure 8. Complex strategy *C*1 (complex aggressive)

```
<LBSM><Complex Strategy>
<Strategy id = "C1">
<Strategies><Strategy id = "S2">
   <Initial>true</Initial>
   <Type>Simple</Type>
   <Name>Simple Strategy S2</Name>
</Strategy>
<Strategy id = "S3">
   <Type>Simple</Type>
   <Name>Simple Strategy S3</Name>
</Strategy></Strategies>
<Transitions><Transition>
   <Start>S2</Start><End>S3</End>
   <Condition>timeRemaining <= 720
   </Condition></Transition>
    ...
</Transitions></Complex Strategy></LBSM>
```

Figure 9. XML representation of complex strategy *C*1

To generalize our ideas, we now provide some key formal definitions for our layered bidding strategy model.

Definition 4.1 *Layered Bidding Strategy Library*

A layered bidding strategy library *LBSL* is defined as a 4-tuple (*SCS, SSS, SBA, STC*), where *SCS* is a set of complex strategies; *SSS* is a set of simple strategies; *SBA* is a set of atomic bidding actions; and *STC* is a set of transition conditions (see Definition 4.5). The *LBSL* defines a set of building blocks for users to specify a new layered bidding strategy.

Definition 4.2 *Complex Strategy*

A complex strategy *CS* is defined as a 3-tuple (*SBS*, *SST*, S_0), where *SBS* is a set of simple and/or complex bidding strategies; *SST* is a set of strategy transitions; and S_0 is the initial strategy of the complex strategy *CS*.

Definition 4.3 *Simple Strategy*

A simple strategy *SS* is defined as a 3-tuple (*SBA*, *SAT*, A_0), where *SBA* is a set of atomic bidding actions; *SAT* is a set of action transitions; and A_0 is the initial action of simple strategy *SS*.

Definition 4.4 *Bidding Action*

A bidding action *BA* is defined as an atomic action that is valid for bidding agents. For example, bidding actions *BasicBidAction* and *PauseBiddingAction* can be used by a bidding agent to place a bid during an auction and to pause for some random time, respectively.

Definition 4.5 *Transition Condition*

A transition condition *TC* is defined as a Boolean expression that evaluates to *true* or *false* based on auction states. For example, the condition `highBid > 500` becomes *true* when the current highest bid in an auction exceeds \$500. Such conditions are used to determine whether or not an action transition or strategy transition can be taken.

Definition 4.6 *Strategy Transition*

A strategy transition *ST* is defined as a 3-tuple (*STS, ENS, TC*), where *STS* is a start strategy; *ENS* is an end strategy; and *TC* is a transition condition.

Definition 4.7 *Action Transition*

An action transition *AT* is defined as a 3-tuple (*STA, ENA, TC*), where *STA* is a start action; *ENA* is an end action; and *TC* is a transition condition.

V. Rule-based Bidding Strategy Model

To facilitate efficient execution of a strategy by the reasoning module, we define a formal language called bidding strategy language (BSL) to specify rule-based bidding strategy models. As we mentioned previously, a rule-based bidding strategy model (RBSM) can be automatically converted from an LBSM, but it allows efficient reasoning, and may potentially support being expanded in real-time with new rules enforced by an auction house. Figure 10 gives the formal definitions of BSL in Backus-Naur Form (BNF).

```
<production rule>::= <strategy rule> | <action
rule> | <initial strategy rule> | <initial
action rule>
<strategy rule> ::= <s-domain> <bidding
  strategy> <condition> -> <bidding strategy>
<s-domain> ::= <s-domain>.<complex strategy> |
  <complex strategy>
<bidding strategy> ::= <simple strategy> |
  <complex strategy>
<condition> ::= <compound condition> |
  <arithmetic condition> | <comparison
  condition> | <boolean condition>
<action rule> ::= <a-domain> <action>
  <condition>  -> <action>
<a-domain> ::= <s-domain>.<simple strategy> |
  <simple strategy>
<action> ::= <basic bid> | <change bid limit> |
  <change dynamic bid increment> | <dynamic
  bid> | …| <pause> | <stop>
<initial strategy rule> ::= <complex strategy>
  -> <initial strategy>
<initial action rule> ::=  <simple strategy> ->
  <initial action>
```

Figure 10. Definition of BSL in Backus-Naur Form (BNF)

