Shaman: Software and Human Agents in Multiattribute Auctions and Negotiations♦

Gregory E. Kersten[1], Ryszard Kowalczyk[2], Hsiangchu Lai[3],
Dirk Neumann[4] and Mohan Baruwal Chhetri[2]

[1]InterNeg Research Centre, Concordia University, Montreal, Canada

[2]Centre for IT Research, Swinburne University of Technology, Melbourne, Australia

[3]National Sun Yat-sen University, Kaohsiung, Taiwan

[4]Institute of Information Systems and Management, University Karlsruhe (TH), Germany

Abstract

Three distinct and interacting types of entities: people, software agents and e-markets are considered in this paper. These entities operate within Shaman, a proposed framework for the construction and operation of heterogeneous systems enabling business interactions such as auctions and negotiations between software and human agents across those systems. Shaman is a dss-centric software environment which cooperates with and serves the users of distributed auction and negotiation systems. The dss are used to provide integration and coordination between the participating systems. Four such systems are discussed: Invite e-negotiation platform, enas negotiation agent suite, meet2trade auction platform and GoGo group buying software platform. The Shaman architecture based on these systems and the examples of their interaction enabled by Shaman are discussed.

♦ This work has been partially supported by the Natural Sciences and Engineering Research Council, Canada and the Social Sciences and Humanities Research Council, Canada.

Keywords: Decision support systems, E-markets, Distributed systems, Software agents, Auctions, Negotiation support, Group buying

1. Introduction

Both human and software agents engage in interactions with the purpose to acquire products or services, undertake joint activities and share information. The agents can meet at various places but in this paper we consider only virtual places. The virtual meeting places can be well established and organized, possibly with specialized services which the agents may use or provide. They can also be set up ad hoc and even be incidental.

Virtual meeting places have particularly become popular in e-commerce and e-business where they have mainly been used to facilitate simple interactions, information exchanges and electronic transactions between both the human and the software agents. However, most existing e-commerce systems have focused on the technical infrastructure and its efficiency rather than the operational effectiveness in supporting high-level decision-making processes people typically get involved in as they engage in business interactions such as auctions and negotiations. According to the estimates of the National Association of Purchasing Management (NAPM), the average time cycle for setting up a master purchase agreement is 12 weeks [40]. Currently available matching, cataloguing, and document and fund transfer technologies may reduce this cycle slightly, but the bulk of the time is taken by the negotiation and conflict resolution processes. Therefore there is a clear need to improve efficiency and effectiveness of the decision-making processes of business exchanges and negotiations in virtual meeting places.

Virtual meeting places are provided and supported by software, for example, e-markets. The supporting software may include software agents that together with human agents use information systems in order to access virtual meeting places and undertake activities there. This necessitates some form of organization of the interactions among the agents and structuring of their participation in the exchange processes. Both can be achieved with protocols. The need for a protocol that formalizes the interactions among agents and allows them to interpret information they obtain has been recognized and different protocols proposed [21, 46]. If human agents interact via software and transact in places organized and controlled by software they also have to follow certain rules [26].

There are many available choices to agents in terms of exchange mechanisms, communication forms and media, and decision rules, to name a few. People may use different systems, including software agents, to deliberate, assess information and make decisions; individually or in groups. If people interact with systems which have some degree of intelligence, they may expect to be able to delegate certain responsibilities to these systems and the systems, in turn, will interact with people effectively. Intelligence and autonomy, which are important characteristics of software agents, raise the issues of expectations, trust, reliability and confidence.

One result of the growth of e-commerce and e-business is the increase in the types and forms of information sources and the exchange mechanisms. The three main types of exchange mechanisms are catalogues, auctions and negotiations; there are many variations of each as well as combinations of two or more mechanisms-types. The agents may thus choose from

many different exchange mechanisms; some of which are complex and difficult to conduct.

Many auction and negotiation mechanisms require cognitive and computational capabilities, and time which people may not have or may be unable to allocate. Decision and negotiation literature gives many examples of biases caused by the lack of understanding or insufficient consideration of decision-making principles and mechanisms (e.g. based on utilities and conditional probabilities), overconfidence and reliance on easily available information [6, 22]. Simultaneous formulation of several offers of the same value (utility), which is suggested by negotiation experts, is computationally difficult unless a negotiation support system is used. Similarly, combinatorial auctions can hardly be conducted without support from specialized software.

We consider three distinct and interacting types of entities: people, software agents and e-markets. Various configurations of human and software agents are possible; they may differ in the allocation of roles and responsibilities to them, and in the selection of e-markets, and their mechanisms and protocols. This brings forth the question of the design of software agents which can engage in commercial activities, cooperate, compete and integrate in order to achieve good deals for these agents' principals. It also brings an issue of the ability and willingness of the people to effectively interact with software agents and the agents' ability to respond to peoples' demands.

The two typical conceptual frameworks to address the issues of the cooperation between people and software agents are:

1. Giving the agents more intelligence and equipping them with conversational and other human-like interaction capabilities; and
2. Expanding and enhancing decision support systems (dsss) with intelligence and other capabilities similar to those of software agents and providing them with environment monitoring and effecting tools.

The first type is popular in the AI community which strives at providing software agents with more and more human-like capabilities, including cognition, learning, synthesizing and conversing [53]. The second type is popular in the IS/DSS community which tries to give new life to the well established but much less popular systems [47, 48].

This paper proposes a different framework for the construction of heterogeneous systems involving people and software agents. The framework is based on the following three key observations**:**

1. People and software agents who represent them and act on their behalf comprise a heterogeneous community in which people are sovereigns;
2. People need to be able to rely on the agents while being also able to take over at any time any task an agent is undertaking on their behalf; and
3. Agents need to be able to observe people's behavior so that they can learn and return control when people decide to undertake some tasks on their own.

The first observation has been widely accepted.[1] Irrespectively of the users' limitations; they must have a final say and may opt not to use an agent or to terminate the agent's service at any time.

The second notion has been studied with the focus on providing the software agents with abilities which make them more trustworthy, reliable, likable, etc. Typically the agents are designed to perform a task until its completion. Little work has been done on providing people with the ability of taking over the agent's task at any time and performing it effectively and efficiently. The logic appears to be that if a person decides to take over a task from a capable artificial intelligent agent, then the agent cease to act and the person is on her own. We argue that this is a self-limiting approach and the agents should be capable of stopping at any time and also taking over from the person in the middle of a task, if the task nature allows for this. The possibility of such collaboration between agents and their principals makes agents more trustworthy.

The third observation reflects the dynamic nature of reality; both people and software agents change because of the changes in their environments. However, people may also change because of their inner discourse or for any other reason which is not known to the agent. Yet an intelligent and useful agent should have an ability and make an effort to recognize such a change.

The architecture which is proposed here retains the old concept of the design of DSSs and their different variations (GDSS, NSS, ENS) where the human user has the intelligence and makes decisions, and the DSS has models, solvers and tools to support decision making [1, 44]. Its purpose is to provide a common platform for enabling interactions and cooperation between *software and human agents in multi-attribute auctions and negotiations* (Shaman). The architecture overview from the bird-eye perspective is presented in Figure 1.

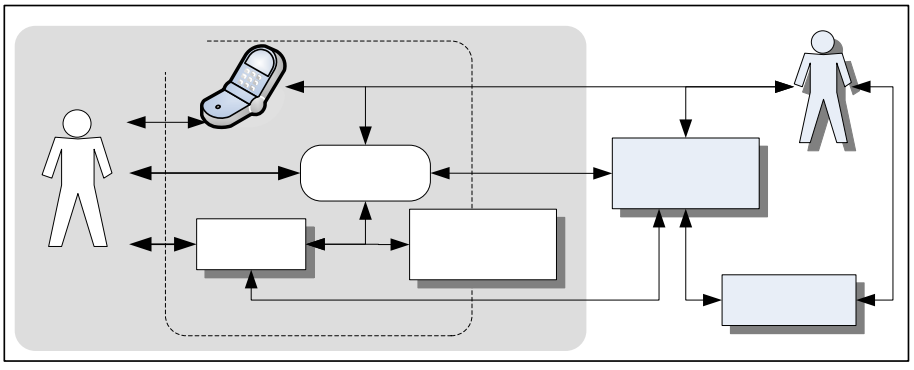

Fig. 1. A high-level perspective of the Shaman architecture

[1] We note that there may be limitations to this observation and ethical issues associated with its blind following. Can, for example, a person with restricted mental capacity turn off an agent which controls the system supporting this person's life? In most applications, people consider the ability to recall an agent an obvious one.

The three underlying principles for Shaman are:

1. Every system which directly or indirectly acts on behalf of and communicates with the person-user has to inform the dss and make all information obtained and produced available to dss.
2. The person-user communicates with external entities, people and systems, via the dss facilities.
3. All components of the agents which are working for the user-principal, with the exception of the components pertaining to intelligence and mobility, are available to or are components of the dss.

We propose a DSS-centric software environment which cooperates with and serves its user. Following the above principles the DSS's role is similar to that of a decision station which is a support system situated in the external environment [48]. The DSS is situated not through the use of its environment sensing and effecting components as it is the case with the decision station. Instead, the DSS receives information that the software agents collect (e.g. through the exchange process with other agents and e-market participants) and it has the models and tools which they use. The DSS retains its primary purpose, which is providing support to the user whenever the user wants it and—as it has been customary in the past—it is passive and does not act on its own.

The DSS gives its user a uniform and familiar interface. The user may interact directly with a software agent, but every software agent has to be also able to interact via the DSS. This simplifies the users interactions with both her own and foreign agents (i.e., from the outside of the user's environment).

Two other roles of the DSS may be more important for the functioning of the overall environment. One is its ability to provide the person-user with models and tools that she may utilize to make decisions and engage in the interaction with external agents on the market. This enables the situation when the user takes over the tasks of some or all of the agents from her environment.

The second role is the provision of a single environment in which all other devices either reside or communicate with. The person-user communicates her decisions and interacts with others only through the DSS. To illustrate this point consider the following case.

> *Mik uses a software agent to sniff and make bids on eBay. He is now bidding for a pair of very special sunglasses. Late Saturday evening Mik checks the state of the auction and notes that the current bid price is $50 with the "buy it now" price being $90. Earlier in the evening Mik's friends told him that they would buy 4 pairs of these sunglasses but for no more than $60. Mik suspends his agent and sends an email to the auction owner saying that he would buy 5 pairs of sunglasses at $50 for a pair. The deal goes through. But the software agent has no clue what has happened.*

One difficulty in using software agents is that people have to communicate only through them. A person is responsible for the communication and has to make sure that old and newly introduced agents alike receive information they require to perform tasks. If this person disables an agent or takes over a part of an activity, other agents have to be informed. And if the person changes her preferences or gets new information, again she has to notify

all agents that may require this information.

