

Automated Trading across E-Market Boundaries

B. Schnizler, S. Luckner, C. Weinhardt
Chair for Information Management and Systems
University of Karlsruhe (TH)
Englerstraße 14
76131 Karlsruhe
{schnizler, luckner, weinhardt}@iw.uni-karlsruhe.de

Abstract: *Online trading has become an important topic in research and general public. As current trading platforms like eBay or Amazon usually require human interaction, these platforms have several drawbacks when the trading process is completely automated using software agents. This paper outlines the conceptual design of an architecture including necessary protocols enabling agents to automatically select market platforms, submit bids, and finally receive agreements. Achieving this, technologies derived from the Web service community are combined with the Montreal Taxonomy as a negotiation classification schema.*

1 Introduction

Online trading, specifically in support of online auctions, is an important topic in artificial intelligence and e-commerce research communities, as well as for the general public. Public auction sites such as eBay, Yahoo!, and Amazon list millions of items for sale and attract a multitude of users. At the moment, these auction sites require human interaction for participation in auctions, usually through web interfaces. Users select their preferred auction platform, select items they wish to purchase, submit bids, and monitor the status of the auctions manually. Some platforms support more advanced automation and monitoring tools like proxy bidding agents. These tools do, however, only provide limited functionality and still require human interaction to make strategic decisions.

But this is likely to change in the near future. The next generation of trading platforms requires automated negotiation facilities, allowing humans to be represented by software agents. In order to illustrate this, suppose the following scenario: A person has decided to purchase a new DVD player and communicates the preferences (model, price, features, location of purchase, and the like) to a software agent. Subsequently, this agent negotiates for the item on behalf of the person. Obviously, the interaction of these preferences can be complex, requiring advanced knowledge to be designed into the agent. The agent selects automatically an adequate auction site trading DVD players and meeting the person's preferences. Autonomously, the agent bids on the available players, making all strategic decisions until an agreement is reached.

A step towards this vision is already tackled using software agents for trading in electronic market platforms [MW06]. Agents are particularly useful in markets where trading might not have been possible otherwise, for example because a lot of information must be processed quickly or because employing human traders in 24-hour small transactions markets is not cost effective. For instance, markets for trading communication bandwidth are such applications [VP05].

At the moment automated trading using software agents is still inhibited by the multiplicity of different market mechanisms implementing a variety of institutional rules. While humans can easily understand the market rules and switch between different mechanisms, software agents have to be tailored to the mechanisms at hand. In practice, market participants usually select a mechanism due to the rules this particular mechanism offers. For instance, suppose a trader wants to buy a notebook and wants to negotiate over the size of the hard disk, the processing speed, and the price. In this case the trader may select a multi-attribute English auction, but not a single-attribute auction. A second example: Suppose a trader wants to bid for a security paper having a high probability of receiving this paper. This trader may select a high liquid market running a continuous double auction which offers a market order. These two examples show that the mechanism has to be chosen according to the market participants' requirements. Thus, the description of market mechanisms including their institutional rules has to be made explicit so that agents can understand these rules and select an appropriate mechanism according to their preferences.

Another obstacle in current platforms is the lack of standard interfaces for accessing market mechanisms. Recently, trading platforms started providing Web service interfaces. In case of eBay this allows for listing new items to the eBay site and for searching item listings using a variety of criteria. At present, eBay processes over a billion Web service requests per month. Amazon also offers a Web service so that client applications can browse its product catalogue and place orders. Although in both cases the Simple Object Access Protocol (SOAP) can be used to access the trading platform, there is no common interface.

Recapitulating, moving towards an automated negotiation environment including computerized agents implies several challenges upon the underlying technical infrastructure: Firstly, market mechanisms have to be described by an adequate taxonomy that enables a consistent mechanism propagation and discovery. Secondly, agents and market operators have to communicate based on standardized protocols.

The contribution of this paper is to tailor an architecture and fundamental protocols for a technically sound implementation of automated trading systems. This is achieved by applying taxonomical negotiation descriptions and technologies derived from service-oriented architectures and the Web service community to the implementation of agents and market mechanisms.

The paper is structured as follows: In section 2, the design requirements for an automated trading platform are elicited. With regard to these requirements, section 3 analyzes complementing work. Section 4 subsequently outlines an architectural proposal for an automated trading platform. Finally, section 5 concludes the paper and gives an outlook on future work.

