

# Managing e-Market Negotiation in Context with a Multiagent System

John Debenham  
University of Technology, Sydney  
debenham@it.uts.edu.au

**Abstract.** An experimental electronic market that supports various mechanisms for negotiation on a single issue—typically price—in electronic exchanges is supplemented here with the addition of a rich form of negotiation that operates in the context of the market data and information extracted from the Internet generally. An alternating offers negotiation mechanism supports multi-issue, time-constrained negotiation where the issue set is open. All transactions in this market, including offers, trades and requests for information, are managed as constrained business processes. A multiagent system based on a three-layer BDI hybrid architecture attempts to manage these processes within their constraints. The long-term goal of this work is to investigate the e-market evolutionary process.

## 1 Introduction

An e-Market is described in [1]. In it single-issue negotiation—typically price—was supported using various market mechanisms in e-exchanges. Other forms of negotiation could only be conducted through a ‘solution provider’ that was little more than a conduit. A multi-issue, one-to-one negotiation mechanism has now been included. All forms of negotiation are now conducted in the context of information extracted from market data and from the Internet generally. The management of these negotiation processes and their attendant contextual information is achieved by a multiagent system.

*Negotiation* is a process whereby two or more agents reach an agreement on a set of issues. One-to-one negotiation, in which there are just two negotiating agents, is sometimes called *bargaining*, or informally “haggling” or “dickering”. An *issue* is any good or service that one agent can provide to another, including money. The *issue set* is the range of possible issues that may be considered during a negotiation. An issue set may be *fixed*; for example, in a single-issue negotiation where the only issue on which agreement is sought is an amount of money. An *open* issue set may contain *any* issue. In a *limited* issue set, the issues that may be included in an offer is limited to those chosen from a set agreed to by the negotiating agents. An issue—for example, “period of warranty”—is normally associated with some value—for example, “two years”. An *offer* consists of a particular set of issues chosen from the issue set, together with values for those issues. During a negotiation with an open or limited issue set the collection of issues in an offer may mutate although in practice it tends to be moderately stable.

A *negotiation mechanism* specifies how a negotiation may proceed; they are sometimes called “interaction protocols” in multiagent systems work [2]. Two forms of negotiation mechanism have received a considerable amount of attention in the economics literature, and in e-Markets research. First, single issue negotiation for

one or more items being offered either to a set of buyers or to a set of sellers; see, for example, the extensive work on forward and reverse auction mechanisms [3]. Second, one-to-one negotiation, or bargaining, mechanisms [4]. Further, two important classes of bargaining mechanisms are alternating offers mechanisms and single-round, “one-hit” mechanisms that may be used when the agents have determined their private valuations in advance. For example, [5] shows that a one-hit “split the difference between bid and ask” mechanism should be preferred by both buyer and seller to *any other* mechanism *ex ante*—that is, before their private valuations are actually determined. Alternatively, the agents’ valuations may be refined as the negotiation proceeds—in which case an alternating offers mechanism in which information is tabled as appropriate—this is the approach taken here.

The negotiation mechanism used is a time-constrained, unbounded alternating offers mechanism [6]. In this two bargaining agents exchange offers until either one agent accepts an offer from the other agent, one agent rejects an offer and withdraws without penalty, or one agent exceeds an agreed time constraint on making an offer. So negotiation using this mechanism could, in principle, proceed indefinitely—hence the description *unbounded*.

Management of the negotiation process in an e-market—both for negotiation through e-exchanges and through single- and multi-issue one-to-one negotiation—includes a continual investigation of the negotiation *context* as well as the construction, evaluation and revision of offers. For example, the bone fide of the opponent may require verification, the quality of the goods should be confirmed, alternatives should be investigated and so on. In the experiments described here this information is assumed to be available on the Internet. A good e-market negotiator should conduct these contextual investigations as an integral part of the negotiation process [7]. “Good negotiators, therefore, undertake integrated processes of knowledge acquisition combining sources of knowledge obtained at and away from the *negotiation table*. They learn in order to plan and plan in order to learn” [8]. In the negotiation processes described here, the information and the offers develop in tandem; they both feed off each other. The term “e-marketplace” is used here to acknowledge this duality between offers and contextual information. An *e-marketplace* is a market in which trading can be conducted over the Internet, and for which sufficient information to trade “well” is available over the Internet. This information may be derived from on-line market data, for mining historic market data, from text-mining news feeds and so on. The Sydney Stock Exchange is an example of an e-marketplace.

The original e-market was constructed during 2001 in a collaborative research project between the University of Technology, Sydney (UTS) and Bullant Australasia Pty Ltd—an Australian software house with an interest in business-to-business (B2B) e-business. The e-market was part of their on-going research effort in this area. It was constructed using Bullant’s proprietary software development tools and was designed by the author. Following the unfortunate events in the United States in September 2001, Bullant ceased trading. The e-market software has recently been rewritten at UTS in Java and is used to support teaching and research.

A three-year research project commencing in 2001 at UTS, is investigating the processes required to support evolution in e-markets [9]. It is presently funded by four Australian Research Council Grants; awarded variously to the author and to Dr Simeon Simoff:

<http://www-staff.it.uts.edu.au/~emrktest/eMarket/>

Market evolution is linked to innovation and entrepreneurship (in its technical, economic sense [10]). Present plans for the three year project are: (1) to build an e-marketplace trader's workbench that, in principle, enables a trader to operate without external information, (2) to assist a trader to identify arbitrage opportunities triggered by the occurrence of rare events [11], (3) to assist a trader to identify innovative forms of trade, and, possibly, (4) to understand something of the evolutionary process itself.

