ADEPT: An Agent-Based Approach to Business Process Management

N. R. Jennings, T. J. Norman & P. Faratin
Department of Electronic Engineering,
Queen Mary and Westfield College,
University of London, London E1 4NS, U.K.

Abstract

Successful companies organise and run their business activities in an efficient manner. Core activities are completed on time and within specified resource constraints. However to stay competitive in today's markets, companies need to continually improve their efficiency — business activities need to be completed more quickly, to higher quality and at lower cost. To this end, there is an increasing awareness of the benefits and potential competitive advantage that well designed business process management systems can provide. In this paper we argue the case for an agent-based approach: showing how agent technology can improve efficiency by ensuring that business activities are better scheduled, executed, monitored, and coordinated.

1 Introduction

Company managers make informed decisions based on a combination of judgment and information from marketing, sales, research, development, manufacturing and finance departments. Ideally, all relevant information should be brought together before judgment is exercised. However obtaining pertinent, consistent and up-to-date information across a large company is a complex and time consuming process. For this reason, organisations have sought to develop a number of IT systems to assist with various aspects of the management of their business processes. Such systems aim to improve the way that information is gathered, managed, distributed, and presented to people in key business functions and operations. In particular, the IT system should: (i) allow the decision maker to access relevant information wherever it is situated in the organisation (this should be possible despite the fact that information may be stored in many different types of system and in many different information models); (ii) allow the decision maker to request and obtain information management services both from departments within and departments from outside the organisation; (iii) pro-actively identify and deliver timely, relevant information that may not have been explicitly asked for; (iv) inform the decision maker of changes which have been made elsewhere in the business process which impinge upon the current decision context; and (v) identify the parties who may be interested in the outcome and results of the decision making activity.

Analysis of a number of business processes, from various industrial and commercial domains, resulted in several common characteristics being identified:

- Multiple organisations are often involved in the business process. Each organisation attempts to maximise its own profit within the overall activity.
- Organisations are physically distributed. This distribution may be across a site, a country, or even continents. This situation is even more apparent for virtual organisations [Mowshowitz1996] which form allegiances for short periods of time and then disband when it is no longer profitable to stay together.
- Within organisations, there is a decentralised ownership of the tasks, information and resources involved in the business process.
- Different groups within organisations are relatively autonomous — they control how their resources are consumed, by whom, at what cost, and in what time frame. They also have their own information systems, with their own idiosyncratic representations, for managing their resources.
- There is a high degree of natural concurrency — many interrelated tasks are running at any given point of the business process.

Research funded by DTI/EPSRC Intelligent Systems Integration Programme.

- Business processes are highly dynamic and unpredictable — it is difficult to give a complete a priori specification of all the activities that need to be performed and how they should be ordered. Any detailed time plans that are produced are often disrupted by unavoidable delays or unanticipated events.

Given these characteristics, it was decided that the most natural way to view the business process is as a collection of autonomous, problem solving agents which interact when they have interdependencies. In this context, an agent can be viewed as an encapsulated problem solving entity which exhibits the following properties [Wooldridge and Jennings1995]:

Autonomy: agents perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they have control over their own actions and their own internal state.

Social ability: agents interact, when they deem appropriate, with other artificial agents and humans in order to complete their problem solving and to help others with their activities.

Pro-activeness: agents take the initiative where appropriate.

Responsiveness: agents perceive their environment and respond in a timely fashion to changes which occur in it.

The choice of agents as a solution technology was motivated by the following observations: (i) the domain involves an inherent distribution of data, problem solving capabilities, and responsibilities (conforms to the basic model of distributed, encapsulated, problem solving components); (ii) the integrity of the existing organisational structure and the autonomy of its sub-parts needs to be maintained (appeals to the autonomous nature of the agents); (iii) interactions are fairly sophisticated, including negotiation, information sharing, and coordination (requires the complex social skills with which agents are endowed); and (iv) the problem solution cannot be entirely prescribed (problem solvers need to be responsive to changes in the environment and to unpredictability in the process and pro-actively take opportunities when they arise).