From the definitions, we can see that a rule-based bidding strategy model is specified using production rules, which include strategy rules, action rules, initial strategy rules and initial action rules. A strategy rule corresponds to a strategy transition with an *s-domain*, which specifies the strategy transition's enclosing strategies at different levels. For example, if complex strategy *C*1 contains simple strategies *S*1 and *S*2, and if *C*1 itself is defined as a component of complex strategy *C*2, a strategy rule that transits from *S*1 to *S*2 would have an *s-domain* of *C*2.*C*1. Similarly, an action rule corresponds to an action transition with an *a-domain*, which specifies the action transition's enclosing strategies at different levels. Note that an *a-domain* of an action rule follows the same principle as that of a strategy rule, but its first enclosing strategy must be a simple strategy rather than a complex one. An initial strategy rule is a special rule that defines the first strategy to be used in a complex strategy; while an initial action rule defines the first action to be taken in a simple strategy. Thus, a strategy defined in BSL is essentially a set of production rules, which defines action rules and initial action rules for simple strategies, and strategy rules and initial strategy rules for complex strategies.

The model conversion algorithm (Algorithm 1) converts a user-specified LBSM to a set of rules that can be executed directly by the reasoning module. The algorithm first checks whether an LBSM describes a complex strategy. If so, it creates an initial strategy rule and a list of strategy rules based on the LBSM. Once the list of strategy rules has been created, the algorithm starts to process each strategy contained in the LBSM recursively. On the other hand, if the LBSM is a simple strategy (i.e., the base case), the algorithm creates an initial action rule and a list of action rules.

Algorithm 1. Model Conversion

```
function convertToRuleBasedStrategyModel (LBSM lbsm)
  if lbsm is a complex strategy
    add a new initial strategy rule:
          lbsm → lbsm.initialStrategy
    for each StrategyTransition st in fbsm
      set up s-domain according to the strategy hierarchy
      add a new strategy rule: s-domain, st.startStrategy,
            st.condition → st.endStrategy
    end
    for each strategy s in lbsm
      convertToRuleBasedStrategyModel (s)
    end
  else if  fbsm is a simple strategy /* base case */
    add a new initial action rule: lbsm → lbsm.initialAction
    for each ActionTransition at in lbsm
      set up a-domain according to the strategy hierarchy
      add a new action rule: a-domain, at.startAction,
            at.condition → at.endAction
    end
  end if
end function
```

Algorithm 2 describes the reasoning algorithm with two parameters: *domain* and *currentAction*. The parameter *domain* refers to the strategy hierarchy of the strategy where *currentAction* is taken, and *currentAction* is the last action taken by the bidding agent. For example, when *domain* is *C*2.*S*1 and *currentAction* is *a*2, it tells the reasoning module that the last action taken by the bidding agent is *a*2, which is defined in simple strategy *S*1, and *S*1 itself belongs to complex strategy *C*2. Note that the last element of *domain* must be a simple strategy because a bidding action cannot appear in a complex strategy. However, if *currentAction* is null, *domain* can refer to either a complex or a simple strategy. In either case, the initial action of *domain* is returned as the next action.

On the other hand, if *currentAction* is not null, the reasoning module will search for the next action from the highest level of *domain*. Any transition at a higher level of *domain* has higher priority than transitions at a lower level. For example, if *domain* is *C*2.*S*1, the reasoning module first searches in strategy *C*2 for any possible strategy transition from *S*1 (i.e., the corresponding transition condition is *true*). If such a transition is found, say *S*1 can switch to *S*2 in *C*2, the next action will be the initial action of *S*2. Otherwise, the reasoning module searches in strategy *S*1 for any possible action transition from *currentAction*. If such a transition is found, say *currentAction* can switch to *a*2 in *S*1, the next action will be *a*2. Otherwise, if all levels of *domain* have been searched and no next action can be found, *currentAction* is returned as the next action.

Algorithm 2. Reasoning Engine

```
function Action findNextAction
        (Domain domain, Action currentAction)
  if currentAction == null
    if domain is a ComplexStrategy
      Search for initial strategy rule isr for domain that leads
      to initial strategy is
      return findNextAction(is, null)
    else if domain is a SimpleStrategy
      Search for initial action rule iar for domain that leads to
      initial action ia
      return ia
    end if
  else if currentAction != null
    Remove and process the first element fe of domain, and
    let the remaining domain be r-domain
    if fe is a ComplexStrategy /* strategy transition */
      Retrieve all strategy rules for the first element of
      r-domain and store them in a list
      while the list is not empty
        Remove and process the strategy rule sr at list head
        if the condition for sr is true
          Let s be the conclusion part of sr
          return findNextAction(s, null)
      return findNextAction(r-domain, currentAction)
    else if fe is a SimpleStrategy /* action transition */
      Retrieve all action rules for the currentAction and store
      them in a list
      while the list is not empty
        Remove and process the action rule ar at list head
        if the condition for ar is true
          return the conclusion part of ar
      return currentAction
    end if
  end if
end function
```

VI. Case Study

To demonstrate how bidding strategies can be easily specified and how execution of different user-specified strategies may directly impact the bidding history of an agent-based online auction, we developed a prototype agent-based online auction system. The system is implemented using JADE [28], where bidding agents can join and participate in multiple auctions at the same time according to user-specified bidding strategies. One of the important components of the system is a graphical user interface called visual strategy builder (VSB) that supports visual specification of layered bidding strategies. In the following sections, we first give a brief introduction to VSB, and then we perform experiments using different bidding strategies for bidding agents. Finally, we demonstrate how our approach can be used to analyze impacts of bidding strategies on bidding histories.