2 Architectural Requirements

The design of a fully fledged architecture enabling the automated and autonomous trading by software agents implies several requirements upon the market discovery and the message exchange between software agents and market platforms. The following requirements can be identified:

1) Specification of institutional and platform requirements: An agent needs to express its institutional requirements upon the preferred mechanism as well as its preferred properties of the underlying market platform. Subsequently, an eligible market platform has to be discovered. Suppose an agent requires a multi-attribute single-sided mechanism for trading a notebook. The apt mechanism should support the negotiation of the price and the size of the notebook's hard disk. The negotiation process should come to an end within the next three days.

Moreover, the underlying market platform should support an encrypted message exchange including keys with at least 128bit length. Thus, the agent needs to formally specify these requirements followed by a meaningful market platform discovery, e.g. the discovery of a multi-attribute English auction mechanism ending in eight hours and including RSA encrypted message exchange with 256bit keys.

2) Interoperable invocation of market platforms: Agents require a flexible and interoperable way of accessing market platforms. As such, well-formulated interfaces and connection protocols have to be defined which regulate the seamless access to the market platforms. The interfaces and protocols have to be known by the agents and have to be implemented by each market platform.

3) Negotiation Protocol specification for submitting and receiving bids: Having discovered and invocated a suitable market platform fulfilling the institutional and architectural requirements, the agent requires a communication protocol to interact with the market. This means that the agent needs to retrieve status information like the current highest bid in order to adapt its trading strategy. Subsequently, the agent needs to know how to express and formulate bids in a common understandable meaning. Moreover, the agent needs to know how to submit these bids to the market platform. Thus, a meaningful negotiation protocol is required which specifies the syntax and the semantics of the message exchange between the entities involved in the bidding process.

After the negotiation process is completed, the market platform operator needs to inform the agents whether their bid was accepted or not. For this, a protocol is required to submit and receive agreements.

3 Complementing Work

Tackling the above mentioned requirements, the following complementing work has to be considered. For expressing the institutional requirements upon market mechanisms, we make use of existing parameterization approaches: Few attempts aim at decomposing auctions into their institutional rules which are treated as parameters [www01]. In order to define an auction, a list of parameters needs to be specified for configuring the market mechanism. By varying the specifications of the parameters, other auction mechanisms can be configured. As such, the configuration space of auctions that can be described parametrically is rather huge. Ströbel and Weinhardt extend this work to general negotiations [SW03]. Their Montreal Taxonomy allows for the description, characterization, and comparison of a broad variety of electronic negotiation designs, ranging from auctions to bilateral bargaining tables. It identifies several transaction phases and provides a common set of phase-relevant parameters for describing electronic negotiations. Parts of the Montreal Taxonomy have been realized by the Market Modeling Language, an XML based modeling language which provides a set of terms describing electronic markets [MW05].

Reviewing the existing approaches, the use of the Montreal Taxonomy is seen as promising for expressing institutional requirements and capabilities as defined in section 2. For specifying institutional parameters and platform requirements we make use of WS-Agreement, a standardization effort for defining an agreement establishment protocol [An05]. Agreement specifications provide a language and a protocol for advertising the capabilities of service providers and creating agreements based on creational offers as well as for monitoring agreement compliance at runtime. WS-Agreement – as a prominent specification within the Service Oriented Architecture and Grid communities – comprises an XML-based representation of agreements and agreement templates. It is capable of representing the requirements which an agent has on a preferred market platform. Thus, WS-Agreement is used to represent the institutional requirements as well as the platform properties.

Finally, a negotiation protocol is required for submitting and receiving bids. Reviewing the literature, there exists no complete negotiation protocol specification. As base for the design of

a new negotiation protocol, we revert to the Minimal Market Model [RNW04], the Auction Reference Model [RE05] and domain ontologies for defining the basic message exchange of our protocol.