When taken together, this set of requirements leaves agent technology as the strongest solution candidate — (distributed) object systems have the encapsulation but not the sophisticated reasoning required for social interaction or pro-activeness, and distributed processing systems deal with the distributed aspect of the domain but not with the autonomous nature of the components.

2 The ADEPT Architecture

The ADEPT architecture can be viewed at two levels: the architecture of the multi-agent system in which an agent acts, and the internal architecture of a single agent. The former represents the structure of the system as a whole, and the role of an agent within that system. The latter represents the separation of concerns of the functional components of a particular agent.

The ADEPT multi-agent architecture is composed of a number of autonomous agencies (see figure 1). The concept of an agency has a recursive definition. An *agency* contains a single *responsible agent*, a possibly empty set of *subsidiary agencies* and a set of *tasks* that are under the direct management of the responsible agent. In the ADEPT environment agents are autonomous; i.e. agents have control over the tasks that they may perform, the resources available to them and how they coordinate their activities with other agents. Therefore, the only way in which such agents may cooperate in solving problems is through *negotiation*. Agents operate by negotiating for *services*, or units of problem-solving activity, in the management of a business process (a task is an atomic service). A responsible agent's agency represents its domain problem solving resources.

The recursive definition of an agency allows a nested (hierarchical) agent system to be constructed in which a responsible agent realises its function through the responsible agents of lower level agencies (these lower level agents have the same structure and can, therefore, have subsidiary agents as well as tasks in their agency). For example, the responsible agent of agency A may represent a legal department whose work is carried out by a number of lawyers, each represented by the responsible agents of agencies such as A.1 and A.2. This structure enables flat, hierarchical, and hybrid organisations to be modeled in a single framework. The differences between the responsible agent of a subsidiary agency (A.1 and A.2 are subsidiaries of A) and that of a peer agent (A, B and C are peers, and A.1 and A.2 are peers) relate to their levels of autonomy and helpfulness. In both cases, the agents negotiate to reach agreements. However the responsible agent of a subsidiary agency cannot reject a proposal outright (although it can counter-propose until an acceptable agreement is reached), and it must negotiate in a cooperative (rather than a competitive) manner since there is a degree of commonality of purpose.

All ADEPT agents have the same basic internal architecture, illustrated by the responsible agent of agency A (figure 1). An agent has a number of functional components concerned with each of its main activities — communication, service execution, situation assessment, and interaction management. The agent architecture reflects this.

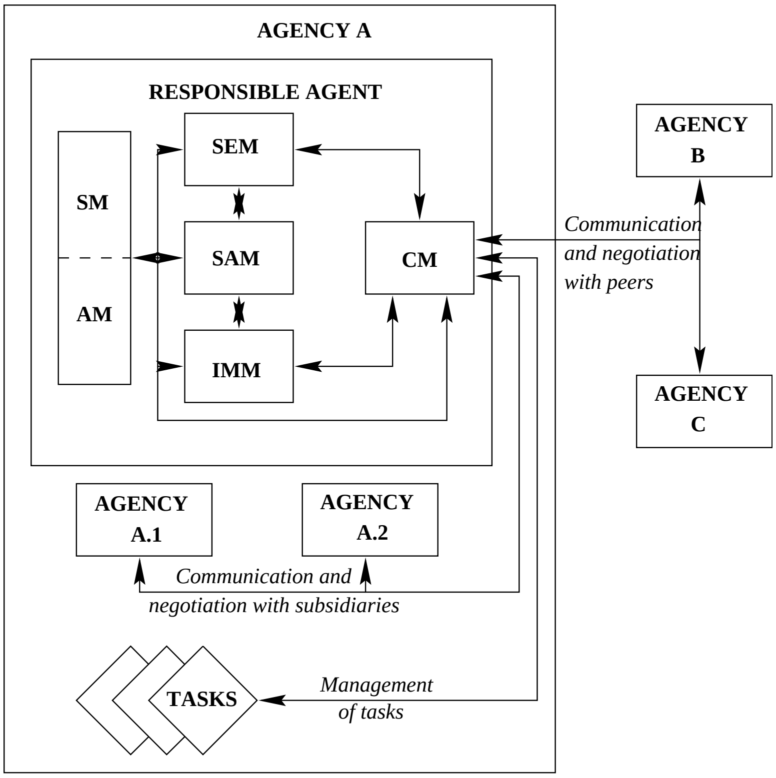

Figure 1: A view of the ADEPT architecture.