A. Visual Strategy Builder

Figure 11 shows the VSB interface for developing simple bidding strategies. The interface provides four buttons that can be used to select a particular component (e.g., an atomic bidding action or an action transition) for editing, create a new atomic action, create a new action transition, and delete a particular component. The supported atomic bidding actions and keywords for defining action transitions are listed in Table I and Table II, respectively. Figure 11 also illustrates a simple strategy *Normal_Strategy_S*1 in the canvas of VSB, which is equivalent to the normal strategy *S*1 defined in Figure 4. Note that the atomic action *DynamicBidAction* may take a parameter of bid increment, and in this case, it is equivalent to a *ChangeDynamicBidIncrement* action followed by a *DynamicBidAction* with none parameter. Figure 12 presents the interface that can be used to edit transition conditions. The interface also supports syntax check for user-defined transition conditions.

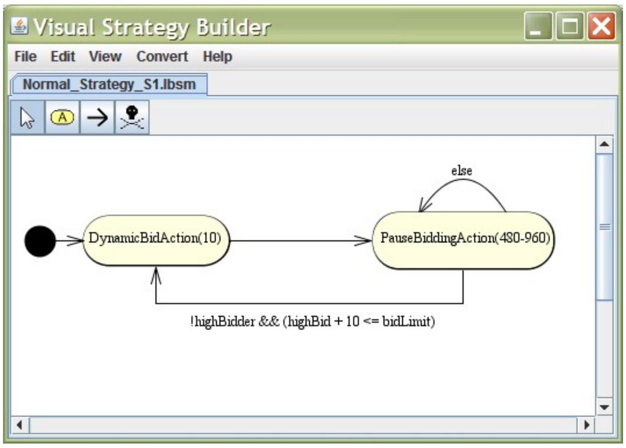

Figure 11. Visual strategy builder with simple strategy *S*1

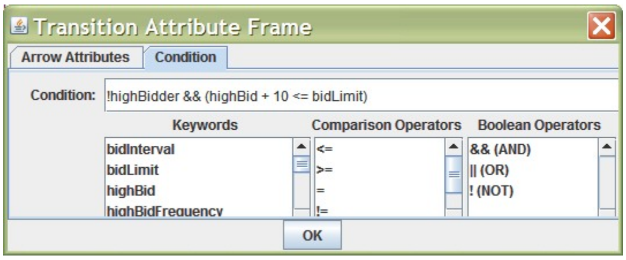

Figure 12. Interface for editing transition conditions

Figure 13 shows the VSB interface for developing complex bidding strategies. Similar to the interface for developing simple strategies, this interface also provides four buttons that can be used to select a particular component (e.g., a simple strategy, a complex strategy or a strategy transition) for editing, import an existing simple or complex strategy, create a new strategy transition, and delete a particular component. In Figure 13, we illustrate a complex bidding strategy *C*2 defined in the canvas of VSB. Note that *C*2 is composed of three simple bidding strategies, namely *S*2, *S*3 and *S*4, where *S*2 and *S*3 have been demonstrated in Figure 6 and Figure 7, respectively, and *S*4 is a strategy that simply wraps the atomic bidding action *StopBiddingAction* into a simple strategy. When *C*2 is selected to execute, the initial action in *S*3 will be executed first. While the value of *highBidFrequency* is no greater than 5, *S*3 is executed continuously. Once *highBidFrequency* becomes greater than 5, a strategy transition to initiate simple strategy *S*2