## 2 Two classes of business process

Here e-marketplace *transactions* include: trading orders to buy and sell in an e-exchange, single-issue and multi-issue negotiations between two parties, requests for information extracted from market data *as well as* from news feeds and other Internet data. One feature of this project is that *every* market transaction is managed as a business process. To achieve this, suitable process management machinery is required. To investigate what is 'suitable' the essential features of these transactions are related to two classes of process that are at the 'high end' of process management feasibility. The two classes are goal-driven processes and knowledge-driven processes. The term "business process management" is generally used to refer to the simpler class of workflow processes [12], although there notable exceptions using multiagent systems [13] [14].

### 2.1 Goal-driven processes

A *goal-driven process* has a process goal, and achievement of that goal signals the termination of the process. The process goal may have various decompositions into possibly conditional sequences of sub-goals such that these sub-goals are associated with (atomic) activities and so with atomic tasks. Some of these sequences of tasks may work better than others, and there may be no way of knowing which is which [15]. A task for an activity may fail outright, may fail to achieve its goal, or may violate process constraints. In other words, a central issue in managing goal-driven processes is the management of task failure. Hybrid multiagent architectures whose deliberative reasoning mechanism is based on "succeed/fail/abort plans" [16] are well suited to the management of goal-driven processes. Goal-driven processes are a more powerful concept than production workflows (called "activity-driven processes" in [17]). *Activity driven-processes* are associated with possibly conditional sequences of activities where performing that sequence is assumed to "work always".

Following [16] a plan for a goal-directed process can not necessarily be relied upon to achieve its goal even if all of the sub-goals on the chosen path through that plan have been achieved. The *success condition* (SC), described in [17], is a procedure whose goal is to determine whether a plan's goal has been achieved. The final sub-goal on *every* path through a plan is the plan's success condition. The success condition is a procedure; the execution of that procedure may succeed (✓), fail (✗) or abort (A). If the execution of the success condition fails then the overall success of the plan is unknown (?). So the four possible plan exits resulting from an attempt to execute such a plan are as shown in Fig. 1. A *plan body* is represented as a directed AND/OR graph, or state-transition diagram, in which some of the nodes are labelled with sub-goals. The generic plan is shown in Fig. 2.

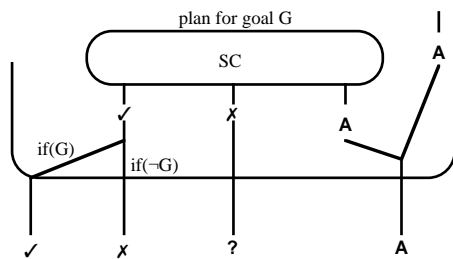


Figure 1. The four plan exits

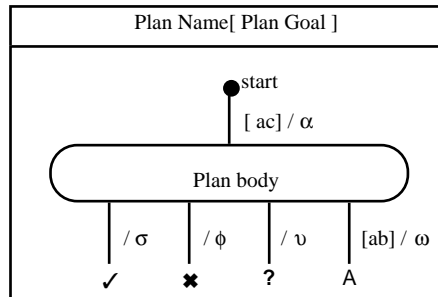


Figure 2. Generic plan

The management of goal-driven processes is shown in a simplified form in Fig. 3. There, starting with the overall process goal, repeated decomposition of plans and goals is performed until either the next goal is a success condition or is an *activity goal*—ie: a goal for which there is a hard-coded procedure. Fig. 3 is simplified because it does not show what happens if the success condition returns fail “X”, or what happens if a plan is aborted. Further it does not show the mechanism for selecting plans for goals. For a goal in a goal-driven process there is, in general, no *ex ante* ‘best’ choice of plan.

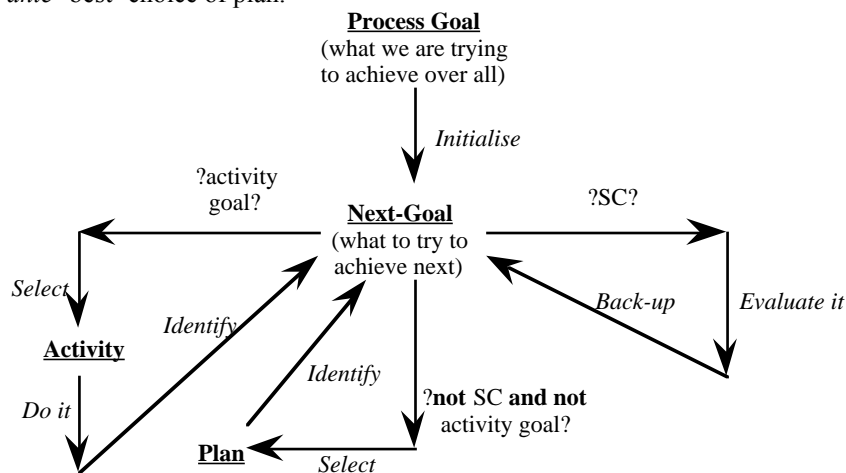


Figure 3. A simplified view of goal-driven process management.