The Minimal Market Model builds a market vocabulary based on structural similarities that many market mechanisms have in common. Such a vocabulary is needed for our message specifications. It comprises the concepts *Participant*, *Intention*, *Product*, *Attribute*, *Agreement*, and *Offer*. A *Participant* represents an agent of any kind that takes part in a market. Market *Participants* submit intentions to a market. An *Intention* thereby represents the smallest closed entity of purpose and refers to a *Product* which may be any good or service. *Attributes* are used to describe properties of products and market participants. Each *Offer* contains one or several intentions and serves as a container of control data for communicating these intentions. An *Agreement* requires matching associated intentions. For specifying the context-specific content of our messages, e.g. the product attributes for a certain product category, we utilize domain ontologies.

Apart from the content of the messages exchanged, software agents also have to know how to send these messages to market platforms. The Auction Reference Model, which is used to describe the internal structure and the functioning of auction mechanisms, proposes three operations: An operation for initializing the mechanism (*init*), another operation for submitting bids (*submitBids*), and finally a third operation for retrieving any kind of information, such as current bids or the agreement reached in the market (*getView*).

Having identified the complementing work, an architecture and its corresponding protocols can be derived.

4 An Architecture for Automated Trading

Tackling an adequate architecture for enabling automated trading by software agents requires the design of open interfaces as well as flexible components. Thus, the relevant components of the proposed architecture as well as their interaction are based on Web service standards in order to provide connectivity, efficiency, flexibility, immediacy, and interoperability of the whole system [La04].

The architecture is basically composed of three major components: several *Market Platforms*, a *Market Selector and Repository*, and a multitude of *Software Agents*.

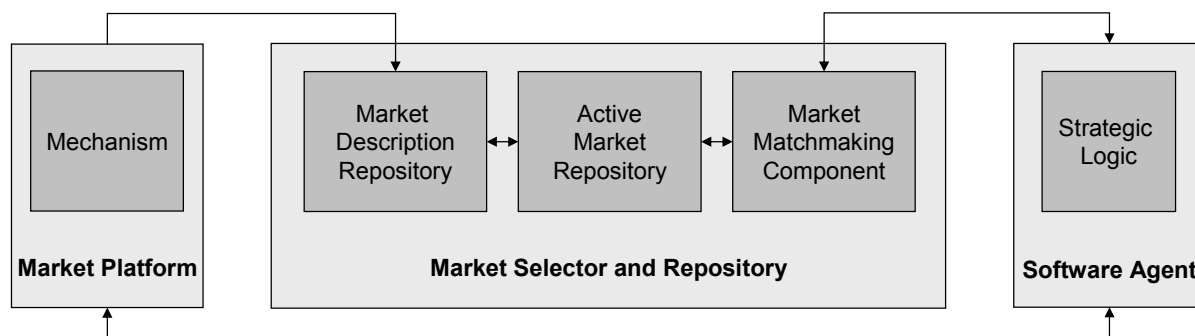


Figure 1: Architecture for an automated trading infrastructure

A high level overview on the proposed architecture is depicted in **Figure 1** and will be described briefly in the following.

A *Market Platform* component is a particular market *Mechanism* including its underlying technical infrastructure. The market platform as a whole is represented as a Web service and provides interfaces for initialization, submitting bids, and receiving trading information and agreements. Thus, a flexible and interoperable invocation by software agents is guaranteed. For instance, this component could represent an English auction with interfaces for submitting encrypted bids and receiving trading information as well as agreements.

The market platform can be described by its institutional rules (e.g. bidding rules, maximal number of participants) and its technical properties (e.g. response time, encryption standards) using the Montreal Taxonomy and the WS-Agreement specification language. These agreement offers, i.e. the capabilities which the market platform provides, are sent to a central *Market Selector and Repository* component in order to offer the market platform service to potential software agents.

The *Market Selector and Repository* component is responsible for storing all available market descriptions within a *Market Description Repository*. Instantiated and, thus, active market platforms are replicated in an *Active Market Repository*. The component is responsible for selecting an appropriate market platform based on an agent's requirements. This selection procedure is done within a *Market Matchmaking Component*.

Finally, the *Software Agent* component represents an autonomous trading agent. The agent has a *Strategic Logic* component which is responsible for specifying the preferences upon the preferred market mechanism. Furthermore, this component can submit bids and receive trading information and agreements. The preferences which an agent has upon a specific market platform are described using the WS-Agreement language. These specifications are submitted as a request to the *Market Matchmaking Component* which then selects an appropriate market platform in consideration of the *Market Description Repository* as well as the *Active Market Repository* and returns all relevant information (e.g. connection address) to the agent. After that, the agent can invoke the selected market platform, submit bids to it, receive relevant trading information, and, finally, receive agreements.