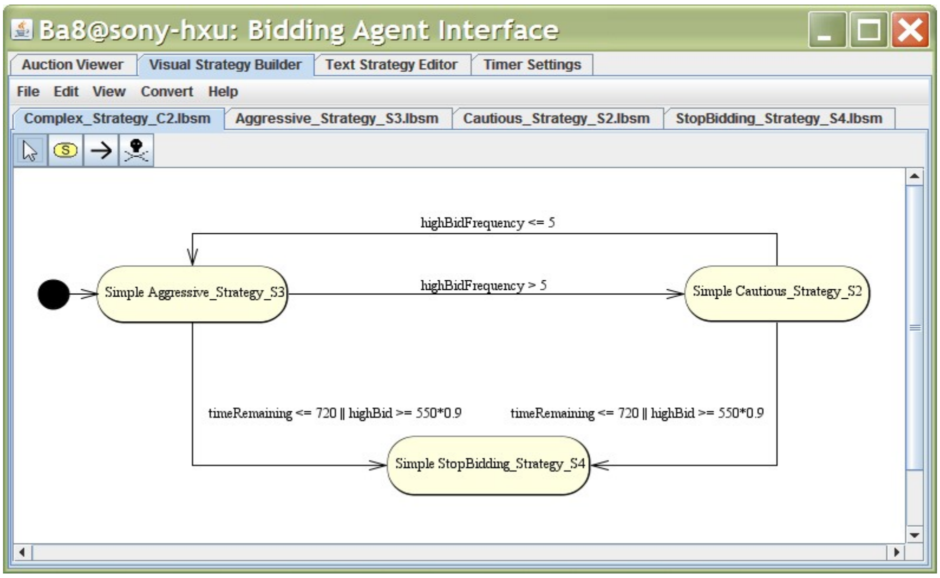

Figure 13. Visual strategy builder with complex strategy *C*2

is taken. Similarly, if *highBidFrequency* drops back to 5, *C*2 also switches back to simple strategy *S*3. Since *S*3 is an aggressive strategy and *S*2 is a cautious one, an agent using *C*2 is typically aggressive and only behaves as a cautious bidder when there is another sufficiently aggressive bidding agent existing in the same auction (indicated by the transition condition `highBidFrequency > 5`). From Figure 13, we can also see that when the remaining time is less than or equal to 720 seconds or the current bidding price is greater than 550*0.9 ($550 is the estimated price of the auctioned item), an agent using strategy *C*2 stops bidding. This indicates that an agent using *C*2 has no intention to win the auction, but only attempts to drive up the bidding price so the winner has to pay more than he otherwise would pay. We call such bidding behavior a *shilling behavior*, which is a type of auction fraud [17-22]. Detection of shilling behaviors in online auctions is a major focus of our other related research [3, 19, 29], but it is beyond the scope of this paper.

B. Experiments and Analysis

We now present a case study for a fictitious auction – for an item with an estimated auction price of around $550. The auction duration is 1800 seconds or 2 hours. In our first set of experiments, we run the auction with six bidding agents, namely *Ba*1 to *Ba*6. Agents *Ba*1, *Ba*2 and *Ba*3 follow *normal* strategy *S*1, *Ba*4 and *Ba*5 follow *cautious* strategy *S*2, and *Ba*6 follows *aggressive* strategy *S*3. Note that all bidding strategies *S*1, *S*2, and *S*3 have been defined in Section IV. Each bidding agent joins the auction at a random time during the *early stage* of the auction, which is defined as the first quarter of the auction duration (following the definition of early stage in [29]). We define bidding percentage rate (BPR) for $bidder_i$ during a period of time *T* as follows:

$$BPR_i = \frac{nBid_i}{\sum_{j=1}^{k} nBid_j}$$

where $nBid_i$ is the number of bids placed by $bidder_i$ during *T*, and *k* is the total number of active bidders during *T*. The value of BPR_i indicates how aggressive $bidder_i$ is in comparison with other bidders during *T*. Note that duration *T* can be the duration of a whole auction or a certain stage of the auction, e.g., the *final stage* of an auction, which is defined as the last 10% of an auction duration (following the definition of final stage in reference [29]).

We run the same auction 10 times with exactly the same settings. Figure 14 illustrates the bidding activities of the six bidding agents in the 10 auctions. The differences of bidding prices among different auctions are due to the random joining time of each bidder in different auctions and the random pause time associated with each bidder after a dynamic bid. The curve in Figure 14 shows the average bidding price of the 10 auctions vs. the auction time, where the average final bidding price amounts to $540.10.

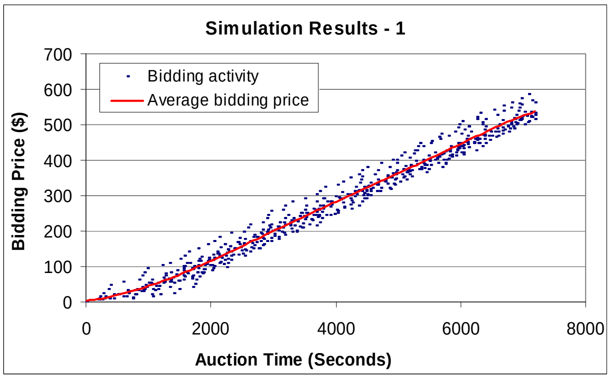

Figure 14. Bidding activities of six bidding agents

In Table III, we list the average BPR for each bidder (*Ba*1-*Ba*6) in all auctions, the average BPR for each bidder during the final stages of all auctions, and the number of auctions won by each bidder.