2.1 Communication Module (CM)

The CM routes messages between an agent and both its agency and peers. During task management (e.g. the activation, suspension, or resumption of a task), messages are routed between the agent's Service Execution Module (SEM) and the tasks managed by that agent. During service execution management (e.g. the initiation or termination of a service being provided by another agent under some agreement), messages are routed between the agent's SEM and the SEM of the agent providing the service. During negotiation, messages are routed between the agent's Interaction Management Module (IMM) and the IMM of the agent, or agents, being negotiated with.

2.2 Interaction Management Module (IMM)

The IMM provisions services through negotiation. The Situation Assessment Module (SAM) invokes the IMM to initiate negotiation for a service. The IMM's decision making capabilities are supported by three types of information: scheduler constraints emanating from the SAM; knowledge an agent has about itself and its own domain (represented in the Self Model (SM)); and knowledge the agent holds about responsible agents of both peer and subsidiary agencies (represented in the Acquaintance Model (AM)). With this knowledge and the agent's negotiation model, the IMM generates initial proposals, evaluates incoming proposals, produces counter-proposals, and, finally, accepts or rejects agreements for the provision of a service (i.e. Service Level Agreements, or SLAs).

2.3 Situation Assessment Module (SAM)

The SAM is responsible for assessing and monitoring the agent's ability to meet the SLAs it has already agreed and any SLAs that it may agree in the future. This involves scheduling and exception handling. The scheduler maintains a record of resource availability which can be used to determine whether SLAs can be met or whether new SLAs can be accepted. The exception handler receives exception reports from the SEM during service execution and decides upon the appropriate response. For example, if a service is delayed then the SAM may decide to locally reschedule it, to renegotiate the SLA, or to terminate it altogether.

2.4 Service Execution Module (SEM)

The SEM is responsible for managing services throughout their execution. This involves service execution management (start executing services as specified by the agent's SLAs), information management (routing information between tasks, services and other agents during execution), and exception handling (monitoring the execution of tasks and services for unexpected events and then reacting appropriately). In the event of task failure for example, the SEM may recover by attempting to restart the task if the present schedule can still be met, or report the exception to the SAM for rescheduling.

2.5 Acquaintance Model (AM) and Self Model (SM)

Within the AM, the agent maintains a record of peers and subsidiaries which can provide services of interest. The SM is the primary storage site for SLAs to which the agent is committed, descriptions of the services the agent can provide, run time application/service specific information, and generic domain information.

Both ADEPT's multi-agent and internal architectures are designed to ensure maximum flexibility to adapt as a business process changes (see [Jennings et al.1996] for more details). The autonomy of each agent and the agreements it enters into with others are the key to this flexibility. For this reason, ADEPT's negotiation technology is discussed further in section 3.

3 Negotiation

Services are associated with one or more agents that are responsible for managing and executing them. Each service is managed by one agent, although it may involve execution of sub-services by a number of other agents. Since agents are autonomous there are no control dependencies between them; therefore, if an agent requires a service which is managed by another agent it cannot simply instruct it to start the service. Rather, the agents must come to a mutually acceptable agreement about the terms and conditions under which the desired service will be performed (i.e an acceptable SLA must be instantiated). The mechanism for making SLAs is negotiation — a joint decision making process in which the parties verbalise their (possibly contradictory) demands and then move towards agreement by a process of concession [Müller1996].

There are three components of the ADEPT negotiation model: (i) the communication protocol, (ii) the service level agreements, and (iii) the reasoning model. The protocol specifies the communication primitives that an agent can use to query an agent on what services it can perform (an agent sends a CAN-DO primitive which is responded to with an I-CAN primitive), to agree on the details of an agreement (PROPOSE, COUNTER-PROPOSE, ACCEPT, and REJECT), and to manage the invocation of an agreement (i.e. instructing agents to activate, suspend or resume a service, and informing agents of completions or failures of a service).

