E-Business and Agency: Some Research Issues[1]

CTIT Research Report

Pascal van Eck and Roel Wieringa

University of Twente
Department of Computer Science
Information Systems Group
P.O. Box 217
7500 AE Enschede, The Netherlands
Email: {vaneck,roelw}@cs.utwente.nl

[1] A shorter version of this report is accepted for the Third International Conference on Enterprise Information Systems, ICEIS'01, Setubal, Portugal, July 7-10, 2001.

Abstract

In digital marketplaces, companies are present in the form of their software, which engages in business interactions with other companies. Each organisation that is active in the marketplace is trying to reach its own business goals, which may be in conflict with the goals of other organisations. The software by which an organisation is present in a digital marketplace must act on behalf of this organisation to reach these goals. Thus, there is a relation of agency between the software and the organisation that the software represents. This relation gives rise to a number of agency requirements on the software, which are identified and compared with functional requirements in this report. Results in the area of Multi-Agent Systems may be applicable in the design of information systems for which agency requirements hold. A number of such results are discussed, and further research issues are identified.

Table of Contents

1 Introduction

This report primarily serves as a position statement with respect to the role of agency in e-business. The omnipresence of the Internet has enabled companies and other organisations to do business in digital marketplaces. In such marketplaces, a company is present in the form of its software, which engages in business interactions with other software on behalf of the company. Like a traditional marketplace, a digital marketplace is a competitive environment. Each organisation that is active in the marketplace is trying to reach its own business goals, which may be in conflict with the goals of other organisations. The software by which organisations are present in a digital marketplace must in some sense implement these goals: they must act to reach these goals. Because the environment is competitive, they must act strategically, i.e., in anticipation of actions of others in the marketplace. If we add to this the requirement that businesses want to engage in flexible cooperations over digital networks, we are led to requirements of agency for the software that represents businesses. This report provides an elaboration of this argument and presents some research issues in this area.

The secondary role of this report is to act as a concise literature survey for the fields of software agents and multi-agent systems. Roughly since 1994, in the area of Artificial Intelligence, the notion of a software agent has attracted much attention. This has resulted in architectures for software systems that can autonomously and rationally pursue their own goals, as well as in theories about design, analysis and evaluation of such systems. We argue that the results from this effort are applicable and valuable for the analysis and design of agency for e-business systems. In the discussion of these topics, the most important literature is briefly reviewed.

The outline of the report is as follows. Section 2 discusses agency and presents a number of agency requirements for e-business information systems. We also contrast the agency requirements with functional requirements on e-business information systems. The ultimate goal of our work is to develop a systematic design method for e-business information systems. In Section 3, we describe some theories and technologies from the area of multi-agent systems in more detail, from the point of view of their applicability to reach this goal. In Section 4, research interests, questions, method, and relations with other research themes are discussed. Conclusions are drawn in Section 5.

2 Agency and E-Business

In this section, we first, in Section 2.1 introduce the notion of agency and motivate its application in the area of e-business information systems. In Section 2.2, we present requirements for information systems for which the notion of agency should apply. In Section 2.3, we discuss the relationship between the requirements introduced in Section 2.2 and requirements with respect to the functionality of e-business information systems. In Section 2.4, the relationship presented in Section 2.3 is discussed from an architectural perspective.

2.1 Agency

The notion of agency is widely used in philosophy and legal and social sciences, albeit with different, but comparable meanings. In law, agency is the relationship between two legal bodies where one legal body, the agent, acts on behalf of the other (the principal), and represents the other legal body towards third parties. In e-business, there is a similar relation between organisations and their information systems that represent them at a digital marketplace: these information systems act “on behalf of” the organisations that deploy them.