TABLE III.
SIMULATION RESULTS WITH SIX BIDDING AGENTS

Bidder	Bidding Strategy	Avg. BPR	Avg. BPR in Final Stage	Wins
Ba1	Normal	0.148509	0.167738	1
Ba2	Normal	0.138293	0.151071	0
Ba3	Normal	0.141673	0.147738	1
Ba4	Cautious	0.098521	0.086905	2
Ba5	Cautious	0.108336	0.043452	1
Ba6	Aggressive	0.364667	0.403095	5

From the auction analysis data, we can see that agent *Ba*6 contributes over 36% (in average) of the total bids for the 10 auctions, and over 40% (in average) of the total bids during the final stage of the 10 auctions. Thus, bidder *Ba*6 is an aggressive bidder. This experimental result is consistent with the fact that *Ba*6 follows an aggressive bidding strategy (*S*3) in the 10 auctions. As a consequence, *Ba*6 wins 5 out of 10 auctions. Similarly, the values of BPR for bidders *Ba*4 and *Ba*5 are lower than those for *Ba*1-*Ba*3; thus *Ba*4 and *Ba*5 are more cautious than *Ba*1-*Ba*3, which is consistent with the strategies taken by these agents. However, examining the wins for these agents show that the cautious bidding strategy is still a good strategy for the goal of winning auctions in this scenario. By analyzing the actual bidding strategies used by each bidder, we can see that although the values of BRP for agents with a cautious strategy is generally lower than that for agents with a normal strategy, agents with a cautious strategy still have a very good chance to win auctions. This is because in the current scenario, there is only one aggressive agent (*Ba*6) in each auction, who pauses around 4 minutes (240 seconds) after each dynamic bid. In this case, whenever the elapsed time since the last bid is greater than 3 minutes, a cautious bidder may have the chance to place a valid bid.

In the second set of experiments, we added a new bidding agent *Ba*7, who takes complex strategy *C*1 (described in Section IV). We again run the same auctions 10 times with 7 bidders. Figure 15 illustrates the bidding activities of the 7 bidding agents in the 10 auctions. The curve in Figure 15 shows the average bidding price vs. the auction time, where the average final bidding price amounts to $567.40.

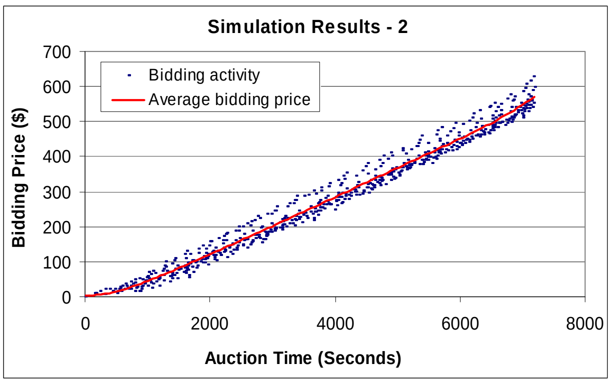

Figure 15. Bidding rates with different strategies

Note that the average final bidding price in the second set of experiments is a little bit larger than that in the first set of experiments. By analyzing the average BPR for each bidder (as illustrated in Table IV), we find that although *Ba*7 has a low average BPR for the 10 auctions, it has a very high average BPR during the final stages of the auctions. This indicates that *Ba*7 has been very aggressive during the final stages of the auctions, to complete with aggressive bidder *Ba*6 in order to win auctions. As a consequence, *Ba*7 wins a significant number of auctions (4 out 10), and the average final bidding price has also been driven up slightly.

TABLE IV.
SIMULATION RESULTS WITH SEVEN BIDDING AGENTS

Bidder	Bidding Strategy	Avg. BPR	Avg. BPR in Final Stage	Wins
Ba1	Normal	0.139651	0.129841	1
Ba2	Normal	0.136188	0.11873	0
Ba3	Normal	0.123372	0.095119	1
Ba4	Cautious	0.075075	0.012500	0
Ba5	Cautious	0.07826	0.044722	0
Ba6	Aggressive	0.340411	0.282579	4
Ba7	Complex_C1	0.107042	0.316508	4

On the other hand, when we look at the complex strategy *C*1 defined in Section IV, we find that *Ba*7 behaves in the same way as *Ba*4 and *Ba*5, with a cautious strategy before the final stage of each auction. This is the reason why the average BPR for *Ba*7 is close to that for *Ba*5 and *Ba*6 for the 10 auctions. However, during the final stage of each auction, *C*1 switches from cautious strategy *S*2 to aggressive strategy *S*3, so the average BPR for *Ba*7 during the final stages of the auctions increases significantly. We also notice that the average BPR during the final stages, for agents *Ba*4 and *Ba*5, drops significantly due to the competition between *Ba*6 and *Ba*7; thus cautious agents can hardly have a chance to place bids. As a result, both *Ba*4 and *Ba*5 do not win any auctions in our experiments.