Slot Name	Instantiated Values
SERVICE_NAME:	Cost-Design-Network
SLA_ID:	a1001
SERVER_AGENT:	DD
CLIENT_AGENT:	CSD
DELIVERY_TYPE:	on-demand
DURATION (minutes):	320
START_TIME:	9:00
END_TIME:	18:00
VOLUME:	35
PRICE (per costing):	35
PENALTY:	30
CLIENT_INFO:	customer_profile
REPORTING_POLICY:	customer_quote

Figure 2: Exemplar service level agreement.

The novel aspects of negotiation in the ADEPT system relate to the types of agreements that agents can make and the models they use to guide their negotiation behaviour. The requirements of the business process domain mean that agreements need to be more encompassing and the reasoning more elaborate than those found in most extant multi-agent systems. To this end, multi-lateral and multi-issue decision mechanisms have been developed [Faratin et al.1998] that assist an agent in evaluating offers and, when necessary, generating new offers. The latter mechanisms are composed of tactical and strategic decision making. Tactics model low level decisions that take into account the agent's environment (such as time, resources and other's behaviours) and its preferences. Strategies, in turn, model coarse grain and general behaviours which determine the overall style of negotiation (such as conciliatory or competitive negotiation).

The nature and scope of the SLAs are derived from the types of legal contract that are often used to regulate current business transactions (figure 2 shows a typical example taken from a BT application (see section 4)). Service_name is the service to which the agreement refers and sla_id is the SLA's unique identifier (covering the case where there are multiple agreements for the same service). Server_agent and client_agent represent the agents who are party to the agreement. Delivery_type identifies the way in which the service is to be provisioned. In figure 2 the delivery type is "on-demand". This indicates that the service may be invoked under this SLA whenever it is required by the client, but within the times stated and at a frequency below that indicated in the volume slot. Alternatively, the SLA may represent a "one-off" service. This indicates that the service can be invoked only once by the client within the time period indicated. The SLA's scheduling information is used by the SAM and the SEM for service execution and management — duration represents the maximum time the server can take to finish the service, and start_time and end_time represent the time during which the agreement is valid. In this case, the agreement specifies that an agent called CSD can invoke an agent called DD to cost and design a customer network whenever it is required between 09:00 and 18:00 and each service execution should take no more than 320 minutes. The agreement also contains meta-service information such as the volume of invocations permissible between the start and end times, the price paid per invocation, and the penalty the server incurs for every violation. Client_info specifies the information the client must provide to the server at service invocation (in this case CSD must provide the customer profile) and reporting_policy specifies the information the server returns upon completion.

Existing theoretical work on negotiation [Nash1950, Raiffa1982, Rosenschein and Zlotkin1994] provides important insights into how agents should negotiate to produce optimal solutions. However, a number of unrealistic assumptions are common in these negotiation models; typical assumptions include the availability of complete action descriptions, a utility function that can order all alternatives in all contexts, and that agents exhibit perfect rationality when selecting actions. In contrast, practical applications typically adopt simplistic approaches to negotiation. In the contract net protocol [Smith and Davis1981], for instance, a manager sends out a request to a number of potential contractors to provide a given service to a given degree of quality. The potential contractors return a bid if they are capable of fulfilling all the requirements. The manager then selects the best bid. However, this model fails to capture many intuitive and important aspects of the negotiation process. For example, bidders cannot counter-propose better options, they cannot modify any of the service agreement parameters, and the emphasis in devising a complete specification is placed solely with the task manager. Given these limitations, the approach within ADEPT has been to develop a deep and explicit model of the process of negotiation. The model covers the whole process of generating initial offers, evaluating offers, and counter proposing if offers are unacceptable (see [Faratin et al.1998] for more details). In this model, agents evaluate proposals. Then, using both a predictive model of the behaviour of other agents and its own preferences, a decision is made on whether to accept or reject the current proposal, or to counter-propose an alternative. Issues such as the time by which an agent requires the service, and minimum or maximum prices that are acceptable for a service are considered when the contents of the SLA to be proposed are determined. The parameters within which an agent may negotiate, e.g. the maximum price it is prepared to pay for a service, are dependent on the application.