In traditional (non-digital) business relations, agency relations occur frequently, in the form of e.g. brokerage, commissionaires, and sales representatives. These agents are autonomous actors in an economic or legal sense, whose actions are motivated by self-interest. Human agents autonomously

choose to engage in agency relations with principals, presumably because doing so promotes, or does not conflict with, their own interests. One could argue that the notion of agency is not applicable to information systems, i.e., to software, because information systems that represent an organisation at a digital market are owned and deployed by that organisation, and are fundamentally deterministic and not autonomous. For two reasons, we think that it is nevertheless beneficial to view agency as a requirement for e-business information systems:

- The ultimate goal of organisations in a digital marketplace is to do business digitally, i.e., to engage in transactions that are valuable for all parties. Organisations deploy information systems (software) to reach this goal. A fundamental assumption with respect to the environment of these information systems is that this environment is *competitive*: each organisation aims at maximising the profit of its transactions, often at the expense of the profit of other organisations[2]. This assumption is unique for e-business information systems. In other application areas, the environment is (implicitly) assumed to be cooperative: there is an (implicit) ultimate goal that all information systems in the environment act to satisfy, and these information systems act benevolently to reach this goal. By engaging in an agency relation, a human agent is bound to moral and legal rights that protect the interest of the principal. In the competitive environment in which e-business information systems operate, an organisation that deploys these information systems needs similar protection: its software should act solely to promote the goals of the organisation. In other words, the organisation needs an agency relation with its information systems.

- The use of well-known notions from other areas than Computer Science enables application of techniques (e.g., game theory, and law) from these areas to e-business information systems. For instance, if information systems can be proved (in a legal sense) to comply with agency requirements, existing or slightly modified laws of agency may be applicable.

2.2 Agency and agency requirement

As stated above, information systems that represent an organisation in a digital marketplace should in some sense be "bound" to obligations that are similar to the obligations in agency relations between legal bodies. These information systems are artefacts, selected or constructed by or on behalf of the organisation, not people nor legal bodies, and so the obligations to which they are "bound" are simply part of the requirements that these artefacts must satisfy. Therefore, the obligations to which these information systems should be "bound" can be stated as requirements on their behaviour.

In our opinion, the following four requirements on the behaviour of an information system ensure that the interests of an organisation that has itself represented by this information system, are protected. The requirements, which we call *agency requirements*, are listed in increasing order of complexity.

- Goal-directed reactivity: the information system should be able to react to events in its environment such that the reaction promotes the goals of the represented organisation.

- Planning and/or re-planning: the information system should devise plans to help the represented organisation to reach its goals, execute these plans, monitor the execution and re-plan if necessary. This requirement does not necessarily imply that the information system devises its plans from scratch (called 'planning from first principles' in Artificial Intelligence). Instead, the

[2] Cooperation frequently occurs in competitive environments, either as a compromise to reach goals that are otherwise unreachable, or because subgoals of different organisations are not in conflict and benefit from synergy.

information system may be equipped with a library of plan templates that it only has to instantiate.

- Subgoal-setting: the information system should, within the charter defined by its overall goal, decompose the goals of the represented organisation into subgoals, for which subsequently plans are devised, and re-planning is performed if necessary.

- Learning/adaptive behaviour: the information system should learn from previous experiences to react, plan and set sub-goals more efficiently in the future.

We expect human agents and legal bodies that act as agents to behave this way when representing a principal, and so we require software agents to behave in a similar way. Together, the four requirements presented above constitute the agency requirements on the behaviour of an information system. Most probably, the requirements can be further decomposed to form a deeper hierarchy of agency requirements. For instance, the requirement of goal-directed reactivity implies that goals are represented in the information system.

In a competitive domain, to reach the goals of the organisation it represents, an agent has to act in anticipation of actions of other organisations. In other words, the agent has to act strategically. The task of computing the optimal action in anticipation of other organisation is, in general, NP-hard or even undecidable. The rationality of an agent, however, is bounded. Therefore, each of the agency requirements should specify to which extent the agent should act strategically. In some cases, other goals imply time constraints that determine the extent. In other cases, time constraints may be specified explicitly.