In the third set of experiments, we further added a new bidding agent *Ba*8 with complex strategy *C*2 (defined in Figure 13). We run the auction 10 times with 8 bidders, and Figure 16 illustrates the bidding activities of the 8 bidding agents in the 10 auctions. The curve in Figure 16 shows the average bidding price vs. the auction time, where the average final bidding price amounts to $760.20. We notice that the average final bidding price of $760.20 is significantly higher than the average final bidding prices of $540.10 and $567.40 in the first and second set of experiments, respectively.

By looking at Table V for average BPR for each bidder, we find that for the 10 auctions, *Ba*8 has a higher average BPR than other bidders except *Ba*6. This indicates that *Ba*8 is a moderately aggressive agent for most of the time during the auctions. Thus, the auction prices have been significantly driven up due to the competition between *Ba*6 and *Ba*8. However, the average BPR for *Ba*8 in the final stages of the auctions is zero, which indicates that

*Ba*8 did not place any bids in the final stage of each auction. As a result, *Ba*8 did not win any auction. This indicates that *Ba*8 may have the malicious intention to drive up the bidding prices but avoid winning auctions. By analyzing the complex strategy *C*2 in Figure 13, we can see that *Ba*8 behaves as an aggressive bidder most of the time during the auctions. This is because *Ba*6 places a bid around every 4 minutes. Thus the value of *highBidFrequency* (the largest number of bids made by a bidder in the past 20 minutes) can hardly exceed 5. However, when it reaches the final stage or when the bidding price reaches the reserve price (defined as *estimatedPrice**0.9 in this paper, which is the lowest price at which a seller is willing to sell the auctioned item), *Ba*8 stops bidding. Such behavior is considered as a typical shilling behavior that can be easily simulated using our proposed framework. Thus, our approach also complements other research efforts such as shill detection using real-time model checking [29] by providing a useful test bed for evaluation purpose.

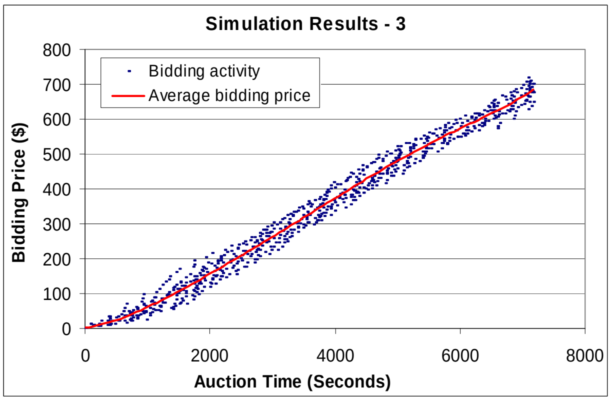

Figure 16. Bidding prices with different strategies

TABLE V.
SIMULATION RESULTS WITH EIGHT BIDDING AGENTS

Bidder	Bidding Strategy	Avg. BPR	Avg. BPR in Final Stage	Wins
Ba1	Normal	0.105421	0.093056	1
Ba2	Normal	0.104298	0.091667	1
Ba3	Normal	0.106738	0.068056	0
Ba4	Cautious	0.05764	0.033333	0
Ba5	Cautious	0.057567	0.055556	0
Ba6	Aggressive	0.297265	0.312222	4
Ba7	Complex_C1	0.084957	0.346111	4
Ba8	Complex_C2	0.186113	0.000000	0

VII. CONCLUSIONS AND FUTURE WORK

In an agent-based online auction system, the efficient specification of bidding strategies is a necessary component to make the system practical and usable. In this paper, we present a model-based approach to specifying layered bidding strategies for autonomous bidding agents. Our layered structure of specified bidding strategies allows human users to easily mix and match various strategies to create their own complex ones. By converting the layered bidding strategy model into a rule-based bidding strategy model, the bidding strategy can be efficiently executed by the bidding agent. The major significance of this work is to support model-based specification of flexible and complex bidding strategies by human users. Due to the layered bidding strategy model, our approach also supports reuse of bidding strategies including simple and complex strategies. For future work, we plan to extend our approach to support real-time inclusion of bidding rules enforced by the auction house. We also plan to integrate our implemented agent-based online system with the agent-based trust management (ATM) framework proposed in our previous research [3], and use the platform as a test bed for evaluation of real-time shill detection mechanisms. Furthermore, to study behaviors of common bidding strategies such as the bid sniping strategy, and to investigate how the presence of human bidders and different auction formats may affect the bidding process of agent-based online auctions are also envisioned as our future, and more ambitious research directions.