## 2.2 Knowledge-driven processes

A second class of process, whose management has received little attention, is called knowledge-driven processes [17]. *Process knowledge* is all the knowledge that is relevant to a process instance. It includes common-sense knowledge, knowledge that was available when an instance is created, and knowledge acquired during the time that that the instance exists. A *knowledge-driven process* may have a process goal, but the goal may be vague and may mutate. In so far as the process goal gives direction to goal-driven—and activity-driven—processes, the process knowledge gives direction to knowledge-driven processes. The body of process knowledge is typically large and continually growing and so knowledge driven processes are seldom considered as candidates for process management. They are typically

supported, rather than managed, by CSCW systems [15]. But, even complex knowledge-driven processes are “not all bad”—they typically have goal-driven sub-processes which may be handled as described above. *Knowledge-base processes* are a special type of knowledge-driven process for which the process knowledge *can* be represented and accessed by a process management system. This proves to be a useful concept for managing e-marketplace transactions.

The management of goal-driven and knowledge driven processes are radically different. Goal-driven processes may be managed by a goal/plan decomposition process (see Fig. 3), and knowledge-driven processes are managed by continually reviewing the growing corpus of process knowledge—this is illustrated in Fig. 4. That Figure is deceptively simple in that the business of managing the process knowledge and of revising the process-goal and next-goal in the light of that growing body of knowledge is far from trivial in even simple examples. In general this problem will be intractable. But in some cases, including the majority of e-marketplace transactions, smart tools may be used to do this. This is discussed in the next section.

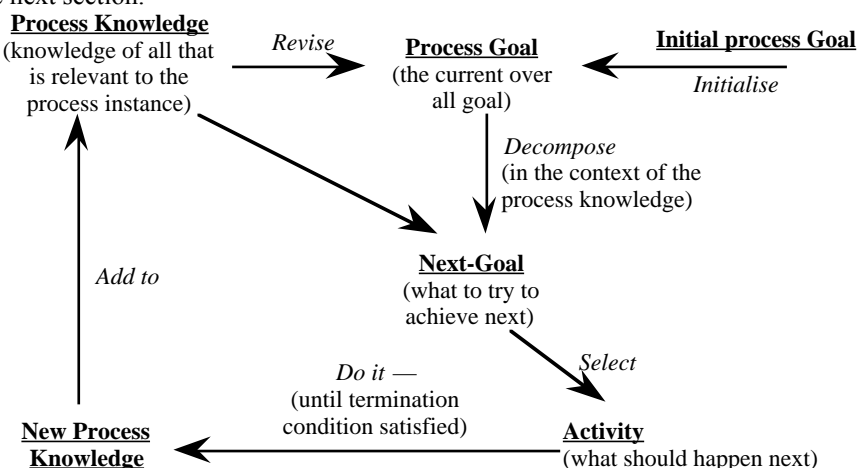


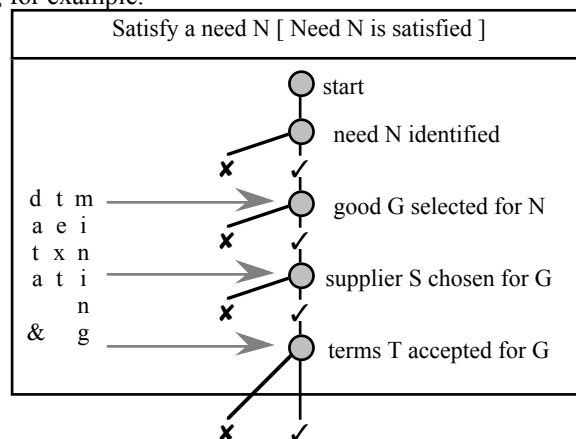
Figure 4. A simplified view of knowledge-driven process management.

### 3 Managing single-issue negotiation

Single-issue negotiation is the most common form of negotiation, in particular where the issue is price. The number of issues in any form of negotiation, including single-issue, can be increased if one of the negotiating parties offers “kick backs”. For example, an offer of two free bottles of wine for every dozen bought *provided that* you have spent more than \$500 with that merchant in the previous twelve months. This sort of offer raises what was initially a single-issue negotiation to a multi-issue negotiation. In this Section it is assumed that the negotiation is strictly single-issue and that the meaning of the issue is understood by both parties. For example, such an issue could be an amount of money. It is argued that single-issue negotiation is appropriately managed as a knowledge-driven process. From a process management perspective this is interesting because the management of “real life” knowledge-driven processes is usually unfeasible.

Consider a transaction to purchase something. Suppose that this transaction can be appropriately managed by: identifying a need, selecting a good to satisfy that need, choosing a supplier for that good and negotiating terms for that good from that supplier. A procedure based on this would not be appropriate for purchasing all classes of goods; it could, however, be suitable, for example, for purchasing a technical book. The appropriateness of this “purchasing procedure” is not of concern here. Suppose further that we wish to select the “most appropriate” good, to choose the “best supplier” and to negotiate “acceptable” terms. In an e-marketplace sufficient information to trade successfully in this sense is assumed to be available.

The use of “software bots” to assist the buying process by extracting contextual information from the Internet is common place. For many classes of goods, bots that do some of this work are freely available: <http://www.botspot.com/> — viz: the sections “Shopping Bots” and “Commerce Bots”. The entire problem considered here lies beyond the capacity of most off-the-shelf bots that at best *recommend* rather than *decide*. Although the use of demographic data, collaborative filtering, clustering, or previously expressed user preferences can deliver reasonable performance in choosing the “most appropriate” good for the user. It could be reasonable to give the authority to one of these bots to select, order and pay for paper stock for photocopiers, but many would be reluctant to permit a bot to select, order and pay for a book on Bayesian Nets, for example.