In the following, the market platform with its interface and the negotiation protocol between the market platform and the agents as well as the matchmaking component are described in more detail.

4.1 Market Platform

A market platform provides the actual trading functionality to market participants, i.e. the allocation and pricing of goods and services. eBay can be seen as an example market platform. All of the market operators have to agree on a common interface in order to facilitate trading across the boundaries of arbitrary platforms.

To foster interoperability, the functionality of a market platform can be used by exchanging SOAP messages. The service interface is specified by means of the Web service Description Language (WSDL) and comprises a list of provided operations with the messages these operations accept and return. The messages that can be exchanged with market platforms are defined with XML Schema Definitions (XSD). All required data types are declared within the WSDL file of the service. The message specifications are based on the Minimal Market Model which provides message templates. Thus, all the concepts that were presented earlier have to be mapped to XSD. Within such a message template, the in-depth information related to a certain transaction such as relevant product attributes is described by means of domain ontologies.

Based on these type and message definitions, the operations provided by the market platforms have to be defined. **Figure 2** shows operations from an example WSDL document used for describing such a service. The main operations are *initiate*, *submitOffer*, and *getView*. By calling the operation *initiate*, a new instance of a market mechanism is created. Market operators can e.g. call this operation for creating a new instance of the required market mechanism. The operation *getView* is used for retrieving information about the running instance such as the currently highest bid in case of an ascending auction or the agreement that was reached in an auction. Offers can be submitted to a market by invoking the operation *submitOffer*. More details on the implementation of such an auction service can be found in [Lu05].



Figure 2: Market service operations

Besides a description of the messages that can be exchanged, software trading agents need information about the state of a market. To give an example, an agent has to know whether his bids are still accepted or not, i.e. whether the goods or services have already been allocated or not. Such kind of information is part of the response message when invoking the *getView* operation. Based on this information and his knowledge about the market mechanism at hand, an agent can then derive a valid negotiation protocol and follow a predefined bidding strategy. In an ascending auction, an agent could submit offers with a minimum increment above the currently highest price as long as the highest bid is below his valuation and not his own bid. **Figure 3** shows an example for an interaction between a market platform and a software agent. First, the agent requests some domain knowledge like product attributes that are described according to a domain ontology. It then invokes the operation *getView* and thus gets some information on the running auction, such as the highest bid or the time remaining until the auction is closed. As long as bids can be submitted to the auction and the agent's valuation of the item sold is higher than the highest bid, he tries to overbid all competitors. The agent waits for a couple of seconds after submitting an offer before requesting another view. Furthermore, it does not overbid its own bids. After the auction ends and an agreement has been generated, the agent gets to know the market outcome by once more requesting its view which then contains information on the agreement.