ACKNOWLEDGMENT

We thank the editors and all anonymous referees for the careful review of this paper and the suggestions for improvements they provided. We also thank C. Liu for her help with processing a large volume of experimental auction data, which significantly improved the presentation of the simulation results. This material is based upon work supported by the U.S. National Science Foundation under grant numbers CNS-0715648 and CNS-0715657.

REFERENCES

[1] P. Maes, R. H. Guttman, and A. G. Moukas, “Agents that buy and sell,” *Communications of the ACM (CACM)*, vol. 42, no. 3, Mar. 1999, pp. 81-91.

[2] D. G. Gregg, S. Walczak, “Auction advisor: an agent-based online-auction decision support system,” *Decision Support Systems*, vol. 41, no. 2, Jan. 2006, pp. 449-471.

[3] H. Xu, S. M. Shatz, and C. K. Bates, “A framework for agent-based trust management in online auctions,” in *Proc. 5th Int. Conf. on Information Technology: New Generations (ITNG 2008)*, Apr. 2008, Las Vegas, Nevada, USA, pp. 149-155.

[4] A. Lee, M. Lyu, and I. King, “An agent-based platform for online auctions,” in *Proc. Int. Conf. Internet Computing*, Las Vegas, Nevada, USA, 2001.

[5] W. Changjie, C. Xiaofeng, and W. Yumin, “An agent-based multi rounds online auction protocol with sealed bids,” in *Proc. 17th Int. Conf. Advanced Information Networking and Applications (AINA’03)*, 2003, pp. 194-197.

[6] J. Trevathan and W. Read, “A simple shill bidding agent,” in *Proc. 4th Int. Conf. Information Technology: New Generations (ITNG 2007)*, Apr. 2007, Las Vegas, Nevada, USA, pp. 766-771.

[7] S. Park, E. H. Durfee, and W. P. Birmingham, “An adaptive agent bidding strategy based on stochastic

modeling," in *Proc. 3rd Annual Conf. Autonomous Agents*, May 1999, Seattle, Washington, USA, pp. 147-153.

[8] H. Ma and H.-F. Leung, "An adaptive attitude bidding strategy for agents in continuous double auctions," *Electronic Commerce Research and Applications*, vol. 6, no. 4, Winter 2007, pp. 383-398.

[9] M. P. Wellman, A. Greenwald, and P. Stone, *Autonomous Bidding Agents - Strategies and Lessons from the Trading Agent Competition*, The MIT Press, Aug. 2007.

[10] VY Software, et al., *Ebay Bid Agent Software*, Free Download Center, 2009, Retrieved on Oct. 3, 2009, from www.freedownloadscenter.com/Best/ebay-bid-agent.html

[11] A. Ockenfels and A. Roth, "The timing of bids in Internet auctions: market design, bidder behavior, and artificial agents," *AI Magazine*, vol. 23, no. 3, 2002, pp. 79-87.

[12] A. Rogers, E. David, N. R. Jennings, and J. Schiff, "The effects of proxy bidding and minimum bid increments within eBay auctions," *ACM Trans. Web (TWEB)*, vol. 1, no. 2, Aug. 2007, Article 9, pp. 1-28.

[13] T. Skylogiannis, G. Antoniou, N. Bassiliades, G. Governatori, and A. Bikakis, "DR-Negotiate - a system for automated agent negotiation with defeasible logic-based strategies," *Data & Knowledge Engineering*, vol. 63, no. 2, Nov. 2007, pp. 362-380.

[14] B. J. Ford, H. Xu, C. K. Bates, and S. M. Shatz, "Model-based specification of flexible and complex bidding strategies in agent-Based online auctions," in *Proc. 6th Int. Conf. on Information Technology: New Generations (ITNG 2009)*, Apr. 2009, Las Vegas, Nevada, USA, pp. 894-900.

[15] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Third Edition, Addison-Wesley, 2004.

[16] J. Arlow and I. Neustadt, *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley Professional, 2001.

[17] R. J. Kauffman and C. A. Wood, "Running up the bid: detecting, predicting, and preventing reserve price shilling in online auctions," in *Proc. 5th Int. Conf. Electronic Commerce*, Pittsburgh, Pennsylvania, 2003, pp. 259-265.

[18] W. Wang, H. Zoltán, and A. B. Whinston, "Shill bidding in multi-round online auctions," in *Proc. 35th Hawaii Int. Conf. System Sciences (HICSS'02)*, Hawaii, 2002.