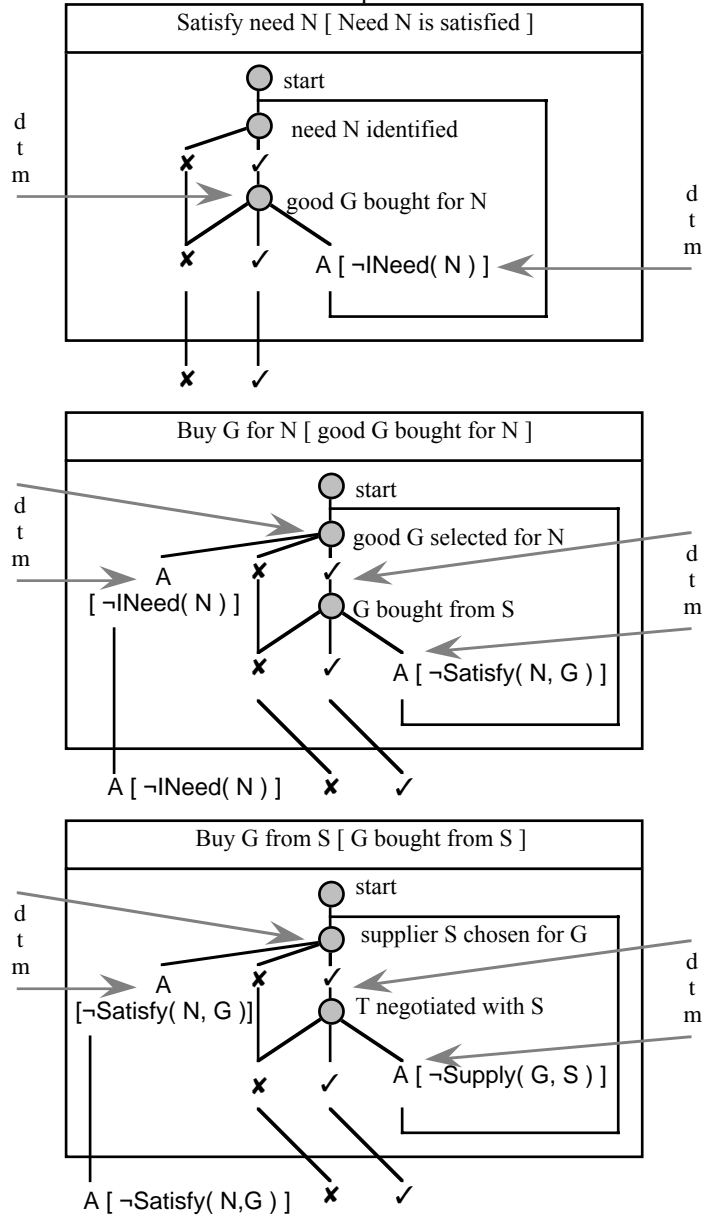


**Figure 5.** A goal-driven plan to satisfy a need based on “succeed / fail” plans.

In this project the contextual information from the Internet is first extracted by a range of data and text mining bots mostly written by undergraduates at UTS. Some of these bots read reviews of products in an attempt to determine the comparative inherent quality of a good as well as its basic attributes. This in general leads to a collection of contradictory evidence that is combined to give coherent advice. The approach taken to plausible inference is described in Sec. 4

A simple, double (succeed / fail) branching plan to manage a “purchase something” transaction is shown in Fig. 5. That plan treats the process sequentially in that, for example, once the good is selected then its appropriateness is not reconsidered. This may be appropriate when the whole transaction can be resolved quickly, but could otherwise lead to a poor result. That plan relies on information from data and text mining bots to support the decision making in the achievement of three of its sub-goals. [Plans for those three sub-goals are not shown here.] That plan

manages the transaction as a goal-driven process. The management of the same transaction as a knowledge-driven process is described below. This is achieved by using the reactive “abort” conditions in the plans.



**Figure 6.** Knowledge-driven plans to satisfy a need based on “succeed / fail / abort” plans.

To simplify the discussion the operation of the data and text mining bots is hidden in the following predicates:  $INeed(N)$  that means “I need an  $N$ ”,  $Satisfy(N, G)$  that means “good  $G$  is the most appropriate good that satisfies need  $N$ ”,  $Supply(G, S)$  that means “supplier  $S$  is the best supplier of good  $G$ ”, and

Fair( G, T ) that means “the terms T for good G are at least reasonable”, and Accept( G, T ) that means “ the terms T for good G are acceptable”. Calculation of values to satisfy these predicates may take some time.

Fig. 6 shows three linked plans for the “purchase something” transaction. In that Figure “d t m” denotes information that is acquired from the Internet and databases by data and text mining bots and combined into coherent advice by a plausible inference network. These plans are more intricate than the previous goal-directed version. Even so they are flawed in that the process may now continue indefinitely; this difficulty is addressed by Constraints in Sec. 6 below. They include reactive abort triggers that redirect the course of the transaction if any prior decision ceases to be valid. The direction, and possible redirection, of this transaction is governed entirely by the contextual information received, and repeatedly reconfirmed, from the data and text mining bots. This plan is useful but is not particularly noteworthy in itself. What is of note, from a process management perspective, is that this is a fully managed knowledge driven process, despite its management by a goal / plan framework. It is a knowledge-base driven process where the knowledge base is the Internet and market data, and the query mechanism is the bots.