With the convergence of technologies it is increasingly possible to thread information and tools according to their content. If Mik used the DSS to write emails, this information would have been available to software which has access to this DSS. This does not mean that the information would have been useful to software agents or other programs; it may be necessary for Mik to structure it and explain its significance. However, it is also possible, that an intelligent software agent realizes that Mik sent an email to the auction owner with whom this agent was interacting and purchased five pairs of sunglasses. This agent then queries Mik as to the possible relation so that in the future the agent may use this information. This way the agent may, for example, search for other buyers of the same product and suggest a group purchase.

There have been many frameworks, models and architectures for e-markets discussed in literature. The framework proposed here is centered on the person-user while taking into account all entities participating in commercial decision-making. The second important feature of the framework is its architecture and multi-disciplinary methodological basis discussed in Section 4 and 5.

Several components of the framework have been built for or can be adapted to its testing and evaluation. The four main components are the Invite e-negotiation platform (Section 3.1), eNAs negotiation agent suite (Section 3.2), meet2trade auction platform (Section 3.3) and GoGo group buying software platform (Section 3.4). These components together with relevant experiments are discussed in Section 3. The Shaman framework and its architecture are discussed in detail in Section 4. Section 5 concludes with a summary and an outlook on further research.

2. Foundations

The Shaman framework is based on the following foundations: (1) decision aids and support systems, (2) software agents, their roles and environment, (3) auction and negotiation protocols and taxonomy; and (4) e-markets. These concepts and their significance are discussed in the remainder of this section.

2.1 Decision support systems

Decision support system is the area which has traditionally been focused on managerial problems.[2] Beginning with the seminal work of Gory and Morton [16], where the term 'decision support systems' first appeared until today, the primary concern of DSSs is with business and managers [3]. With many applications being designed for consumers and businesses alike, the orientation on managerial support may have became a weakness.

[2]Several of the arguments presented in this section have been earlier formulated by Vahidov and Kersten [48].

Silverman, Sprague, Carlson and others note [42, 43] that the need for decision support in the age of the internet and e-business is now becoming even more critical than before. This requirement is not reflected in the breath and depth of DSS research and applications. DSSs were one of the most popular areas of research in information systems in the 1980s and 1990s, but lately the interest in DSS appears to be waning. Arnott and Pervan's [3] analysis of the professional and practical contributions of DSS research shows a field that is facing a crisis of relevance. They report that half of DSS research has low or no practical relevance, and only around 10% of papers are rated as having high or very high relevance. These findings call for a closer analysis of requirements for decision support tools posed by the new dynamic environment.

Traditionally DSS research and software development focused on solving generic decision problems, in particular, on the preference elicitation and utility construction process and the construction of the formal problem representation. The sphere where actual business operations or transactions took place was often seen as a secondary concern for the adoption and implementation of DSS. While criticism of the "stand-alone" DSS approach and the need for closely linking DSS with business work processes have been voiced [5], this theme has not yet resulted in the introduction of new concepts, frameworks or architectures [48].

The requirement of the DSS connectedness to its environment builds upon the concept of an active DSS [2, 35]. The advocates of active DSS point out the weakness of the traditional support for being passive, where the user has to have full knowledge of the system's capabilities and must exercises initiative to perform decision related tasks. An active DSS need not be capable of undertaking all the tasks on its own and complete them without the user's intervention. Ideally, a proactive DSS would establish a two-way interaction with both the user and environment, and would be capable of maintaining those links if any of the entities is active. Such a system would allow for the integration of decision-making and decision implementation activities.

Effective linking of DSSs to their problem environments would enable improvement of strategic capabilities of organizations through timely response to the dynamically arising challenges and management of organizations "by wire", that is, combining high level decision-making with automation and IS support of various business operations [19]. The terms "cockpit of the business" and "cyberspace cockpit" have been coined to signify the new requirements for computer-based support. Furthermore, we're witnessing an increased interest in the real-time, more responsive breed of DSSs [47]. On the consumer side, the predictions are made about the emergence of a "new breed of consumer ... more selective, better informed, and with a range of powerful tools at his or her disposal".

The difficulty in the design and implementing active and connected DSSs was caused by the lack of connectivity among different information systems in an organization and between these systems and the organization's environment. The ubiquitous network and pervasive computing demand new approaches that will allow a DSS to become a part of the information infrastructure, through interaction with its environment, and leveraging its cognitive capabilities through its connectedness to the decision problem's environment. This would provide a DSS with means to sense the problem environment, offer decision support to a decision maker and act upon the environment to adequately respond to his/her needs that

may undergo changes and refinement during the process. In other words, these systems will be the *situated* decision support systems.

The above discussion stresses recent trends regarding the adoption of the connected and situated systems. The goal of a situated, connected and active DSS is to provide all services necessary for decision-making and implementation. To reflect the comprehensive nature of such a system and also its integration with other systems and with the environment Vahidov and Kersten [48] call it a decision station (DS). Thus, a DS can be seen as a software component of a dedicated workstation. A DS is used to: (1) sense what's going on in the problem domains; (2) utilize traditional DSS facilities to inform decisions; (3) make choices; and (4) undertake implementation and monitoring activities.

2.2 Software agents and MAS

Text (single spacing) Agents present a compelling vision of future computational systems [13]. They are to exhibit such characteristics as intelligence, awareness and flexibility all of which promise great advantages to the way we do business [21, 40]. In particular, systems that use software agent technologies are proving to be effective in helping users make better decisions in various stages of the exchange process, including, product finding, supplier finding, product ordering, delivery monitoring, etc. [4].

Software agents can also play an important role in providing support and automation for the negotiation stage of online trading [18]. Early examples of such systems include PersonaLogic and Kasbah [9]. More sophisticated automated trading systems have been proposed, which offer multi-attribute intelligent matching such as MIT's Tete-a-Tete [18] and ITA [30]. However, most of these systems support simple one-to-one negotiation between the participants.

There are a number of technical and theoretical difficulties that need to be resolved before these systems realize their full potential. Moreover, the problems of supporting one-to-many negotiations and interactions between human and software agents are even harder. The existing agent-based negotiation systems rely mostly on rigid rules and are highly structured. They often use economic and game theoretic techniques in mechanism design in order to set up auctions that guarantee certain properties [38]. Such settings have various advantages, but fail to support scenarios in which less structured and more flexible negotiation involving both human and software agents are needed.

Despite the lack of a well-formulated and widely accepted definition of the concept of software agent [15, 53], we adopt a natural metaphor view of an agent [20] based on synthesizing the relationships between software agent capabilities and relevant tasks in different negotiation phases within a coherent framework.

The first important issue to be addressed is what types of agents can be useful in supporting negotiation tasks. Franklin and Graesser [15] have proposed a classification scheme for agents based on the properties they possess. Nwana and Ndumu [39] have identified autonomy, cooperation, and learning as subset of dimensions for deriving classes of agents. In their schema, agents possessing cooperation and autonomy features would be referred to

as "collaborative agents", while those with learning and autonomy properties would be described as "interface agents".

Table 1. Examples of human agent negotiation roles and main tasks

Agent role	Description
Principal	Problem owner with the ability to delegate
Stakeholder	An agent materially interested in the results of the exchange
Negotiator	Participates in an auction and/or negotiation
Coordinator	Collects information about and coordinates activities of others
Advisor	Recommends and advices another agent; critiques and proposes (candidate) offers
Mediator	Helps agents to engage in or conduct an exchange; proposes offers; suggests concessions
Expert	One who has domain knowledge not available to non-expert agents

Agents possessing all three features were identified as "smart" agents. In order to conceptualize the role of agents in deal making, it would be useful to think of the negotiation situations along two dimensions. One relates to the cooperative behavior of the agents (e.g. willingness of the negotiators to disclose their private preferences to a third party), which promises to make the negotiation process more efficient. The other one relates to the degree of certainty regarding negotiator preferences and strategies (i.e. the degree to which the negotiator's task can be regarded as being "structured").

One way to determine types of agents useful in e-negotiations is to consider the roles people play in negotiations. There are seven main such roles and they are listed in Table 1. Every one of these roles can be played by a human or by an agent. In some, very complex negotiations, for example those involving state governments there may be additional and more specialized persons involved in the process. These people and also some concrete tasks they undertake may be used to determine the types of software agents that may be involved in e-negotiations. Such a list is given below.

- User profile agent. The purpose of this type of agent is to elicit user preferences, and to assist the negotiator in deciding on objectives and strategies. Ideally agents of this type would be able to adapt to the changes in user behavior in the process of negotiations.
- Information agent. Agents of this type would engage in actively seeking, retrieving, filtering, and delivering information relevant to the issues on the table.
- Opponent profiling agent. The primary purpose of this agent type would be to identify the objectives, preferences and strategies of the opponent. Knowing the opponent better makes offer generation and evaluation a better informed decision making process. The information and opponent profiling agents could be regarded as "intelligence" agents.

- Proposer agent. The aim of this type of agent is to generate a set of promising offers to be considered for submission to the opponent. In negotiation problems which involve multiple issues, the generation of an offer may involve search in a very large space of possible offers.
- Critic agent. The purpose of the critic is to evaluate the offers received from and addressed to the opponent and provide "verbal" feedback on the drawbacks and, possibly benefits of these offers. The proposer and critic agents could be regarded as a type of "adviser" agents.
- Negotiator agent. This agent may be capable of conducting negotiations in a semi-autonomous or fully autonomous fashion. Applicability of full automation depends on the degree of certainty in objectives, preferences, and tactics of the negotiator (i.e. the level of structuredness of the negotiation task from the negotiator's perspective).
- Mediator agent. The main purpose of this agent is to coordinate the activities of the negotiating parties, and to attempt to generate mutually beneficial offers. The role of this agent increases when the parties are willing to provide their information to a third party agent.

2.3 Auction and negotiation protocols and taxonomy

The phase model of deal-making allows for a structured approach to negotiation preparation and conduct. It also facilitates modeling and assignment of activities. The model presents the process as a sequence of well-defined phases with each phase having a different purpose and several specific activities. The model positions the negotiators in the centre of the process; they undertake the activities and move from one phase to another. As each activity is complex, it is broken into specific tasks and actions.

The phase model allows for linking decision-making, communication and negotiation concepts with perceptions, understanding and context. It also allows us to position the negotiation in a broader context and highlight the fact that the achievement of the compromise is neither a simple process nor is it the conclusion of the process.

The seven-phase model presented in Table 2, is based on Gulliver's model [17] applied in the design of e-negotiation systems [24], and extended with two additional phases (Matchmaking and Updating). These two phases have been added in order to bridge auctions and negotiations and include processes in which software agents participate.