4 Exemplar Application: BT's Provide Customer Quote Process

ADEPT technology has been used to develop business process management systems for a number of real-world applications. Here we outline one such application: a system for managing a British Telecom (BT) process for providing a quotation for designing a network to provide particular services to a customer. The process receives a customer service request as its input and generates as its output a quote specifying how much it would cost to build a network to realise that service. It involves up to six parties: the sales department, the customer service division, the legal department, the design division, the surveyor department, and the provider of an out-sourced service for vetting customers.

The process is initiated by a customer contacting the customer service division. The customer's details are captured, and, while the customer is being vetted (in terms of its credit worthiness, false ID, etc.), their requirements are elicited. If the customer fails the vetting procedure, then the quote process terminates. Assuming the customer is satisfactory, its requirements are recorded and mapped against the service portfolio. If the requirements can be met by a standard off-the-shelf item then an immediate quote can be offered based on previous examples. In the case of bespoke services, however, the process is more complex. The customer service division further analyses the customer's requirements and while this is occurring the legal department checks the legality of the proposed service. If the desired service is illegal, then the entire quote process terminates. If the requested service is legal, then the design phase starts. To prepare a network design it is usually necessary to have a detailed plan of the existing equipment at the customer's premises. Sometimes such plans might not exist and sometimes they may be out of date. In either case, the designer determines whether the customer site(s) should be surveyed. On completion of the network design and costing, the customer is informed of the service quote. The business process then terminates.

From the business process description, the following agent system was designed (figure 3). The agents (denoted by the circles) were chosen to represent distinct departments or enterprises involved in the business process. The VC agents represent the concerns of external enterprises as this activity is out-sourced. Agent SD is within DD's agency because the design division has overall management responsibility for the surveyors.

The process is triggered when the sales agent sends a request to the CSD agent to provide a customer quote. The CSD agent identifies the SLA associated with the request: in this case it relates to the Provide-Customer-Quote service. The corresponding service description is parsed to create a tree of possible routes that the SEM can take. A depth first path is selected and the tasks and services in that path are scheduled and resourced (by the SAM). The SEM begins executing the constituent sub-services and tasks. One of the first sub-services it encounters is to vet the customer (this occurs in parallel with the capture-customer-requirements task and after capture-customer-details). When the SEM comes to execute this service it realises (by checking its SM) there is no associated SLA and so it reports an exception to the SAM. The SAM determines that the service cannot be realised locally (by referring to its SM) and so it must be bought in from an external agent. It also decides that the service should be provisioned in an on-demand manner because it is an activity that is needed on each invocation of the business process. As such, it is preferable to negotiate for a longer term SLA covering multiple invocations rather than negotiating for one each time the business process is invoked. In addition to identifying the service name and the desired provisioning mode, the SAM indicates any scheduler information which influences the provisioning of the service (e.g. the service's earliest start and latest end times).

Vet customer service provisioning begins with the CSD agent sending CAN-DO messages to all the agents it can identify (using its AM) as being potentially able to provide this service (in this scenario there are three such agents: VC1, VC2 and VC3). These messages emanate from the IMM. Negotiation proper begins when CSD concurrently sends out initial proposals (in the form of instantiated SLAs) to all the vet customer agents which responded with I-CAN. This initial proposal may be acceptable to one of the VC agents in which case an agreement is made and the negotiation is terminated. However, in most cases the VC agents find some part of the proposal unsatisfactory (it is a competitive negotiation after all) and so return a revised counter proposal to CSD. The CSD and VC agents then engage in several concurrent rounds of exchanging SLA messages until either the CSD comes to an agreement with one of the VC agents or all the VC agents reject all the offers and break off negotiation. If the CSD agent receives more than one acceptable offer, it selects the one closest to its specified optimum. The chosen agent is informed of its success and an SLA for the Vet-Customer service comes into force. Within the CSD agent, the IMM tells the SAM of this new agreement. The SAM then instructs the SEM to continue the execution of Provide-Customer-Quote service with the freshly agreed Vet-Customer SLA stored in its SM. Since the agreement is for on-demand provisioning, the CSD agent can ask the chosen VC agent to vet customers as and when new customers are presented to it from the sales department. The SEM of the CSD agent sends a service activation request to the SEM of the selected VC agent within the time frame specified in the SLA. When the customer has been vetted, the client VC agent informs the CSD agent of the result (as specified by the SLA's reporting policy). If the customer fails the vetting procedure then Provide-Customer-Quote fails and the sales department is informed. If the customer is successfully vetted, the CSD agent starts executing the next sub-service.