#### **4 Managing multi-issue negotiation**

The discussion in Sec. 3 on the management of single-issue negotiation is rather simplified in that it assumes that the single-issue in which the offers are expressed is unambiguously understood. In practice negotiation is more complicated than this. For example, when the issue is money then an amount expressed in Euros is readily understood, but when and where the payment has to be made may not be. If the issue set contains things like an “unconditional warranty” then it may be prudent to clarify quite what this really means. So the feature of multi-issue negotiation that is explored here is the opponent agent as a source of information, that is used to clarify the meaning of an offer or otherwise.

A negotiation process with fairly minimal functionality is shown in Fig. 7. There the process is triggered by the arrival of an offer from the opponent. This is analysed to ensure that the meaning is clear—to uncover the “fine print”— and to detect any inconsistencies. Then the offer is evaluated to determine what it is “worth”—this can lead to either acceptance, outright rejection, or to the development of a counter offer. Part of the context for the generation of a counter offer is the history of offers received in this negotiation—this enables an assessment to be made of “where are the opponent is at”. Eg: “is she about to give in?”. The process illustrated in Fig. 7 can be interpreted as an attempt to satisfy the high level goal “attempt to negotiate a satisfactory outcome”. But the direction that the process takes is determined entirely by the flow of information—from the offer itself, the data and text mining bots, the opponent and from the growing history of offers. So this is a knowledge-driven process. At present the evaluation function is available from the bots as described above. At the time of writing the rest of the machinery is not yet available, but plans are to achieve this by the end of 2002. There is much to be done, for example the detection of inconsistencies in an offer is not trivial even if the terms of the offer can be represented in Horn clause logic.

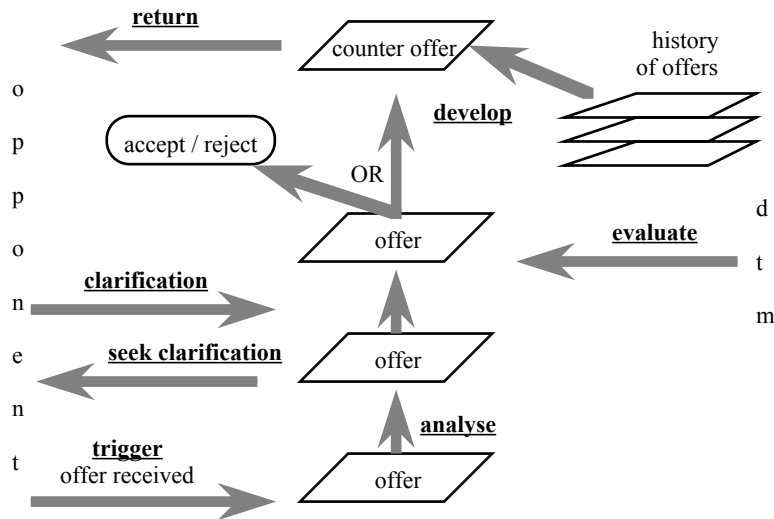


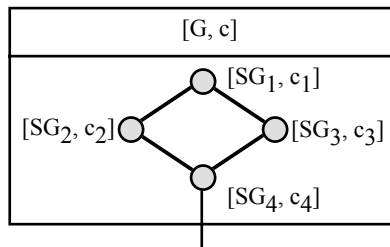
Figure 7 High-level view of the negotiation process

## 5 e-Market Transactions and Contextual Information

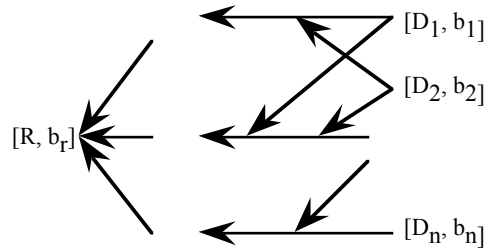
*E-market transactions* include: trading orders (both single-issue and multi-issue negotiations as described above), requests for market data *as well as* requests for information extracted from news feeds and other Internet data. *All* e-market transactions are managed as constrained knowledge-driven processes.

Sec. 3 discusses single-issue one-to-one negotiation. Single issue negotiation also takes place in exchanges, for example a ‘buy’ trading order to “buy a chair and a desk for less than \$100”. This may be represented (see Fig. 8) as a plan with goal [desk and chair have been purchased, cost < 100]. This plan has sub-goal SG<sub>1</sub> = ‘chair and desk selected’, [SG<sub>2</sub>, c<sub>2</sub>] = [chair purchased, cost < 30], [SG<sub>3</sub>, c<sub>3</sub>] = [desk purchased, cost < 50], and [SG<sub>4</sub>, c<sub>4</sub>] = [desk and chair delivered, cost < 20]. This purchase order is represented as a plan whose goals have monetary constraints.