Table 2. Deal-making phase model

Phase	Description
1. Planning	Construction of the representations (partial or complete): problem, own interests and requirements, potential participants and the process (deal-making protocol).
2. Matchmaking	Assessment of the potential participants, their assessment and election.
3. Exploring	Updating information about the problem and participants; detailed specification of the process.
4. Offer exchanging	The parties make (exchange) offers and, if the protocol allows, supporting arguments and promises.
5. Reaching agreement	An agreement is reached or it is sufficiently close to use a simple decision rule (e.g., split the difference).
6. Concluding	Post-settlement discussion, search for joint improvements, fulfillment, verification.
7. Updating	Review of the process and its outcomes, lessons learned; updating of the knowledge bases.

The deal-making process begins with the planning phase which begins when the decision is made regarding an exchange. The following phases are subsequently executed; however it is possible for the participants to return to a phase that was executed earlier or to bypass a phase.

The key concepts used to specify a negotiation protocol are presented in Figure 2. The process model, strategies, tactics, and activities are derived from behavioral negotiation theory, approaches, and models form the theory-based specification part. Behavioral theory posits that activities depend on the negotiators' characteristics and the negotiation context (e.g. power distribution, relationship, and the relative importance of outcomes). The characteristics and the context determine the negotiators' approaches, their strategies and tactics leading to the selection of specific activities in the different phases of a negotiation.

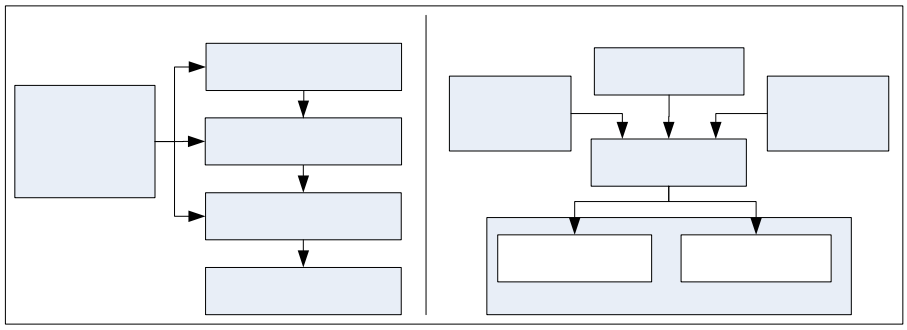

Fig. 2. Theory and protocol-based activity specification

Behavioral research does not provide sufficiently precise insights into the selection of activities required for the specification of an e-negotiation protocol, because of the number of possible combinations of negotiators' characteristics, interdependencies between characteristics of negotiators, dependence of the negotiators' behavior on external factors (e.g. relationship with other stakeholders or the consideration of future situations) as well as the complexity of the problem and process [25].

Hence, the specification of a negotiation protocol and, thus, the selection of activities depend—as illustrated in Figure 2—not only on the process model, the selected strategy and tactic models, but also on assumptions on the part of the protocol designer about "useful" activities and their assignment to negotiation phases (protocol-based specification part).

Table 3. Characteristics of negotiations and auctions

Aspect	Description
Issues	One, few, many
Problem	Fixed, modifiable, evolving
Parties	Two, few, many
Protocol	Fixed, modifiable, open
Information exchange	Uni-, bi-, multi-directional
Information content	Complete offer, partial offer, message, mix

2.4 E-markets and other meeting places

Along with the increase of the number of participants in online interactions and with the increase of the variety and number of products, services, collaborations and other arrangements, the complexity of these interactions increase. There are many possible configurations and types of markets, stores, agora and other meeting places where

transactions may take place. They may be set up ad hoc for one particular purpose or for established places providing a variety of services; they may be commercial or not-for profit. To simplify we use the term e-market for all these forms.

E-market is an information system which provides virtual space for its participants to exchange information with the purpose of at least one participant providing certain information or a physical good or service to one or more other participants.

Note that this definition does not require payments or any other activities typical for commercial markets. A place which is used by townsfolk to discuss budget and make concrete proposals and demands to town councillors is, according to this definition, an e-market.

3. Four platforms

In order to study the interactions between different types of agents that use various markets and engage in different processes we need to clarify the key components, i.e., markets, agents, environments and processes. Selected software programs and platforms which have been used to study auction and negotiations in e-business are discussed here. They all provide many of the components for the Shaman environment.

3.1 Invite e-negotiation platform

Invite (InterNeg virtual integrated transaction environment) is a software platform designed to construct, in real-time various e-negotiation or auction systems in an integrated environment [25, 27, 45]. The generation of both auction and negotiation systems is based on predefined negotiation protocols [26]. Invite can generate, among others, several versions of the Inspire system. At present, these systems are used for research and training purposes.

Foundations. Activities—from the perspective of the negotiators— are the most concrete elements of a negotiation. They are, however, not well-suited as abstractions for the development of enss. As shown in Figure 2, activities are formulated based on negotiation theories, approaches and models. In order to describe the Invite prototype and its use in electronic negotiations, we take a bottom-up approach and begin with the representation of activities.

General architecture. Invite platform is based on a three-tier software architecture built on the Fusebox framework, which enables the model-view-controller (mvc) design. The three types of components and their main subcomponents implemented in Invite are depicted in Figure 3.

Invite generates an ENS instance based on the negotiation manager or a user who requests a particular type of the negotiation. This is done by the controller that extracts the negotiation protocol (process model) that corresponds to the requested type. The protocol and other complementary models determine the type of negotiation and the type and content of information exchanged between the negotiators via the system and between the negotiator

and the system model-type components. The view-type components are used to compose and layout web pages and insert navigation links into these pages (Figure 3).

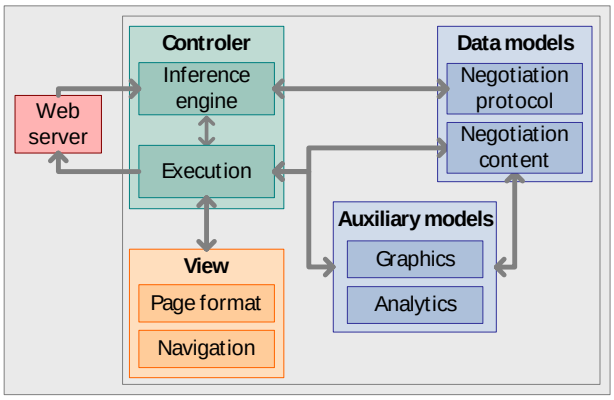

Fig. 3. Overview of the Invite platform

The Invite platform has been designed to allow execution of different negotiation processes defined by protocols. It also allows for the parties to follow different protocols; in effect each party may have different abilities determined by this party's protocol. Figure 4 shows that each party in a bilateral negotiation is using their own instance of an Invite ENS. The coordination of the two instances is achieved through their controller.

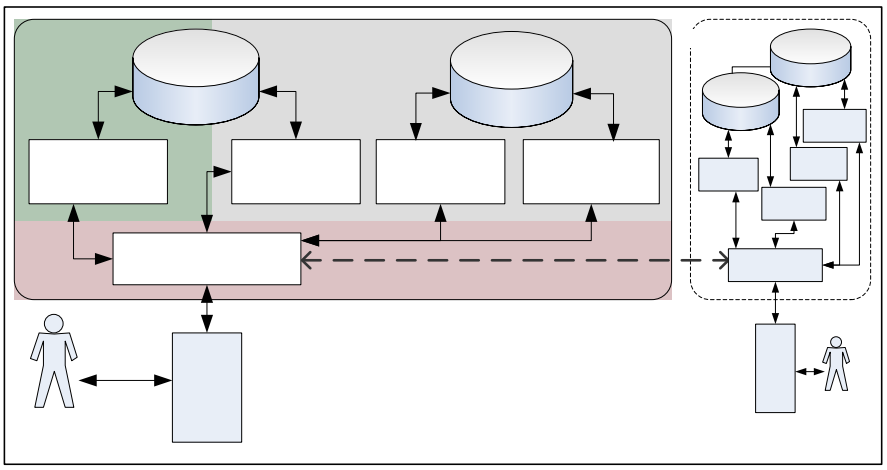

Fig. 4. Two instantiations of Invite ens for bilateral e-negotiation support

Implementation and testing. We designed protocols for several negotiation types and the components that implement all required negotiation activities for these negotiations.

Invite uses the protocol to generate an ENS supporting particular negotiation. Because of the separation of the view component and the protocol, it is possible to construct the same

mechanism (model and controller) for different interfaces. Example of six such layouts is presented in Figure 5; each layout has been designed for a different type of negotiations (defined by a combination of characteristics given in Table 3).

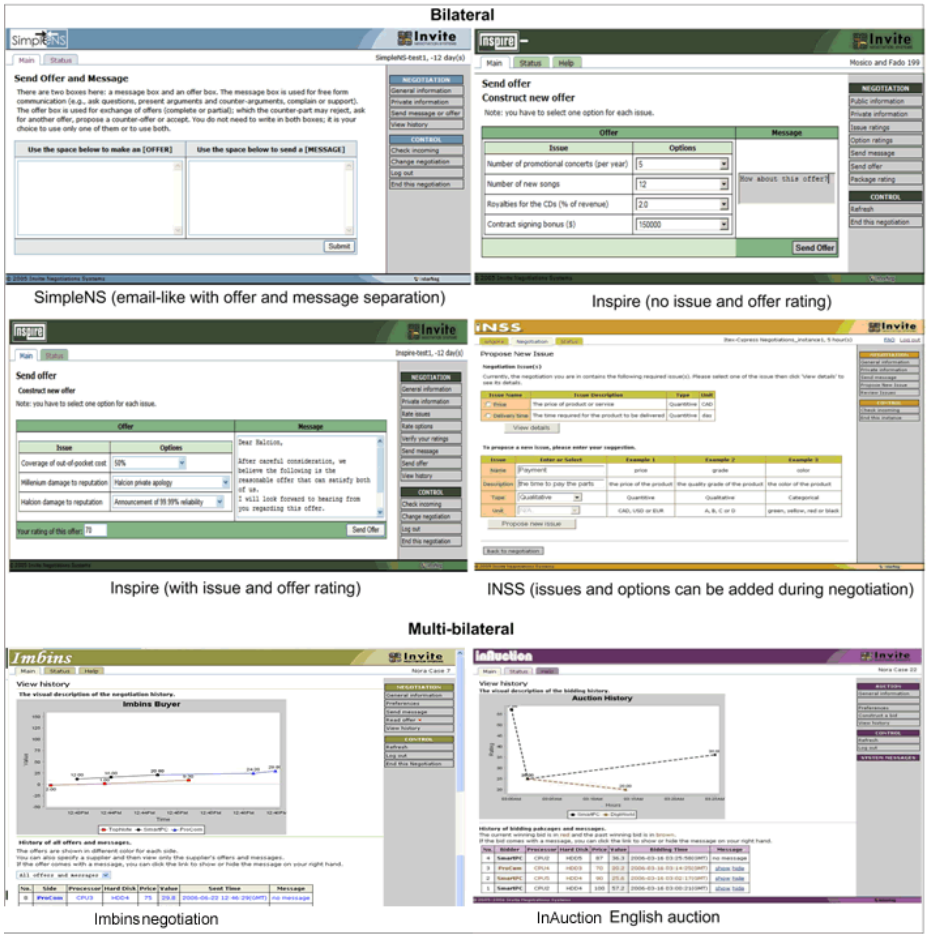

Fig. 5. Example screenshots of six Invite enss generated by different protocols

Observe that the interfaces are similar. A similar-looking interface layout is used for every system in order to minimize the impact of the distinct interface features on the negotiators' performance and to compare the use and usefulness of each system and its role in the negotiation process.