2.3 Relationship with e-business functional requirements

The hierarchy of agency requirements presented in the previous section constitutes one dimension of requirements for e-business information systems. Another dimension can be distinguished, which consists of a hierarchy of required system functions. (Non-functional, or quality, requirements such as ease of use, scalability, robustness, and ease of maintenance can be distinguished as a third dimension.) The Electronic Commerce Domain Task Force (ECDTF) of the OMG presents a number of general e-business functions (ECDTF, 1997), from which the following list is derived:

- Sourcing: finding suppliers of goods and services, finding matches with another business party.

- Brokerage: matching other business parties (the agent performing the brokerage function is not itself engaged in the match).

- Supply chain management: maintaining relations with suppliers to ensure availability of resources in the future, managing sales channels to ensure possibilities for sale in the future.

- Selection/negotiation: selecting suppliers based on the results of sourcing and brokerage and settling of incompatibilities between goals of suppliers and own goals. Negotiation is one of many ways to settle disputes that arise due to conflicting goals of agents. The literature on negotiation between humans span a spectrum ranging from psychological and emotional skills, such as treated in the well-known textbook (Fisher & Ury, 1991) from the Harvard negotiation project to mathematical analysis of rational economic behaviour in negotiations. The textbook by Raiffa (1982) brings together the ends of this spectrum, discussing both answers to negotiation questions provided by game theory and advice on how to employ emotional skills when game theory is of no use.

- Contracting: management of the process of making the results of selection and negotiation explicit. A contract is "a promise enforceable by law". Contracts may play more than one role in negotiation and strategic behaviour. The most prominent and evident role of contracts is as a tangible end result of negotiation: the agreement reached is described in a contract. Neither negotiation nor strategic behaviour, however, has to stop after a contract is signed. During the execution phase of a contract, a business party may try to act as strategically as possible within the borders set by the contract. In a specific type of contracts, called levelled commitment contracts, strategic behaviour is very likely. A levelled commitment contract provides agreed-upon terms for breaking the contract, such as payment of a fee. In the execution phase, it is both legally and morally possible for one of the parties to break the contract if doing so is needed to reach a specific goal.

- Collaborative work: contrary to goods, the value of services is created in the delivery process of the service to the customer.

The ECDTF of the OMG distinguishes additional functions, such as payment and cataloguing, and use different characterisations of the functions. The goal of the ECDTF is to define standardised CORBA facilities, which form one of the four categories of CORBA objects (the other are: services, domain objects, and application objects). Specific e-business information systems provide one or more of the functions presented above. For each of these functions, one or more of the agency requirements may apply. As an example, consider matchmakers in e-business. Agents that match services in e-commerce applications collect information on supply and demand of these services. As a result, matchmakers have the opportunity to create added value by composing services for which no supply but ample demand exists, acting as the supplier of the composed service. A requirement analysis for an information system that autonomously manages service composition may result in Table 1.

Table 1 indicates that, in this example, brokerage, supply chain management and collaborative work functions are not required at all. Sourcing, selection and negotiation, and contracting functions demand various agency requirements. Thus, the agency requirements can be used as a searchlight to identify relevant requirements for e-business information systems.

2.4 An architectural perspective

From an architectural perspective, a relation between the two-dimensional approach presented in the previous subsection and the *reflection architectural pattern* (Buschmann *et al.*, 1996) can be identified. The goal of the reflection pattern is to provide a level of adaptability to a software system by making the software aware of its own functionality. To this end, the software system is split in two levels: a level that provides the required functionality (called the base level) and a level that determines *how* this functionality is achieved (the meta level). The meta level can be requested to change specific properties of the functional level. For example, suppose that the functional level provides a service that can sort a list of items. One of the properties that may be managed by the meta level can be the sorting algorithm used, which determines quality aspects such as scalability and speed. Another example (which is central in the description of the pattern in the textbook of Buschmann *et al.*) is a system that contains an object persistency service. The persistency service (which is located at the base layer) gets information on the types of objects used in the system from the meta level. If new object types are added to the system, the persistency service does not have to be adapted, as the meta level informs it about the new object types.