An example of a request for information is “find out all you can about ABC Corp within five minutes”. This triggers a process to locate, extract, validate, condense and combine information. All, except combining information, is achieved by data/text mining bots [9]. Combining contradictory information is presently achieved by a Dempster-Shafer belief network. The use of belief nets that can be trained “off line” is very tempting and is currently being investigated [18]. The data/text mining bots produce output in the form [ data, belief ]—ie: some data and a measure of the belief held in the validity of that data. A request for information is first represented as a goal/constraint pair: [ find\_info\_about(‘ABC Corp’): time\_upper\_limit = now + 5mins ]. Given a goal/constraint pair a plan (see Fig. 8) is selected for it—the mechanism for selecting a plan is described in Sec. 6. A *plan* for a goal/constraint pair is a possibly-conditional sequence of sub-goals over which constraints are distributed as described in Sec. 6. For ‘find\_info\_about’ processes, the plans uses Dempster-Shafer networks (see Fig. 9) to combine results [D<sub>i</sub>, b<sub>i</sub>], in the form [ data, belief ], extracted from the Internet by a suite of data/text mining bots. The network



**Figure 8.** A plan for goal [G, c]



**Figure 9.** Belief network combines information

actually does more than combine information. If the level of belief,  $b_r$ , in a result,  $R$ , derived by the network is below a set threshold then a ‘reverse calculation’ identifies ‘inputs’ whose belief levels are responsible for the low level of belief in  $R$ . Then further data/text mining is sought in an attempt to raise this level of belief at least for future calculations if not for the present.

## 6 Transaction constraints

All e-marketplace transactions are assumed to be constrained by time constraints and possibly by cost constraints or success constraints. *Time constraints* may be the maximum (or minimum) time by which a deal must be struck and/or by which the goods should be delivered. The *cost constraints* could be constraints on the cost of the transaction, the cost of the goods or a combination of the two. *Success constraints* may be constraints on the outcome of the deal; for example, “I must have a car for the weekend, get the best deal you can”.

The e-marketplace transaction management system attempts manage transactions to “deliver the best it can whilst satisfying the constraints”. To do this it selects plans to achieve goals on the basis of expected time and cost estimates. Further, if actual performance differs significantly from these estimates then estimates for subsequent plans are adjusted leading, possibly, to a revised plan. This will occur if network performance is unexpectedly degraded, for example. To derive time and cost estimates for each plan it gathers performance measurements on each plan and sub-system, such as an information gathering bot, and maintains running estimates of future expected performance. It then adjusts these estimates when measurements are observed outside expected limits. For example, if the network is slow when gathering data from New York, then time estimates for extracting data from London may be adjusted to some extent.

Time and cost performance measurements are made for each plan and for each atomic sub-system whenever it is used. These measurements enable the transaction management system to choose a plan for a goal (eg:  $G$  in Fig. 8) and to determine the constraints  $\{c_1, \dots, c_4\}$  for each sub-goal in that plan. A plan’s *performance estimate* is the expected time “ $t$ ” and cost “ $c$ ” to satisfy the plan’s goal. These estimates are calculated from performance estimates for each atomic sub-system. The parameters  $t$  and  $c$  are assumed to be normally distributed—this is a wild assumption—but it provides a framework for identifying measurements that abnormal. Given a parameter,  $p$ , that is assumed to be normally distributed, an estimate,  $\mu_p$ , for the mean of  $p$  is revised on the basis of the  $i$ ’th observation  $ob_i$  to  $\mu_{p_{new}} =$

$(1 - \alpha) \times \text{ob}_i + \alpha \times \mu_{p_{\text{old}}}$  which, given a starting value  $\mu_{p_{\text{initial}}}$ , and some constant  $\alpha$ ,  $0 < \alpha < 1$ , approximates the geometric mean  $\left[ \frac{\sum_{i=1}^n \alpha^{n-i} \times \text{ob}_i}{\sum_{i=1}^n \alpha^{n-i}} \right]$  of the set of observations  $\{\text{ob}_i\}$  where  $i = n$  is the most recent observation. In the same way, an estimate,  $\sigma_p$ , for  $\sqrt{\frac{2}{\pi}}$  times the standard deviation of  $p$  is revised on the basis of the  $i$ 'th observation  $\text{ob}_i$  to  $\sigma_{p_{\text{new}}} = (1 - \alpha) \times |\text{ob}_i - \mu_{p_{\text{old}}}| + \alpha \times \sigma_{p_{\text{old}}}$  which, given a starting value  $\sigma_{p_{\text{initial}}}$ , and some constant  $\alpha$ ,  $0 < \alpha < 1$ , approximates the geometric mean  $\frac{\sum_{i=1}^n \alpha^{n-i} \times |\text{ob}_i - \mu_p|}{\sum_{i=1}^n \alpha^{n-i}}$ . The constant  $\alpha$  is chosen on the basis of the

stability of the observations. For example, if  $\alpha = 0.85$  then “everything more than twenty trials ago” contributes less than 5% to the weighted mean.

Given a transaction and its constraints (expressed in terms of  $t$  and  $s$ ), the transaction management system makes two decisions. First it selects a feasible plan for that transaction's goal. Second it determines the constraints on each sub-goal in that plan. Then further plans are selected for those sub-goals, and so on. Each time a plan for goal  $G$  is used measurements are made of  $t$  and  $c$  for each sub-goal in that plan. Further each of those sub-goals may be invoked by other plans. So the estimates of the mean and standard deviation of  $t$  and  $c$  for those sub-goals may be expected to be more accurate than the estimates for goal  $G$ . So each time a plan is considered, the  $t$  and  $c$  estimates for its goal are re-computed from those on the estimated costliest path through that plan.