All screen shots presented in Figure 5 come from a different ENS. The first four belong to systems supporting bilateral negotiations (SimpleNS, Inspire⁻, Inspire and INSS) and the last two to multi-bilateral negotiations (Imbins and InAction).

We conducted ten sessions of laboratory experiments using the Inspire⁻ and Inspire systems implemented by the Invite platform. The total number of participants was 114, mostly graduate and undergraduate students majoring in business and engineering. Each session

allowed for the maximum one hour of negotiation. No training on how to use the system was offered before the start of negotiation. In all negotiations, we observed active exchange of offers and messages.

Out of 57 bilateral negotiations, in 41 an agreement was reached. No difficulties in using the system were reported by users. Most questions raised by the participants during the negotiation session were related to the negotiation case and the preference elicitation model. We believe these results indicate that the framework not only allows to reduce context dependency but also to develop ENSs with a high degree of usability.

Based on the available components implemented in the Inspire system, two other systems were designed for the comparative studies of auction and negotiation systems. One of them, Imbins, (InterNeg multi-bilateral integrative negotiation system), extends the current bilateral negotiation to the multi-bilateral cases. The second system is InAuction (InterNeg auction system), which supports a limited-information multi-attribute English auction. These two systems are built with similar user interfaces, functions, and architecture (see Figure 5).

Preliminary results of the comparison of multi-bilateral negotiation and auction mechanisms showed no significant effect of the mechanism on economic measure (i.e., seller's utility, buyer's utility and social welfare).[3] The empirical tests of subjective measures indicate that agents in e-market exchange are driven by the goal of utility maximization; the utility that the agents gained has positive impact on their satisfaction. Utility has significant and positive effect on satisfaction of auction and negotiation winners.

Among the non-winners, it was found that the mechanism has a significant effect on agents' satisfaction with outcome. At a 10% level, mechanisms have significant effect on agents' satisfaction with self-performance. Auction leads to higher levels of agents' satisfaction with outcome and self-performance. A possible reason is that auction provides fast and accurate feedback to agents during the exchange, while negotiation provides such information only in the predefined exchange task.

3.2 eNAs e-negotiation agency

The e-negotiation agency (eNAs) offers a platform for autonomous agent-based negotiation in e-markets. It provides a suite of negotiation agents that on behalf of the users, can engage in automated negotiations with others over the Internet, using different negotiation mechanisms suited for various user preferences, exchange types and applications.

[3] See B. Yu, "Negotiations or Auctions: Experimental comparison of two e-market mechanisms," M.Sc. Thesis, J. Molson School of Business, Concordia University, May 2007.

Foundations. The enas suite provides a variety of negotiation mechanisms involving different negotiation protocols, objects and decision mechanisms that support autonomous agent-based negotiations ranging from basic single-attribute bi-lateral negotiations to complex multi-attribute multi-lateral negotiations.

The negotiation protocols included in enas support the following four exchanges:

1. One-to-one negotiation following a protocol of iterative exchange of proposals and counter-proposals [29-31, 40]
2. One-to-many negotiation following the iterative contract net protocol (incp [14]) and its extension supporting two way exchange of offers (i.e. the initiator, in addition to calling for proposals cfp and receiving offers from the participating agents as per incp, can also make proposals to other agents) [10, 11].
3. Many-one-to-one negotiations based on a number of concurrent coordinated one-to-one negotiations, where the negotiation agents on one side are coordinated by a coordination agent that decomposes the overall request, distributes the requests to the individual negotiation agents, evaluates the individual results after each negotiation cycle and issues new instructions accordingly (e.g. redistribution of individual requests) [10, 11, 29].
4. Many-one-to-many negotiations where the coordinator agent coordinates a number of agents (the initiators in incp) involved in one-to-many negotiations [11].

The eNAs agents are able to negotiate about complex objects including (1) multi-attribute objects of negotiation (i.e. multi-issue negotiation) and (2) the case in which the attributes are constrained by individual (e.g. min, max) and relational constraints (e.g. relation between price and volume) within an object and between multiple objects (e.g. for bundling, aggregations) [31, 41];

The decision-making capabilities of the eNAs agents are provided through different decision mechanisms as follows:

- Constraint-based [40] and fuzzy constraint-based [29] reasoning;
- Multi-attribute utility theory with a number of heuristic negotiation strategies [10];
- Qualitative decision-making based on possibility theory supported by predictive on-line and off-line opponent modeling [8]; and
- Case-based reasoning for negotiation partners selection, and supporting coordination in multi-party negotiations [7].

General architecture. The enas platform consists of a number of negotiation agents that can negotiate with one or more agents over the Internet as depicted in Figure 6.

The eNAs agents share information about negotiation objects and conduct negotiation through information exchange following a common negotiation protocol (which is typically predefined for a specific e-market application). Each agent acts on behalf of its user who instructs the agent about the requirements (i.e. preferences, utilities, constraints, reservation values, deadline, etc.) and chooses a negotiation strategy for the agent to use during negotiation.

The eNA agents support a number of negotiation strategies and can also select a strategy that is appropriate for the negotiation protocol, object and requirements. The main components of

each agent are described in more details in the next subsection.

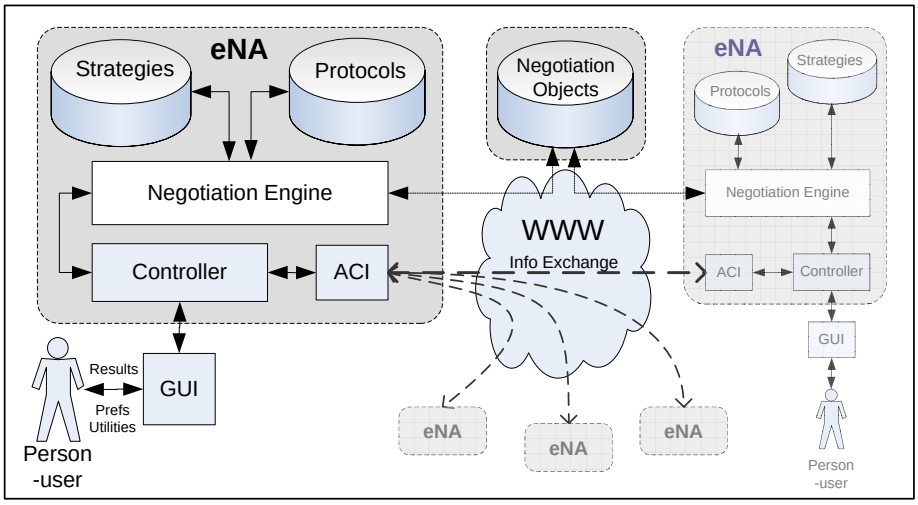

Fig. 6. Overview of the e-Negotiation Agency

Main components. Each enas agent consists of a number of components (see Figure 6) as follows:

- Negotiation Engine is the main component providing an agent with the decision-making capabilities required for evaluation of proposals, acceptance/rejection of the proposals and generation of counter-proposals guided by selected negotiation strategies and protocols via Controller.
- Controller is responsible for the overall operations of an agent including control of the negotiation process according to selected negotiation strategies and protocols, communication with the external world (the user and other agents) and coordination of all the components of the agents
- Libraries of Strategies and Protocols consist of negotiation strategies and protocols, respectively, available to each agent.
- Gui and aci are the user and agent communication interfaces for information exchange with the user and other agents, respectively. Gui is used to gather user input (e.g. requests, preferences, utilities etc), and display the progress and results of the negotiation. aci uses a standardized agent communication language [14] to exchange information with other agents during negotiation.

Areas of application. Different versions of enas have been developed and demonstrated for a number of real-world application scenarios; three examples are given in Figure 7.

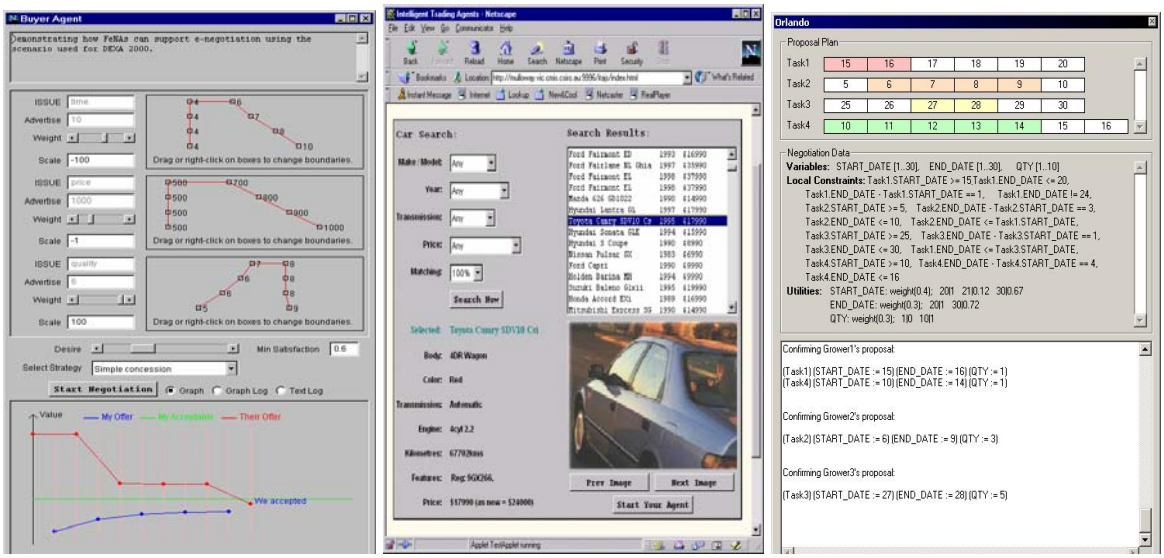

Fig. 7. enas GUI sample screenshots: (a) user preferences and progress of a printing service negotiation, (b) car trading agency, and (c) supply chain scheduling

The three versions illustrated in Figure 7 are:

1. e-Commerce trading demonstrated for car trading and a negotiation of printing services [40]
2. Supply chain coordination through negotiated scheduling of inter-organizational supply networks demonstrated for the wine production supply chain
3. Web Service compositions involving coordinated negotiation and renegotiation of quality-of-service (QoS) from different service providers [10, 11, 36] demonstrated for compound service provision of the telecommunication, logistics and multimedia services and Internet services

A more detailed description of different aspects of the eNAs agency, negotiation mechanisms and applications can be found in the literature referenced in this section.