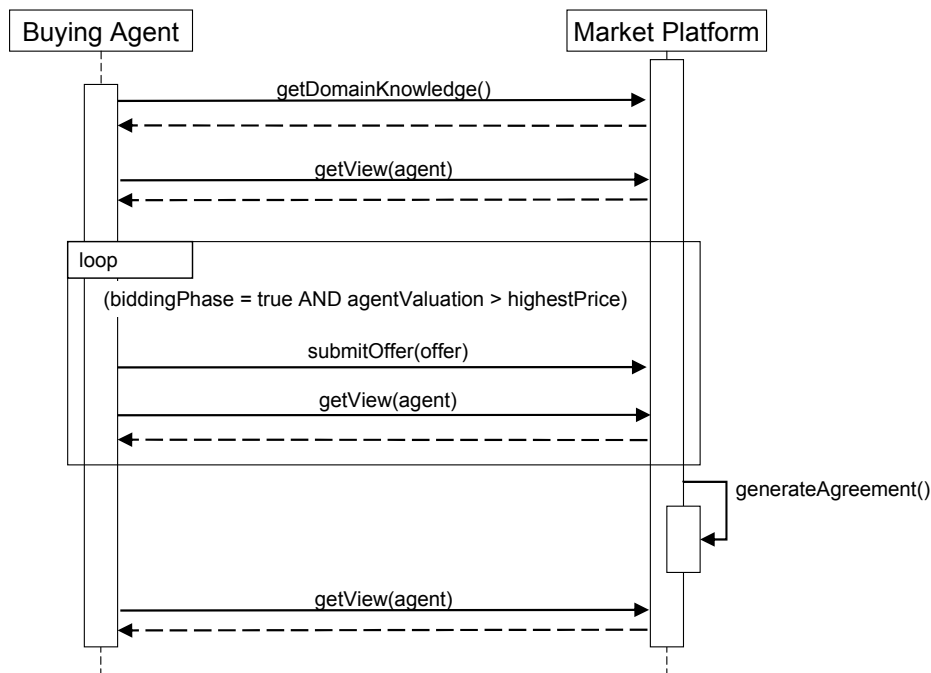


Figure 3: Interaction between agents and markets

4.2 Market Selector and Repository

Having designed the interfaces and protocols for a seamless invocation of mechanisms, the *Market Selector and Repository* component can be tackled. Reviewing the requirements defined in section 2 and taking the complementing work described in section 3 into account, the application of the Montreal Taxonomy can be used to describe the institutional rules of a market platform.

The Montreal Taxonomy is a comprehensive collection of concepts in an attempt to characterize all types of e-negotiations. It allows not only for the exact characterization and comparison of a broad variety of electronic negotiation designs and systems but could also lead towards a more structured approach to the design of electronic negotiations [SW03]. The taxonomy provides a set of classes including several attributes and characteristics. For instance, the class *Offer Submission* characterizes the offer submission process of a negotiation and – exemplarily – includes the attribute *Sides*. This attribute denotes whether one market side (*single sided*) or multiple sides (*multiple sides*) can submit offers. Thus, the *Sides* attribute has two possible characteristics, namely *single sided* and *multiple sides*.

Although the Montreal Taxonomy can be used to describe the institutional rules of a market platform, it is not sufficient for automating the discovery of market mechanisms. In fact, a language which can be used to represent the attributes and characteristics of the Montreal Taxonomy as well as the quality of service properties is needed. We therefore make use of WS-Agreements in combination with the Web service definition described in the previous subsection.

WS-Agreements are descriptions of the functional and non-functional properties of a service. These agreements can be used to encode the Montreal Taxonomy attributes and their offered, respectively required properties of the platform. **Figure 4** illustrates the main structure of a WS-Agreement [An05].

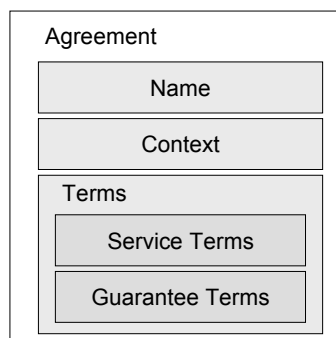


Figure 4: WS-Agreement structure

The agreement *Name* and *Context* contain, among others, information concerning the parties involved (in our case the market operators and agents). Functional and non-functional capabilities of the service are specified in *Terms*. *Service Terms* describe the service functionalities of a market (e.g. a function to submit a bid) while guarantee assurances for these functionalities can be specified with *Guarantee Terms* (e.g. the market platform has to support multi-attribute bids). The attributes of a negotiation described by the Montreal Taxonomy can subsequently be mapped to *Guarantee Terms*.

As another example, the market may have the Guarantee Term *Transaction*. The transaction term is an attribute of the Montreal Taxonomy and can have the characteristic *logged* (the transaction history is available to the agents) and *unlogged* (there is no transaction history available). Suppose an agent is requesting a particular market mechanism which has to offer the transaction history. The agent may specify this requirement using a WS-Agreement as shown in **Figure 5**.