Each plan has subjective values representing the expected mean and variance of the quality,  $q$ , of the results that it generates. The transaction management system selects a plan for a given goal  $G$  using the stochastic strategy: the probability that a plan is selected is the probability that that plan is the “best” plan. This strategy has been found to work well for managing high level processes [17]. Here *best* means the plan expected to deliver the highest quality result whilst satisfying the constraints placed on it. Given two plans  $A$  and  $B$  for the same goal  $G$ , if their expected performance is represented by a parameter  $p$  (in terms of  $q$ ,  $t$  and  $c$ ) that is assumed to be normally distributed then the probability that plan  $A$  is “better than” plan  $B$  is the probability that  $(p_A - p_B) > 0$ . Using elementary statistics, an estimate for this probability is given by the area under the normal distribution with:

mean =  $\mu_A - \mu_B$  [ where  $\mu_A$  and  $\mu_B$  are estimates of the means of  $p_A$  and  $p_B$  ]

standard deviation =  $\sqrt{\frac{\pi}{2} \times (\sigma_A^2 + \sigma_B^2)}$  [ where  $\sigma_A$  and  $\sigma_B$  are estimates of  $\sqrt{\frac{2}{\pi}}$  times the standard deviations of  $p_A$  and  $p_B$  ] for  $x > 0$ . This method may be extended to estimate the probability that one plan is better than a number of other plans.

Plan  $A$  for the constrained goal  $[G, c]$  is *feasible* if  $c > \mu_A + \kappa \times \sigma_A$ , where  $\kappa$  is a constant usually  $> 1$ . If  $c < \mu_A - \kappa \times \sigma_A$  then the plan is not expected to achieve its goal within constraint  $c$ . This enables the constraints to be relaxed on some sub-goals as long as the estimated costliest path through the plan satisfies  $c$ . If a sub-goal  $SG_i$

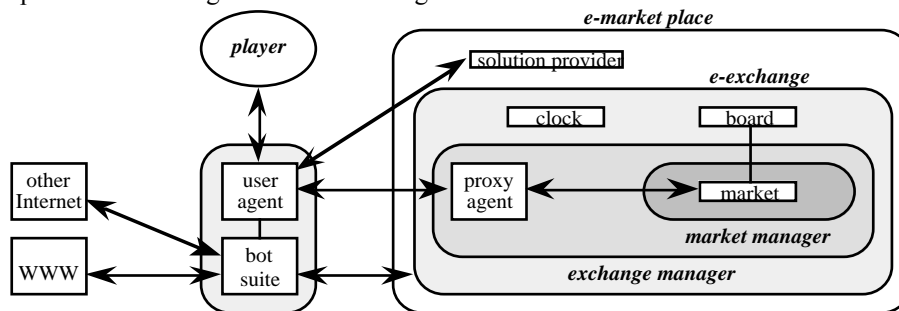
of plan P for goal G is not achieved within its constraint  $c_i$  then *first* another plan is sought for  $SG_i$  and for any other as-yet-unsatisfied ‘down stream’ sub-goals, for which an allocation of constraints in P is feasible, and *second* the whole plan P fails and another plan is sought for G with tighter constraints than c.

The adjustment of estimates in the light of measurements that fall outside expected ranges is achieved using the geometric weighted mean method used to estimate s and t. The *multipliers*  $v_{ij}$  mean: if measurement  $m_i$  of service i lies outside the expected range then multiply the estimate for service j by  $\frac{m_i}{\mu_i} \times v_{ij}$ . This is crude but in a sense these multipliers are no cruder than the estimates that they are adjusting. What is known is the network topology and so too potential causal links between components’ performance. The use of some form of belief net [18] is appealing in that the learning mechanism has a scientific basis, although here the nets represent conditional t and c estimates rather than conditional probabilities. This is presently being investigated.

## 7 The system architecture

The system consists of an e-marketplace, the World Wide Web and other Internet sources, and a user agent with access to a suite of data/text mining bots for each player. See Fig. 10. The e-marketplace is written in Java and is a framework into which individual markets and exchanges, with their own rules, can be ‘plugged’. For example, one market in the e-marketplace is linked to the Sydney Stock Exchange; it enables agents to ‘trade’ without risking any funds. There are two functional components in the *e-marketplace*: an e-exchange and a solution provider.

An *e-exchange* is a virtual space in which a variety of *markets* can take place. The market regulations include the specification of the *market mechanism* [19]. Designing market mechanisms is an active area of research [20]. A human *player* works through a PC from which a *user agent* communicates with a *proxy agent* or a solution provider situated in the e-market. Transactions are discussed in Sec. 5. The *user agents* are described below; they are essentially transaction management agents with a friendly front end. The user agents together form the multiagent system that manages the e-marketplace transactions. The *proxy agents* are ‘dumb’ and simply represent the user agent in an e-exchange.



**Figure 10.** High-level view of the e-marketplace and player.

A variety of architectures have been described for autonomous agents [2]. A fundamental distinction is the extent to which an architecture exhibits deliberative (feed forward, planning) reasoning and reactive (feed back) reasoning. Architectures

that combine these two forms of reasoning are called *hybrid architectures*. One well reported class of hybrid architectures is the three-layer, belief-desire-intention (BDI) agent architectures. One member of this class is the INTERRAP architecture [21], which has its origins in the work of [16]. An architecture similar to INTERRAP was developed to manage emergent processes [17] and is used here.

An agent's world beliefs [21] are derived *either* from reading messages received from the player, another agent, the e-marketplace, *or* the bots. Beliefs play two roles. First, they can be partly or wholly responsible for the agent committing to a goal, and may thus initiate an intention (eg: a plan to achieve what a message asks, such as "buy xyz"). This is *deliberative reasoning*. Second, they can be partly or wholly responsible for the activation of a 'procedure trigger' that will pass data to a partly executed plan. This is *reactive reasoning*.