3.3 meet2trade auction platform

Foundations. The meet2trade auction platform is a generic market server that allows to design and combine various auction formats in real time. meet2trade not only constitutes a pure auction platform but also a software suite for supporting the entire auction design process from the first idea over the roll-out to the operation. Most steps of the auction design process are fully or at least partially automated [51]. The meet2trade system supports the design of:

- One-to-many price based auctions following either a recurring, iterative bidding process or a one shot process.
- Many-to-many price based auctions
- One-to-many multi-attribute auctions allowing for both iterative and one-shot processes.
- Many-to-many multi-attribute continuous auctions
- Spontaneously varying auction formats

The meet2trade agents can be used to act on those configured auctions, where rather simple strategies (e.g. Zero-Intelligence-Plus and Gjerstad-Dickhaut agents) are already implemented. Furthermore, the AMASE system allows a rather convenient API to expand the strategies.

General architecture. As aforementioned, the meet2trade architecture was constructed in order to host various auction formats at a time and to integrate them in real time. From a system development's point of view this requires:

- A flexible auction management component that allows for both, double sided and single-sided auctions;
- A dynamic offer management that allows the user to submit one single offer to a combination of auctions simultaneously. This combination ranges from simple sequences of auctions the offers passes through to a complex structure with parallel and sequential auction segments; and
- An adaptive user interface that has the ability to represent different views for specific auction formats

To allow these three requirements, the meet2trade generic market server follows client-server architecture. As Figure 8 illustrates, the meet2trade server consists of the 3-tier architecture: (i) communication modules to manage communication between the trading server and the trading client, (ii) storage functionalities, which are responsible for the log in of data produced in the trading, and (iii) auction run-time environment (ARTE), which represents the Shaman market component.

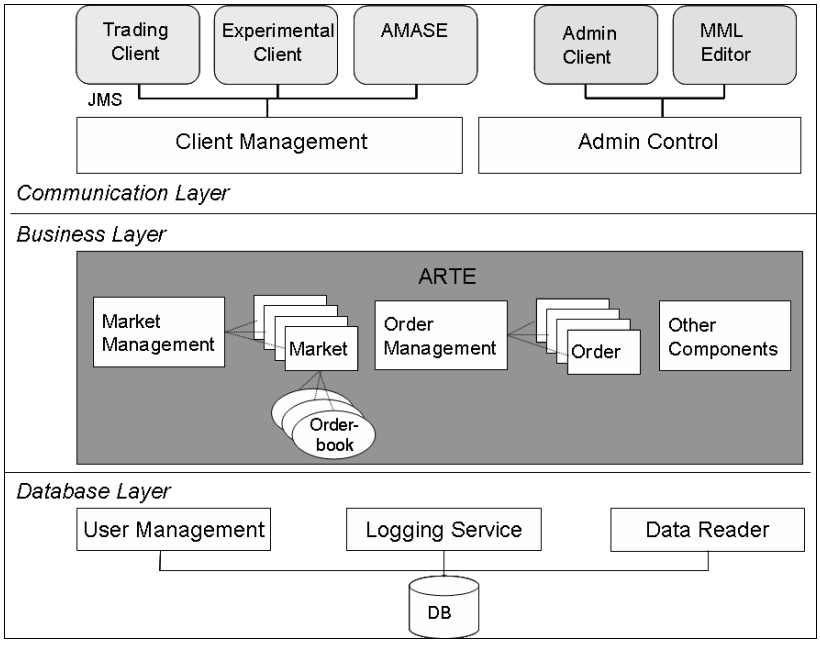

Fig. 8. Overview of the meet2trade architecture [50]

The functions of each layer and its main components illustrated in Figure 8 are:

- The communication layer prepares the data for client presentation, provides the communication, and, moreover, administers all connected clients. The connected clients comprise mainly of: (i) the generic trading client, which allows the traders to submit offers to any kind of auction formats, (ii) the amase component, which constitutes the Shaman meet2trade agents, (iii) and the experimental client mes, which is responsible for economic testing allowing control over the auctions for laboratory experiments.
- The business layer consists of the core market environment called arte (Auction Runtime Environment) on which all auctions are running and all offers are processed.
- The database layer encapsulates the database access and provides the logging of all trading data as well as the management of the user and depot data.

To achieve a high degree of platform independence, the meet2trade client was developed as Java application. For communication between client and server the standard Java Message Service (JMS) is used. This enables distributed reliable and asynchronous communication. The messages are encoded in XML schemata, and therefore, provide a high degree of readability and re-usability. In order to keep the client generic, it is necessary to adapt the components to the market's requirements. Consequently, the design of the GUI is defined in XML messages according to the requirements of the specific auction format and the available offers. The GUI description messages are provided by the meet2trade core named ARTE.

Main components. There are five components of meet2trade:

- Market component – arte
  The design of market mechanisms is based on the parameterization approach – i.e. any auction can be described by a set of parameters representing their rules. For this purpose an XML-based language to define auctions – namely, the market modeling language MML – has been developed. The mml schema thereby represents the parameter structure, while the mml instances define the concrete auction formats. Arte (auction run-time environment) instantiates any conceivable market mechanism, which is part of the design space. Hence, arte is at the core of the meet2trade tool suite. Arte is fed by a configuration-editor, which allows the generation of a MML instance of an auction [34].
- User interface – Adaptive Client (ac)
  The ac configures a trading gui on the basis of the mml. This automatically generated gui can be adapted by the users according to their needs. The adaptive client offers the following advantages (1) the client is an easily configurable drag & drop- click. Current client configurations can be stored in a xml-format (2) the client behavior can be controlled and monitored by the server. This is especially important when conducting experiments with the built in experimental system MES. With ARTE and the adaptive client at hand, the technical problems of the market engineering approach are addressed.
- Decision Support – kads
  Another component within meet2trade is the dss kads (Knowledge-Based Auction Design Support), which prescribes what the market mechanism should be like in order to attain the desired goal. The decision support system kads stores and makes economic design knowledge accessible to the user of the workbench. In essence, economic design knowledge is captured by simple rules, where the antecedents of the rules are certain indicators of the environment as well as the market mechanisms and the consequents are

the impact of the market mechanisms on the market outcome. Taking into consideration that the design knowledge is inherently incomplete, the prescriptions of kads are in many cases vague and also contradictory. This problem is imminent to market engineering and cannot be removed [37].

– Agent – amase
  In order to add more confidence to the kads recommendation, and to support early prototypes of the electronic market service, the CAME toolkit integrates a simulation tool to evaluate the market mechanisms in certain environments. Amase is an agent-based simulation environment, which allows for automated testing of market mechanisms. Simple test scenarios can be produced on-the-fly, while more complex scenarios require some coding of the agent behavior [12]. Amase renders predictions about how the market mechanisms will perform. The technique of simulations allows valid predictions even about sophisticated market mechanisms, where the analytical determination of equilibrium outcomes is too complex.
– Experimental System – mes
  In order to examine the new auctions which have been designed using the MML, the mes was added to the meet2trade software suite. The main objective is to conduct experiments on the original system instead of having to design, simplify and implement them using standard experimental software like zTree. On the one hand this approach facilitates experimental studies because the market has to be modeled only once within meet2trade, and on the other hand it uses the standard meet2trade-client with the same look-and-feel of the normal trading client instead of a simple graphical user interface of the standard software [28].

Areas of application: The meet2trade system has been used for several real-world application scenarios comprising:

– Financial trading with innovative order types and auction formats [32]
– Consumer-to-consumer auctions [49]
– Trading computer resources
– Trading emission allowances.

The system allows for setting up a number of different auction mechanisms, both single-issue (e.g., price only auctions), multi-issue and combinatorial auctions. Multi-attribute English auction is illustrated in Figure 9.

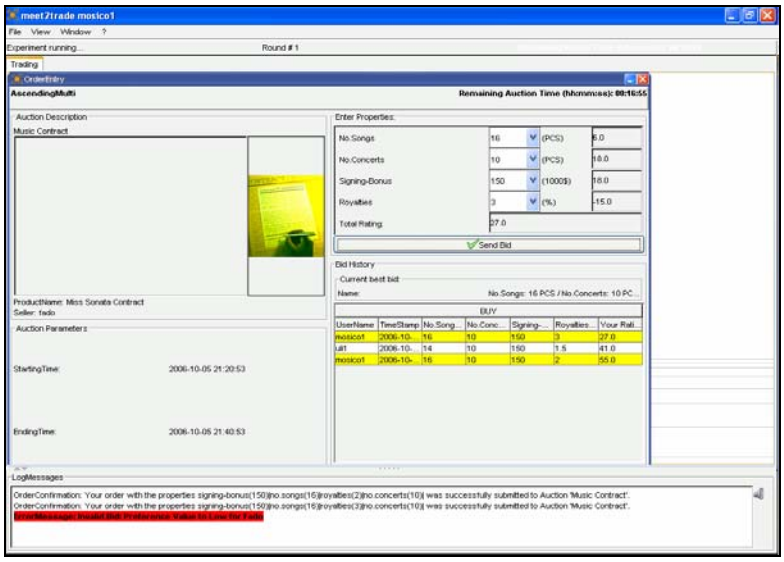

Fig. 9. Screenshot of meet2trade (multi-attribute English auction)

A comprehensive overview of the meet2trade system is given in [52].

3.4 GoGo group-buying platform

Group-buying on the Internet is defined as a computer-based mercantile exchange mechanism that allows consumers to take advantage of volume discounts by shopping together, which can be coordinated by technological capabilities that support many new approaches to price curve specification and to coalition formation [23].

Foundations. There are three key participants in the group-buying e-market:
1. The initiator is the one who initiate the group-buying transaction, for example, a consumer, a retailer, or a manufacturer;
2. The consumer who is interested in or participates in the group-buying.; and
3. The supplier who sells the products to the initiator.

Every user needs a variety of services in order to be satisfied and able to engage in effective and efficient group buying. These services are briefly described below.

Initiator. A group-buying model includes the decision-making of the target product, price curve (relations between quantity or value and unit price) and deadline. For an initiator, the available group-buying protocols could be designed based on five dimensions: (1) the initiator's characteristics; the (2) extent of product variety; (3) the number of involved sellers; (4) the bargaining power base, and (5) the conditionality of the sales offers [33]. The initiator may need the following functions from agents and decision support tools from enss:
1. Agents:
   - Initiate a group-buying transaction which may include product, price curve, deadline, qualifications of participants, etc.
   - Monitor current status of the ongoing group-buying.