The market operator can specify its institutional rules as well as the platform properties by a WS-Agreement offer and send this to the *Market Description Repository*. This repository can be a simple database storing all available market descriptions. The market operator can choose whether to instantiate the mechanism immediately or wait until an agent is requesting this particular mechanism. If the mechanism is instantiated, it will be replicated to the *Active Market Repository*.

```
<wsag:ServiceDescriptionTerm wsag:Name="submitOffer"
  wsag:ServiceName="Market"/>
<wsag:GuaranteeTerm wsag:Name="History">
  <wsag:ServiceScope>
    <wsag:ServiceName>Market</wsag:ServiceName>
  </wsag:ServiceScope>
  <wsag:ServiceLevelObjective>logged</wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```

Figure 5: WS-Agreement example

Whenever an agent wants to select a mechanism automatically, it also specifies its requirements using a WS-Agreement and sends this request to the *Market Matchmaking Component*. This component is a matchmaker selecting an appropriate mechanism on base of an agent's request and the existing mechanisms. This matchmaking includes a multi-attribute matchmaker, e.g. as described and implemented in [Ve03]. If the agent only requests instantiated platforms, the matchmaker restricts its search to the *Active Market Repository*. In other cases, the matchmaker will parse the complete Market Description Repository.¹ In case of multiple matching market platform descriptions, the matchmaker will return a predefined number of possible platforms to the agent.

Finally, the selected mechanism (in case a suitable mechanism is found) is announced to the agent. This announcement includes the description of the market platform and related metadata, for example the connection address.

5 Conclusion and Future Work

The paper at hand tailors fundamental protocols and architectural requirements for enabling automated trading systems. The approach combines taxonomical descriptions of negotiation mechanisms and technologies derived from the Web service community.

After an elicitation of the protocol and architectural requirements, a market platform for automated trading by agents is introduced. This includes the representation of the market platform as a Web service to enable a seamless invocation for agents as well as a negotiation protocol which defines the interaction and the message exchange between agents and market platforms. Subsequently, a market selector component is described which matches adequate market requests and offers. These offers are encoded using WS-Agreements and the Montreal Taxonomy.

Future work will include a detailed conceptualization of the matchmaking component especially the specification of decision rules if multiple market platforms would satisfy the requirements of an agent. The generality and the applicability of the protocols and the architecture have to be evaluated. This includes the extension to further existing taxonomical description languages of negotiations. Moreover, the presented work will be implemented as a proof-of-concept.

¹ It is also possible to restrict the search to non instantiated platforms. In this case, the matchmaker will consider all platforms of the *Market Description Repository* which are not in the *Active Market Repository*.

References

- [An5] Andrieux, A.; Czajkowski, K.; Dan, A.; Keahey, K.; Ludwig, H.; Nakata, T.; Pruyne, J.; Rofrano, J.; Tuecke, S.; Xu, M.: Web services Agreement Specification (WS-Agreement), Version 2005/09, GRAAP Working Group.
- [La04] Lawler, J.; Anderso, D.; Howell-Barber, H.; Hill, J.; Javed, N.; Li, Z.: A Study Of Web services Strategy In The Financial Services Industry. Proceedings of ISECON, 2004
- [Lu05] Luckner, S.; Rolli, D.; Momm, C.; Weinhardt, C.: Market Agents with a Sense for Mechanisms. IEEE International Conference on e-Business Engineering (ICEBE 2005), Beijing, China.
- [MW05] Mäkiö, J.; Weber, I.: Modeling Approach for Auction Based Markets. IEEE Proceedings of the 2005 International Symposium on Applications and the Internet (SAINT), 2005
- [MW06] MacKie-Mason, J.K.; Wellman, M.: In: Automated Markets and Trading Agents North-Holland, Amsterdam, to be published in 2006
- [RE05] Rolli, D.; Eberhart, A.: An Auction Reference Model for Describing and Running Auctions. 7. Internationale Tagung Wirtschaftsinformatik, 2004
- [RNW04] Rolli, D.; Neumann, D.; Weinhardt, C.: A Minimal Market Model in Ephemeral Markets. TheFormEMC, 2004
- [SW03] Ströbel, M.; Weinhardt, C.: The Montreal Taxonomy for Electronic Negotiations. Journal of Group Decision and Negotiation 12(2), pp. 143-164, 2003
- [Ve03] Veit, D.: Matchmaking in Electronic Markets, Springer, 2003
- [VP05] Vulkan, N.; Preist, C.: Automated Trading in Agents-based Markets for Communication Bandwidth, The International Journal of Electronic Commerce, 2005, to be published
- [WWW01] Wurman, P.; Wellman, M. P.; Walsh, W.: A Parameterization of the Auction Design Space. Games and Economic Behavior, 35, p. 304-338, 2001