Reactive reasoning play two roles: first, a plan is aborted if its specified abort condition is satisfied, and second, data is passed to partly executed plans for goals an agent is committed to achieve. Of these two roles the first takes precedence over the second. Reactive reasoning is achieved by rules of the form:

**if** <trigger state> **and** <belief state> **then** <action> **and** <trigger state>

where the <trigger state> is a device to determine whether the trigger is active or not, and <belief state> is something that may be in the agent's world beliefs. Data is passed to partly executed plans using *procedure triggers*. For example, the sub-goal [SG<sub>2</sub>, c<sub>2</sub>] of the plan illustrated in Fig. 8 is [chair purchased, cost < 30]. This sub-goal may be achieved if "chair has been purchased for \$28" is present in the agent's world beliefs. So until such a belief materialises this sub-goal may "hang". This situation is managed by linking the sub-goal to a procedure trigger "**if** waiting for chair purchase **and** chair has been purchased for \$x **then** goal [chair purchased, cost < 30] is satisfied **and** not waiting for chair purchase". This procedure trigger "watches" the user agent's world beliefs.

## 8 Conclusion

The negotiation process, and other e-marketplace transactions, are knowledge-driven processes—the direction that they take is determined by contextual knowledge and by knowledge generated during a process instance. They may be managed by a hybrid multiagent system based on a 3-layer BDI architecture. These processes are typically heavily constrained. These constraints complicate the business of selecting plans to achieve goals in the unreliable e-marketplace environment that includes the Internet. Methods are given to achieve this that are based on continually undated performance measurements of each atomic chunk of the system.

## References

- [1] Debenham, JK. Supporting the actors in an electronic market place. In proceedings Twenty First International Conference on Knowledge Based Systems and Applied Artificial Intelligence, ES'2001: Applications and Innovations in Expert Systems IX, Cambridge UK, December 2001, pp29—42.
- [2] Weiss, G.(ed). Multi-Agent Systems. The MIT Press: Cambridge, MA (1999).
- [3] Klemperer, P. (Ed). The Economic Theory Of Auctions. Edward Elgar Publishing (2000).

- [4] Osborne, MJ and Rubinstein, A. *Bargaining and Markets*. Academic Press (1990).
- [5] Myerson, R. & Satterthwaite, M. Efficient Mechanisms for Bilateral Trading. *Journal of Economic Theory*, 29, 1–21, April 1983.
- [6] Kraus, S. *Strategic Negotiation in Multiagent Environments*. MIT Press, 2001.
- [7] Debenham, JK and Simoff, S. Designing a Curious Negotiator. In proceedings Third International Workshop on Negotiations in electronic markets - beyond price discovery — e-Negotiations 2002, September 2002, Aix-en-Provence, France.
- [8] Watkins, M. *Breakthrough Business Negotiation—A Toolbox for Managers*. Jossey-Bass, 2002.
- [9] Debenham, JK and Simoff, S. Investigating the Evolution of Electronic Markets. In proceedings Sixth International Conference on Cooperative Information Systems, CoopIS 2001, Trento, Italy, September 5-7, 2001, pp344-355.
- [10] Israel M. Kirzner. "Entrepreneurial Discovery and the Competitive Market Process: An Austrian Approach" *Journal of Economic Literature* XXXV (March) 1997 60—85.
- [11] Debenham, JK. Identifying Arbitrage Opportunities in e-Markets. In proceedings 3rd International Conference on Electronic Commerce and Web Technologies—EC-Web 2002, September 2-6, 2002, Aix-en-Provence, France.
- [12] Fischer, L. (Ed). *Workflow Handbook 2001*. Future Strategies, 2000.
- [13] Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P. and Odgers, B. Autonomous Agents for Business Process Management. *Int. Journal of Applied Artificial Intelligence* 14 (2) 145-189 (2000).
- [14] Huhns, M.N. and Singh, M.P. Managing heterogeneous transaction workflows with cooperating agents. In N.R. Jennings and M. Wooldridge, (eds). *Agent Technology: Foundations, Applications and Markets*. Springer-Verlag: Berlin, Germany, 1998, pp. 219—239.
- [15] van der Aalst, W. & van Hee, K. *Workflow Management: Models, Methods, and Systems*. MIT Press (2001).
- [16] Rao, A.S. and Georgeff, M.P. "BDI Agents: From Theory to Practice", in proceedings First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, pp 312—319.
- [17] Debenham, JK. Supporting knowledge-driven processes in a multiagent process management system. In proceedings Twentieth International Conference on Knowledge Based Systems and Applied Artificial Intelligence, ES'2000: Research and Development in Intelligent Systems XV, Cambridge UK, December 2000, pp273—286.
- [18] Cowell, RG, Dawid, AP, Lauritzen, SL and Spiegelhater, DJ. *Probabilistic Networks and Expert Systems*. Springer-Verlag, (1999)
- [19] Bichler, M. *The Future of E-Markets: Multi Dimensional Market Mechanisms*. Cambridge University Press, 2001.
- [20] Perry, M., Wolfstetter, E. & Zamir, S. A Sealed-Bid Auction That Matches The English Auction. *Games and Economic Behaviour*, 33: 265—273, 2000.
- [21] Müller, JP. *The Design of Intelligent Agents: A Layered Approach (Lecture Notes in Computer Science, 1177)*. Springer Verlag (1997).