- Notify the change of transaction status of the ongoing group-buying based on conditions set up by the initiator or the deadline of the ongoing transaction.
- Promote the ongoing group-buying.
- Evaluate the suppliers, participants, and group-buying model performance.
- Search the past transactions, suppliers and former customers by products, time period, number of participants, suppliers, group-buying model, transaction result, evaluation, etc.

2. Decision support tools:
   - Price curve design
   - Product selection
   - Target participant selection
   - Supplier selection
   - Group-buying protocol selection
   - Marketing strategies
   - Negotiation vs. auction decision

Consumer. A consumer may participate in a group-buying transaction or express her desire to initiate group-buying by herself or ask someone else to initiate group-buying for a desired product. Therefore, a consumer may need the following functions from agents and support from enss:

1. Agents:
   - Join in a group-buying transaction
   - Monitor the current status of the ongoing group-buying which the consumer has joined or may join
   - Notify any change of transaction status of the ongoing group-buying based on conditions set up by the consumer or the deadline of the ongoing transaction.
   - Evaluate the initiators and the group-buying performance.
   - Call for product request
   - Search the past transactions, initiators, time period, number of participants, group-buying model, transaction result, evaluation, etc.
2. Decision support tools:
   - Negotiation vs. auction vs. group-buying decision
   - Ongoing group-buying selection
   - Initiator selection
   - Timing of joining in a group-buying transaction
   - Group-buying protocol selection

Supplier. A supplier in a group-buying market can promote herself in order to get attention of consumers or initiators. A supplier also has to decide which transaction model, negotiation or bidding will get better result in order to place group-buying order. Proposing a price curve is another important concern. Overall, a supplier may need the following functions from agents and support from enss:

1. Agents:
   - Bid for a group-buying transaction
   - Negotiate with initiators for group-buying transactions

- Monitor the current status of ongoing group-buying transactions
- Notify any change of transaction status of the ongoing group-buying based on conditions set up by the consumer or the deadline of the ongoing transaction.
- Evaluate the initiators and group-buying performance
- Search the past transactions, initiators, time period, number of participants, group-buying model, transaction result, evaluation, etc.

2. Decision support tools:
   - Promoting strategies
   - Negotiation vs. bidding decision for a group-buying transaction
   - Group-buying protocol selection
   - Price curve proposal
   - Selection of appropriate products for group-buying transaction
   - Initiator selection

General architecture**.** The architecture of the GoGo system is illustrated in Figure 9.

For each member, the system provides transaction protocols, transaction database, agent base, decision support systems, controllers and interface. Through the interface, the person-user can interact with the system. The controller will manage the whole interaction process which may invoke the agents and/or decision supports. When the interactions between different members or between GoGo and other platforms such as Invite, eNAs and meet2trade are required, the controller will invoke the agent communication interface to act on behalf of the user. The details of each component are described in the next subsection.

Main components. Based on the foundations and architecture, we can see (Figure 7) that there are seven key components for every type of members of the group-buying e-market. They are transaction protocols, transaction database, agent base, decision support systems, controllers, agent communication interface and interface.

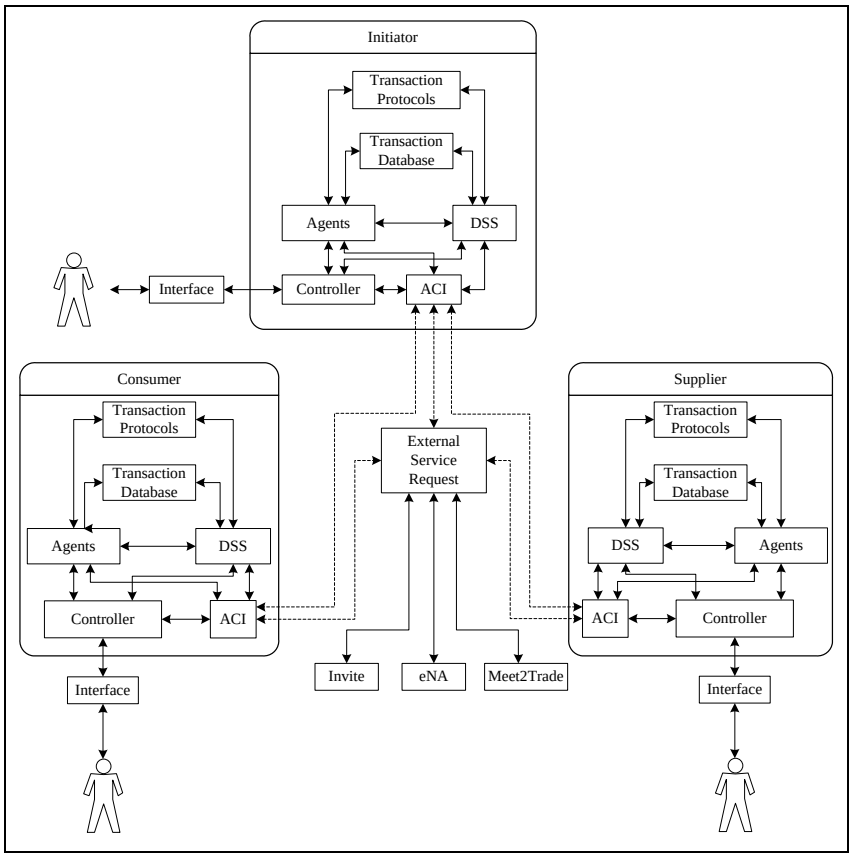

Fig. 10. Overview of the GoGo architecture

The purpose and function of each component are:

1. Interface serves to exchange information between a user and the group-buying system. It gathers the user's input and passes the results from the controller to the user
2. Agent communication interface (aci) is in charge of all interactions with other GoGo users or with other market platforms such as Invite, eNAs and meet2trade.
3. Controller is responsible for the overall operations required for the interactions between the system and its users. It may invoke aci in order to interact with the other type of users or the other platforms
4. Agents base includes all kinds of agents which can provide services required by the users. As discussed in Section 3.4.1, the system provides each type of members with different kinds of agents. Ideally, the agent can be invoked by person-user or even do something autonomously on behalf of the person-user
5. Dss provides all kinds of decision support. It can be invoked by the controller directly, by agents or by aci. Decision support tools required by the users have been listed in Section 3.4.1.
6. Transaction protocols include transaction models. Different type of users may need different decision supports and the protocols associate users' needs with the support tools that may meet these needs
7. Transaction database keeps data of all transactions. It provides the information requested by agents and dsss.

Implementation and testing. GoGo has been used in several experiments of group-buying that involved graduate and undergraduate students from a Taiwanese university. The current version of the system has interface in Chinese; an example given in Figure 10 shows the screenshot of initiating a group-buying activity; the parameters listed in the figure are required in order of setting up such a group.

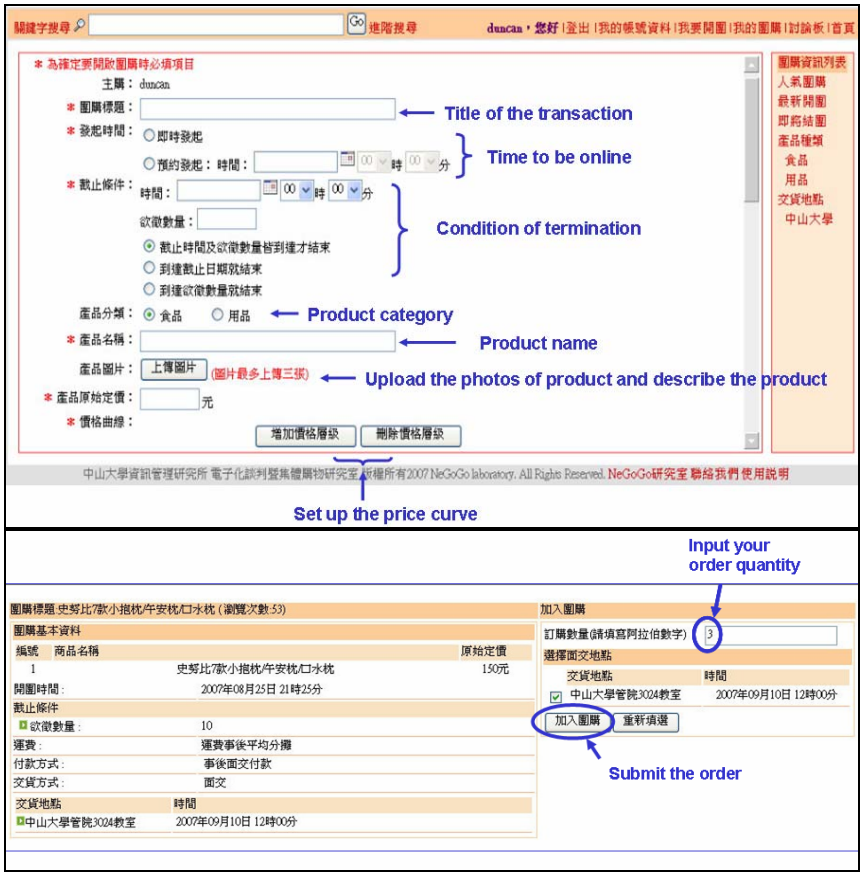

Fig. 11. Two GoGo screenshots: (a) setting up a group-buying; and (b) joining an existing group

Areas of application. There are many types of group-buying businesses in which such system as GoGo can be used. They include such existing businesses as PetroSilicon (www.petrosilicon.com), a B2B business that supports online demand aggregation for crude oil and petroleum products; and www.52Marketplace.com, a Netherlands-based company that created the Open Source Auction Network. The Network has the purpose of bringing together auction-based suppliers with groups of consumers. The company's Web site explains that it helps buyers looking for the same products to form buyer groups and enables sellers to submit their best price. It claims that the support they offer results in the discounts for buyers and more sales for sellers. Another example in the United States is OnlineChoice (onlinechoice.com), which provides a platform for group-buying for telephone long distance services, home heating services, prescriptions for pharmaceuticals, various kinds of personal insurance, and other services. The website focuses more on pooling buyers' demand and

then encourages suppliers to offer discount prices. LetsBuyIt.com is another example. It provides the consumers with three different group-buying purchase price choices: the current price (a buy-it-now option), the closing price (when the co-buying auction closes), and the best price (which is the lowest stated price in the co-buying auction).

4. Shaman framework and functions

The Shaman project builds on the four projects described in the previous section and its purpose is to create heterogeneous environment in which people and software agents can share and use resources.