The next sub-service checks whether the customer's request is for a portfolio item. If it is a portfolio item then the service is identified (identify-service) and a quote is looked up (provide-quote) and returned to the sales department (as specified in the SLA between the CSD and the sales department). Execution of Provide-Customer-Quote then terminates.

If the desired service is bespoke then the next sub-service to be executed is Cost-Design-Customer-Network. Again the SEM informs the SAM that there is no associated SLA in place. The SAM decides the service must be bought in (after examining its SM) and that it should be provisioned in an on-demand manner (because it is required every time a customer requests a bespoke service. A one-off SLA would be justified if a significant proportion of the customer service requests were for portfolio items). It then asks the IMM to obtain an appropriate agreement. The IMM notes from its AM that the only agent offering this service is DD and so it starts negotiating with it. Assuming the two agents reach an agreement, the IMM of the CSD agent informs its SAM which informs its SEM that an appropriate SLA is now in place (see figure 2). When CSD indicates that the Cost-Design-Customer-Network service should be invoked, the DD agent starts executing it under the newly agreed SLA. When the customer's requirements have been analysed in more detail, the legality of the customer's request is checked. The DD agent realises (by checking its AM) this service can only be provided by the LD agent and so it starts to negotiate with it. The service is provisioned in a one-off manner because it is too expensive to have waiting idle when there are no designs to check. When the agreed legal service is invoked, the requirements are checked and the

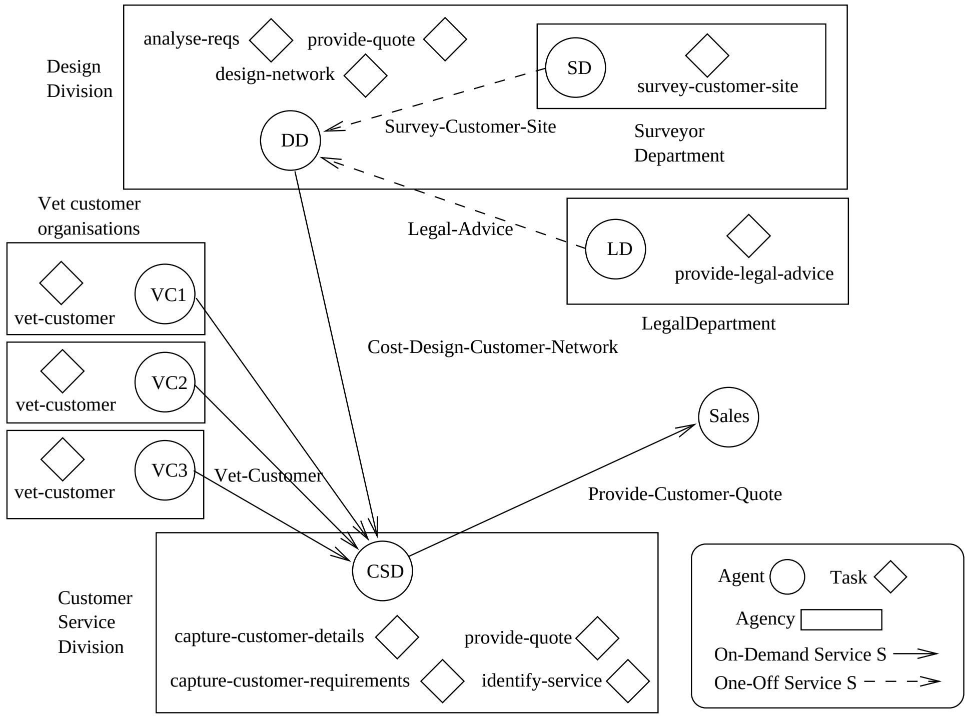

Figure 3: Agent system for managing the provide customer quote business process.

appropriate course of action is taken depending on the outcome of this review.