[19] H. Xu and Y.-T. Cheng, "Model checking bidding behaviors in Internet concurrent auctions," *Int. Journal of Computer Systems Science & Engineering (IJCSSE)*, Jul. 2007, vol. 22, no. 4, pp. 179-191.

[20] H. S. Shah, N. R. Joshi, A. Sureka, and P. R. Wurman, "Mining eBay: bidding strategies and shill detection," *WEBKDD 2002 - MiningWeb Data for Discovering Usage Patterns and Profiles*, LNCS 2703, Sept. 2003, pp. 17-34.

[21] J. Trevathan and W. Read, "Detecting shill bidding in online English auctions," *Handbook of Research on Social and Organizational Liabilities in Information Security*, M. Gupta and R. Sharman (eds.), IGI Press, Chapter 27, 2008, pp. 446-470.

[22] F. Dong, S. M. Shatz, and H. Xu, "Combating online in-auction fraud: clues, techniques and challenges," *Computer Science Review (CSR)*, 2009, in press.

[23] E. Gimenez-funes, L. Godo, J. A. Rodrguez-aguilar, and P. Garcia-calves, "Designing bidding strategies for trading agents in electronic auctions," in *Proc. 3rd Int. Conf. Multi-Agent Systems (ICMAS 1998)*, Jul. 1998, Paris, France, pp. 136-143.

[24] G. Governatori, M. Dumas, A. H. M. Ter Hofstede, and P. Oaks, "A formal approach to protocols and strategies for (legal) negotiation," in *Proc. 8th Int. Conf. Artificial Intelligence and Law (ICAIL)*, ACM, 2001, pp. 168-177.

[25] E. Kontopoulos and N. Bassiliades, "Graphical representation of defeasible logic rules using digraphs," *Advances in Artificial Intelligence (AAI)*, vol. 3955, 2006, pp. 529-533.

[26] P. Anthony and N. R. Jennings, "Evolving bidding strategies for multiple auctions," in *Proc. 15th European Conf. Artificial Intelligence (ECAI-2002)*, 2002, France.

[27] M. He, N. R. Jennings, and A. Prügel-Bennett, "A heuristic bidding strategy for buying multiple goods in multiple English auctions," *ACM Trans. Internet Technology (TOIT)*, vol. 6, no. 4, Nov. 2006, pp. 465-496.

[28] F. L. Bellifemine, G. Claire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*, John Wiley & Sons. Ltd., 2007.

[29] H. Xu, C. K. Bates, and S. M. Shatz, "Real-time model checking for shill detection in live online auctions," in *Proc. Int. Conf. Software Engineering Research and Practice (SERP'09)*, Jul. 2009, Las Vegas, Nevada, USA, pp. 134-140.

Benjamin J. Ford received the BS degree in Computer Science from University of Massachusetts Dartmouth, North Dartmouth, MA in 2008. He is currently a graduate student at the Computer and Information Science Department at University of Massachusetts Dartmouth. His research interests include e-commerce and agent-based technology.

Haiping Xu received the BS degree in Electrical Engineering from Zhejiang University, Hangzhou, China, in 1989, the MS degree in Computer Science from Wright State University, Dayton, OH, in 1996, and the PhD degree in Computer Science from the University of Illinois at Chicago, IL, in 2003. Since 2003, he has been with the University of Massachusetts Dartmouth, where he is currently an Associate Professor at the Computer and Information Science Department, and a Co-Director of the Concurrent Software Engineering Laboratory. His research interests include distributed software engineering, formal methods, e-commerce, Internet security, multi-agent systems, and service-oriented systems. His research has been supported by grants from the US National Science Foundation and US Marine Corps. He is a senior member of the IEEE Computer Society and a senior member of the ACM.

Christopher K. Bates received the BS degree in Computer Science from University of Massachusetts Dartmouth, North Dartmouth, MA in 2008. He is currently a graduate student at the Computer and Information Science Department at University of Massachusetts Dartmouth. His research interests include e-commerce, agent-based technology, and formal methods including real-time model checking.

Sol M. Shatz received the BS degree in Computer Science from Washington University, St. Louis, Missouri, and the MS and PhD degrees, also in Computer Science, from Northwestern University, Evanston, Illinois, in 1981 and 1983, respectively. He is a Professor in the Department of Computer Science at the University of Illinois at Chicago, where he is a Director of the Concurrent Software Systems Laboratory. His research interests include formal methods for specification and analysis of concurrent and distributed software, especially the application of Petri net-based models. His research has been supported by grants from the US National Science Foundation and ARO, among other agencies and companies. He is a senior member of the IEEE and a senior member of the ACM.