4.1 Framework

The proposed framework comprises four unified software platforms (Invite, meet2trade, eNAs and GoGo) each with similar but also distinctively different functionalities. The overview of the four platforms comprising the Shaman framework is given in Figure 12.

In each of the software platform we can distinguish the following three main functions which are accessible to the platform users, both people and software agents:

1. The market function which allows the users of the platform to interact, engage in exchange of information and conduct transactions.
2. The agent function which allows the users to interact with a software agent, formulate requests to these agents and activate them to undertake specific actions.
3. The dss function, which allows users and software agents to select and use tools helping them to construct and solve models, analyze solutions, etc.

In each platform these three functions have been differently implemented and vary in the degree of their specificity and scope. For example, in Invite we consider an agent whose sole purpose is to provide advice to the negotiators and explain to them the purpose of the Invite tools and protocols. In meet2trade software agents have been used as bidders who compete with persons on the auction markets. eNAs provides software agents that autonomously negotiate with other software agents on behalf of their users. GoGo software agents monitor the market and group members, inform their principals and engage in group-buying activities on their behalf.

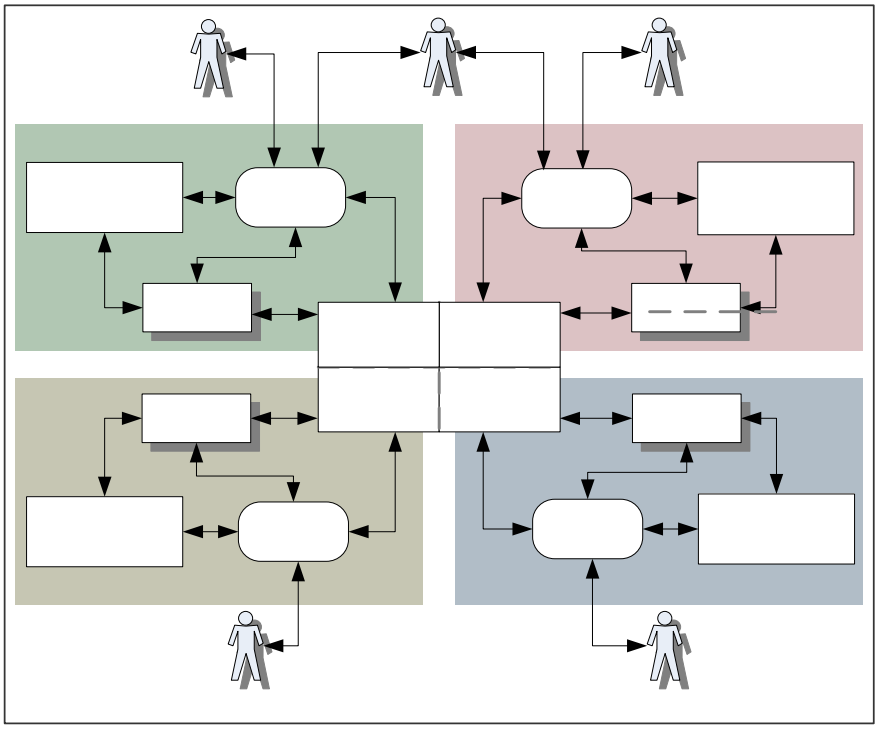

Fig. 12. Shaman's framework

The DSS function is not necessarily separated in every platform; however each platform has components which provide its own tools and aids with decision support functionality. For example, all platforms have tools in which different preference elicitation and utility construction models are implemented. Visualization, an important dss functionality, is available in Invite where it is used to depict the negotiation progress in each user's utility space. Similar visualization is provided by eNAs to show the users the progress of negotiation between software agents.

Invite
E-negotiatio
market

Each software platform, together with other programs, utilities, dbms and os, is a separate computing environment. In order to establish communication between the platforms we either need new components or add new capabilities to the existing components. We selected the former in the form of component Interface/Coordinate, so that no major changes are required in the existing components.

The Interface/Coordinate component has the capability of translating and interpreting the requests which are received from outside of its platform. It can interpret requests made by other components belonging to its platform in order to pass it to other platforms. Therefore, it needs to know the other platforms functionalities and the capabilities of their Interface/Coordinate components.

The platform's DSS "knows" the functionalities and capabilities of its own platform and therefore it can support users and agents. The Interface/Coordinate component has "knowledge" of other platforms functionalities and capabilities that allow it to coordinate interactions

between the Agents and DSS which belongs to its platform with the other platforms' Agents and DSSs. It performs decision aiding functions in the sense that it informs both the external entities about the capabilities of "its" Agents and DSS components and the internal entities about the foreign Agents and DSS components. This knowledge also allows the component to request foreign (i.e., belonging to other platforms) agents and DSSs to perform activities that its local agents and DSS cannot perform. It also allows foreign entities to use its own platform functions. Thus the four Interface/Coordinate components depicted in Figure 12 act as a "harness" that brings together the four platforms making them interoperable and allowing users of one platform access functions and capabilities of every other platform. Their role goes beyond an application programming interface (API) which provides information software can interact; the components are able to use and coordinate other software, and informing its users about the capabilities of software with which the Interface/Coordinate components do not directly interact.

4.2 The use of Shaman

To illustrate the role and functions of Shaman and the use of its Interface component, the interactions of the users of one its platforms who wish to use services provided by another platform(s) are briefly discussed in the section.

Invite negotiator bids on meet2trade. Mary, a user of Inspire, an Invite e-negotiation system, wants to participate in an auction on the meet2trade platform. One possibility is that Mary joins meet2trade and uses its DSS. This would require that Mary learn the meet2trade DSS; she would rather use Invite DSS with which she is very familiar. In particular, Mary finds the preference elicitation scheme implemented in Inspire easy to use and she would also like to see the auction progress using Inspire graphs.

The DSS in the Invite platform allows Mary to select a meet2trade auction rather than a local e-negotiation process and specify if she wants to set up her own auction or join an existing auction as a bidder. Mary wants to join an existing auction and the Invite DSS requests that the Interface/Coordinate component obtain the list of on-going and open auctions from meet2trade.

The Invite Interface communicates with the meet2trade Interface and specifies Mary's request. The meet2trade Interface/Coordinate passes this request to the Controller, which verifies the auctions availability and passes the list of available auctions and their rules to Mary via the two Interface/Coordinate components. Mary selects an auction for a one week vacation in Monaco.

Mary cannot negotiate directly on the auction market; she can do this via one of the meet2trade Agents or the DSS. If she wants to make bids by herself she can use the same DSS tool as the meet2trade users.

eNAs agents bid on meet2trade for Invite negotiator. Mary, an Invite user, has posted a few bids on the auction hosted at meet2trade for a one week vacation in Monaco. The deadline for this auction is in 3 days and Mary will not be able to continue bidding because she has to undergo a minor surgery. She really does not want to quit so she decides to delegate the bidding to a software agent. Although she could use a meet2trade agent, she prefers to use an external agent.

Mary has access to the list of eNAs agents, two of them are specialized in auction bidding. She selects both and the Invite DSS connects her via Interface/Coordinate with the eNAs platform. This DSS passes on her request to the eNAs agent; this agent uses the eNAs Interface/Coordinate to participate on her behalf in the meet2trade auction. After finishing the auction the meet2trade advises her about the outcome of the auction and eNAs DSS explains the progress and the results of bidding made on her behalf.

GoGo user evaluates the available auctions on meet2trade. Winnie, a user of GoGo, plans to buy an iPhone. The dss in GoGo allows her to buy it from either an auction market or a group-buying market. Using an auction she may pay higher price but be able to get iPhone earlier while using group-buying she may have to wait longer but pay less. Winnie decides to check if there is an iPhone being auctioned or a buying group being formed.

The GoGo DSS requests the Interface component obtain the list of ongoing auctions from meet2trade. The meet2trade interface passes this request to the Controller to verify the availability of iPhone auction. The GoGo DSS also checks if there is any ongoing group-buying. Then the dss aggregates the obtained information and displays it to Winnie.

eNAs agents negotiate on behalf of GoGo users. After evaluating the ongoing auctions on meet2trade and group-buying instances on GoGo, Winnie considers initiating a group-buying by herself because after she talked to a retailer she may get a better price thanks to the collaborative bargaining power. Because she is not confident about her own negotiation ability, she decides to get help from an enas agent. The GoGo Interface/Coordinate passes her request to enas. Upon receiving the list of available enas agents, the GoGo dss helps Winnie choose an agent.

The selected eNAs agent negotiates with the retailer on behalf of Winnie in order to get better price curve depending on the number of recruited buyers. After finishing the negotiation, the eNAs negotiation agent informs the result to Winnie and eNAs DSS explains to her the progress and the results of negotiation made on her behalf.

4.3 Local DSS

Every platform has its local DSS. The main purpose of local DSSs is to provide tools and aids to its users: people and software agents. These tools and aids can also be accessed, via Interface/Coordinate by the users of other platforms.

A person's environment comprises individuals and groups of identical or similar social positions and social roles. Human environment is also the culture that the person was educated and lives in, the people and institutions with whom the person interacts, and the

norms and laws in which this person operates.

The agent's environment consists of: (1) the set of all entities which provide the agent with information and affect its ability to act on this information; (2) the norms and rules which the agent has to take into account when seeking information and undertaking actions; and (3) the infrastructure which allows the agent to communicate and act.

The importance of the environment is due to the heterogeneous nature of the market participants. They are situated in very different environments; therefore they need to establish concrete means and channels for communication (typically via a user interface). However, both human and software agents may also communicate using different channels. This is may be the case of a badly designed, malicious software agent but also due to an error or lack of understanding of the "rules of the game" of the human agent. If some communication bypasses local DSSs, then these systems base their recommendation on partial information and may provide inaccurate advice or assessment.

5. Conclusions

A DSS-centric software environment for human-agent e-markets called Shaman is proposed in this paper. It aims at supporting the construction and operation of heterogeneous systems to enable business interactions such as actions and negotiations between software and human agents across those systems. The DSS are used to provide integration and coordination between the participating systems and their users. The proposed conceptual framework of Shaman has initially been developed and illustrated on the basis of the four distinct systems: Invite e-negotiation system, eNAs negotiation agency suite, meet2trade auction platform and GoGo group-buying system. It offers a new type of e-market interoperability and business interactions across different systems forming Shaman environment.

It should be noted that there are a number of issues pertaining to the development and use of such a complex and distributed software environment as Shaman that need careful consideration for its successful deployment in real-world applications. It includes different parameters for the configuration of multi-agent heterogeneous systems and a unified set of design principles for their construction. For example by varying the configuration, the agents engaged in the activities in order to obtain different types of products and/or services, and employing different types of exchange mechanisms, can differently affect the socio-economic processes such as the users' satisfaction and the usefulness of the system and its components, the relationships between users' characteristics and the efficacy of the software agents, market mechanisms and exchange processes. The future work on Shaman includes studying and understanding these processes in order to ensure the effectiveness and efficiency of the human-agent markets enabled by Shaman.