As part of the design activity, a survey of the customer's premises may be needed. If this is the case, the SEM of the DD agent informs its SAM that no SLA is in place. The SAM notes (by examining its AM) that an agent (SD) within its agency can provide the service. It decides the service should be provisioned in a one-off manner (because the service is only occasionally required) and so the DD's IMM negotiates with SD. Assuming they reach an agreement (and they should since the negotiation is inherently cooperative), the DD agent invokes the agreement and requests SD to obtain a survey for the customer's premises. When the survey is complete or after the service is declared legal if no survey is required, the design-network task is carried out and then a costing is produced. The cost of the service is returned to the CSD agent as specified in the Cost-Design-Customer-Network SLA (figure 2). The Provide-Customer-Quote service then completes and the quote is returned to the sales department.

For subsequent service quote requests, several of the basic agreements for managing the business process are already in situ. The CSD agent has an on-demand SLA for vetting customers and it may also have an agreement for costing and designing the customer's network. This means there is less of a negotiation overhead on subsequent process invocations. The services that may generate further negotiations in subsequent quote processes are those which are only occasionally invoked — legal services and survey customer site.

5 Conclusions

The ADEPT system is presented as a novel solution to the problem of software agent inter-operation in domains such as business process management and electronic commerce. The architecture can model the structure of hierarchical or flat organisations, or a mixture of the two, through the concepts of agents and agencies. In coordinating the actions of agents within a multi-agent architecture it is important to find a balance between the autonomy of agents within the system and the communication overheads involved in coordinating action. Agents with little autonomy typically require less communication bandwidth; for example, a subservient agent will simply follow instructions. Agents with greater autonomy must be persuaded to act on another's behalf, and hence agents must negotiate for services. The ADEPT architecture supports the encapsulation of services through the hierarchy of agencies, and so enables abstracted services to be negotiated for, reducing communication overheads. To enable service encapsulation, subsidiary agencies behave more cooperatively with the responsible agent of their agency, surrendering a degree of autonomy. However, these agents retain control over their own resources, the tasks that they perform and their coordination and communication with other agents. They simply cooperate in negotiation with their responsible agent wherever possible; i.e. they are subsidiary, not subservient. Peer agents have no such disposition, and so the provision of a service is predicated on there being a mutually acceptable agreement produced through negotiation. However, an agent may be more cooperative with a peer that represents a different department of the same organisation than a peer representing the interests of a different organisation.

References

[Faratin et al.1998] Faratin, P., Sierra, C., and Jennings, N. R. 1998. Negotiation Decision Functions for Autonomous Agents. *International Journal of Robotics and Autonomous Systems.*

[Jennings et al.1996] Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., O'Brien, P., and Wiegand, M. E. 1996. Agent-based business process management. *International Journal of Cooperative Information Systems* **5**, 2&3, 105–130.

[Mowshowitz1996] Mowshowitz, A. 1996. Social dimensions of office automation. In *Advances in Computers 25*, pp. 335–404.

[Müller1996] Müller, H. J. 1996. Negotiation principles. In G. M. P. O'Hare and N. R. Jennings Eds., *Foundations of Distributed Artificial Intelligence*, pp. 211–229. Wiley.

[Nash1950] Nash, J. F. 1950. The bargaining problem. *Econometrica* **28**, 155–162.

[Raiffa1982] Raiffa, H. 1982. *The Art and Science of Negotiation.* Harvard University Press.

[Rosenschein and Zlotkin1994] Rosenschein, J. S. and Zlotkin, G. 1994. *Rules of encounter: Designing conventions for negotiation among computers.* MIT Press.

[Smith and Davis1981] Smith, R. G. and Davis, R. 1981. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics* **11**, 1, 61–70.

[Wooldridge and Jennings1995] Wooldridge, M. and Jennings, N. R. 1995. Intelligent agents: Theory and practice. *Knowledge Engineering Review* **10**, 2, 115–152.