While the initial focus is on e-business and e-commerce, Shaman is also applicable to non-commercial interactions and activities. One example of such activities involves support for participatory democracy in which many thousands of citizens interact in order to arrive at a better understanding of the problem (e.g., municipality budget problem). This may involve:

matchmaking, organization of effective communication among thousands of people, and interactions among agents representing various interests groups across different systems integrated with Shaman.

References

Alter, S., Decision Support Systems: Current Practice and Continuing Challenges. 1980, Reading, Mass: Addison-Wesley.

Angehrn, A.A. and H.J. Luthi, Intelligent Decision Support Systems: A Visual Interactive Approach. Interfaces, 1990. 20(6): p. 17-28.

Arnott, D. and G. Pervan, A critical analysis of decision support systems research. Journal of Information Technology Theory and Applications, 2005. 20: p. 67-87.

Bailey, J.P. and Y. Bakos, An exploratory study of the emerging role of electronic intermediaries. International Journal of Electronic Commerce, 1997. 1(3): p. 7-20.

Balasubramaniam, R. and M. Kannan. Integrating Group Decision and Negotiation Support Systems with Work Processes. in Proc. of the 34th Hawaii International Conference on System Sciences. 2001. Hawaii, IEEE.

Bazerman, M., Judgment in Managerial Decision Making. 4 ed. 1998, New York: Wiley.

Brzostowski, J. and R. Kowalczyk. On Possibilistic Case-based Reasoning for Selecting Partners for Multi-attribute Agent Negotiation. in The 4th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2005). 2005. Utrecht, Netherlands.

Brzostowski, J. and R. Kowalczyk. Adaptive negotiation with on-line prediction of opponent behaviour in agent-based negotiations. in The 2006 IEEE / WIC / ACM International Conference on Intelligent Agent Technology. 2006. Hong-Kong, China.

Chavez, A., D. Dreilinger, R.H. Guttmann, and P. Maes. A Real-Life Experiment in Creating an Agent Marketplace. in Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97). 1997. London.

Chhetri, M.B., J. Lin, S. Goh, J. Yan, J.Y. Zhang, and R. Kowalczyk. An Architecture for the Agent-based Coordinated Negotiation of Service Level Agreements for Web Service Compositions. in Proceedings of the 2006 Australian Software Engineering Conference (ASWEC '06). 2006. Sydney, Australia: IEEE Computer Society Press.

Chhetri, M.B., J.Y. Zhang, J. Brzotowski, S. Goh, J. Lin, B. Wu, and R. Kowalczyk. Experimentation with Three Different Approaches of Agent-based Negotiation. in Proceedings of the Workshop on Service-Oriented Computing and Agent-based Engineering (SOCABE '06). 2006. Hakodate, Japan.

Czernohous, C. Simulation for Evaluating Electronic Markets - An Agent-based Environment. in IEEE Proceedings of the 2005 International Symposium on Applications and the Internet (SAINT 2005). 2005.

Dickinson, I., Human-Agent Communication. 1998, Bristol: HP Laboratories.

FIPA, Foundation for Intelligent Physical Agents, http://www.fipa.org/.

Franklin, S. and A. Graesser. Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents. in Intelligent Agents III: Agent Theories, Architectures, and Languages, ECAI'96 Workshop Proceedings. 1997. Budapest: Springer.

Gorry, G.A. and M.S.S. Morton, A Framework for Management Information Systems. Sloan Management Review, 1971. 13(1): p. 1-22.

Gulliver, P.H., Disputes and Negotiations: A Cross-Cultural Perspective. 1979, Orlando, FL: Academic Press.

Guttman, R.H., A.G. Moukas, and P. Maes, Agent-mediated ecommerce. A survey. Knowledge Engineering Review, 1998. 13(3): p. 147-159.

Haeckel, S. and R. Nolan, Managing by Wire. Harvard Business Review, 1993. Sept.-Oct.: p. 122-132.

Jennings, N.R. and M.J. Wooldridge, Applications of Intelligent Agents, in Agent Technology: foundations, applications and markets, N.R. Jennings and M.J. Wooldridge, Editors. 1998, Springer: Berlin et al. p. 10ff.

Jennings, N.R., An Agent-Based Approach for Building Complex Software Systems. Communications of the ACM, 2001. 44(4): p. 35-41.

Kahneman, D., New Challenges to the Rationality Assumption. Journal of Institutional and Theoretical Economics, 1994. 150(1): p. 18-36.

Kauffman, R.J. and B. Wang, New Buyers' Arrival under Dynamic Pricing Market Microstructure: The Case of Group-Buying Discounts on the Internet. Journal of Management Information Systems, 2001. 18(2): p. 157-188.

Kersten, G.E., Support for Group Decisions and Negotiations. An Overview, in Multicriteria Analysis, J. Climaco, Editor. 1997, Springer Verlag: Heilderberg. p. 332-346.

Kersten, G.E., S.E. Strecker, and K.P. Law. Protocols for Electronic Negotiation Systems: Theoretical Foundations and Design Issues. in Fifth International Conference EC Web 2004. 2004. Zaragoza, Spain.

Kersten, G.E. and H. Lai, Satisfiability and Completeness of Protocols for Electronic Negotiations. European Journal of Operational Research, 2007. 180(2): p. 922-937.

Kim, J.B., G.E. Kersten, S. Strecker, and K.P. Law, On Designing E-negotiation Systems: Component-based Software Protocol Approach. Group Decision and Negotiation, 2007. forthcoming.

Kolitz, K. and C. Weinhardt. MES - Ein Experimentalsystem zur Untersuchung elektronischer Märkte. in MKWI 2006. 2006. Passau, Germany.

Kowalczyk, R. and V. Bui. On Fuzzy e-Negotiation Agents: Autonomous negotiation with incomplete and imprecise information. in DEXA Workshop on e-Negotiation. 2000. London, UK.

Kowalczyk, R. and V. Bui, On Constraint-based Reasoning in e-Negotiation Agents, in Agent Mediated Electronic Commerce III, F. Dignum and U. Cortés, Editors. 2000, Springer. p. 31 - 46.

Kowalczyk, R., V. Phiong, S. Dunstall, and B. Owens, Towards Supporting Collaborative Scheduling in Adaptive Supply Networks with Negotiation Agents. Journal of Decision Systems, 2004. 13(4): p. 441-460.

Kunzelmann, M., D. Neumann, and C. Weinhardt. Zwischen Limit und Market Order - Neue Ordertypen zur Reduktion impliziter Transaktionskosten. in 10th Symposium on Finance, Banking, and Insurance. 2005. Karlsruhe.

Lai, H. Collective Bargaining Models on the Internet. in SSGRR 2002S, International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, e-Medicine on the Internet. 2002. L'Aquila, Italy.

Maekioe, J. and I. Weber, Component-based Specification and Composition of Market Structures, in Coordination and Agent Technology in Value Networks, M. Bichler and C. Holtmann, Editors. 2004, GITO: Berlin. p. 127-137.

Manheim, M. An Architecture for Active DSS. in Proc. of the 21st Hawaiian International Conference on Systems Sciences. 1988.

Momotko, M., M. Gajewski, A. Ludwig, R. Kowalczyk, M. Kowalkiewicz, and J.Y. Zhang, Towards Adaptive Management of QoS-aware Service Compositions. The International Journal of Multiagent and Grid Systems, 2006.

Neumann, D., Market Engineering - A Structured Design Process for Electronic Markets, in Fakultät für Wirtschaftswissenschaften. 2004, Universität Karlsruhe (TH): Karlsruhe.

Nisan, N. and A. Ronen, Algorithmic Mechanism Design. Games and Economic Behavior, 2001. 35: p. 166-196.

Nwana, H.S., J. Rosenschein, T. Sandholm, C. Sierra, P. Maes, and R. Guttmann. Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints. in Second International Conference on Autonomous Agents. 1998. Minneapolis/St.Paul, MN: ACM Press.

Rahwan, I., R. Kowalczyk, and H.H. Pham. Intelligent Agents for Automated One-to-Many e-Commerce Negotiation. in ACSC. 2002.

Roth, A.E., V. Prasnikar, M. Okuno-Fujiwara, and S. Zamir, Bargaining and Market Behaviour in Jerusalem, Ljubljana, Pittsburgh, and Tokyo: An Experimental Study. The American Economic Review, 1991. 81(5): p. 1068-1095.

Shaw, M., J., M. D. Gardner, and H. Thomas, Research Opportunities in Electronic Commerce. Decision Support Systems, 1997. 21: p. 149-156.

Silverman, B.G., M. Bachann, and K. Al-Akharas, Implications of buyer decision theory for design of e-commerce websites. International Journal of Human-Computer Studies, 2001. 55(5): p. 815-844.

Sprague, R.H.J. and E.D. Carlson, Building Effective Decision Support Systems. 1982, Englewood Cliffs, NJ: Prentice-Hall.

Strecker, S., G. Kersten, J. Kim, and K.P. Law. Electronic Negotiation Systems: The Invite Prototype. in Proceedings of the Collaborative Business MKWI'06. 2006. Potsdam, Germany: GITO.

Ströbel, M., Design of Roles and Protocols for Electronic Negotiations. Electronic Commerce Research Journal, 2001. 1(3): p. 335-353.

Tseng, C.-C. and P.J. Gmytrasiewicz. A Real Time Decision Support System for Portfolio Management. in Thirty-Fifth Annual Hawaii International Conference on System Sciences. 2002. Big Island, Hawaii: IEEE.

Vahidov, R. and G.E. Kersten, Decision Station: Situating Decision Support Systems. Decision Support Systems, 2004. 38(2): p. 283-303.

Weber, I., C. Czernohous, and C. Weinhardt. Simulation of Ending Rules in Online Auctions. in Eleventh Research Symposium on Emerging Electronic Markets (RSEEM 2004). 2004.

Weinhardt, C., C.v. Dinther, K. Kolitz, J. Mäkiö, and I. Weber. meet2trade: A generic electronic trading platform. in Proceedings of the 4th Workshop on e-Business (WEB 2005). 2005. Las Vegas, USA.

Weinhardt, C., D. Neumann, and C. Holtmann, Computer-aided Market Engineering. Communications of the ACM, 2006. 49(7): p. 79.

Weinhardt, C., C. van Dinther, M. Grunenberg, K. Kolitz, M. Kunzelmann, J. Mäkiö, I. Weber, and H. Weltzien, CAME-Toolsuite meet2trade - auf dem Weg zum Computer Aided Market Engineering. 2006: University Press Karlsruhe.

Wooldridge, M. and N. Jennings, Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 1995. 10(2): p. 115-152.