	Goal-directed reactivity	Planning & re-planning	Subgoal-setting	Learning/ adaptivity
Sourcing	X	X	-	X
Brokerage	-	-	-	-
Supply chain management	-	-	-	-
Selection/ negotiation	X	X	X	X
Contracting	X	-	X	-
Collaborative work	-	-	-	-

Table 1: Agency requirements and functional requirements.

It is not yet clear in detail how the reflection architectural pattern can be applied to realise agency requirements. A first observation is that, compared to the applications of the pattern presented in (Buschmann *et al.*, 1996), agency requirements seem to be at a much higher aggregation level. A possible architecture based on the reflection pattern may consist of functional components that delegate their work to generic agent components. The meta layer provides information on the generic components available and facilitates dynamic changes with respect to the delegation structure.

3 Application of multi-agent systems metaphor

In the field of multi-agent systems research, software systems are analysed and designed as if they consist of autonomous, rational actors, called agents, that share a common environment. This view, or *metaphor*, is often motivated from a software engineering perspective: autonomous, rational actors are used as a structuring principle at a higher level of abstraction than objects or components (Wooldridge & Jennings, 1995). Proponents of the field argue that this higher abstraction level is more suitable for the design of software for today's complex environments, which may or may not be competitive e-business environments. The field of (multi-)agent research also studies the design of software *technology* that supports this view, i.e., actual components that are able to act rationally.

During the past years, quite a few researchers have attempted to define the notion of an agent. The most influential definition is probably the 'weak notion of agency' defined by Wooldridge and Jennings (1995)[3]. The weak notion of agency defines an agent as a hardware- or software-based computer system that is autonomous, reactive, pro-active and has social ability. Franklin and Graesser (1997) survey ten alternative definitions found in the agent literature (and then propose their own definition).

Although the definition given by Wooldridge and Jennings refers to concepts related to the agency requirements presented in Section 2.2, there is an important difference: the weak notion of an agent, and most alternative notions, do not necessarily refer to the *relation* of agency as presented in Section 2.1. These notions also apply to agents that rationally try to reach their own, endogenous goals, without the existence of a principal. Notwithstanding this difference, because multi-agent systems research focuses on rational behaviour of self-interested actors, we expect that results from this research are applicable and valuable for the analysis of agency requirements for e-business systems.

[3] Apart from presenting two well-known notions of an agent, the paper by Wooldridge and Jennings (1995) is an important survey of the field's state of the art in 1994. Another survey, which complements the survey by Wooldridge and Jennings, is (Nwana, 1996). Bradshaw (1997) is a collection of influential papers from, roughly, the period 1994-1997. Two recent textbooks are (Weiss, 1999) and (Ferber, 1999).

Results in the area of multi-agent systems are often characterised as relating to either agent theories, agent architecture, or agent languages. (This distinction was introduced by Wooldridge and Jennings (1995) and is used to date in the influential ATAL workshop series, see e.g. Wooldridge & Jennings, 1994, Müller, Wooldridge & Jennings, 1997, Müller, Singh & Rao, 1999.) In the following three subsections, a number of results in agent theories, architectures, and languages are discussed that are relevant from the point of view of agency requirement.

3.1 Agent theories

The area of agent theories is concerned with the development of conceptual frameworks and relations between concepts (e.g., theorems) that can be used to reason about the behaviour of individual agents and of a multi-agent system. The following approaches may be of importance for agency requirements in e-business:

- In a common approach, an agent is "either conceptualised or implemented using concepts that are more usually applied to humans." (Wooldridge & Jennings, 1995), e.g., mentalistic notions such as beliefs, desires, intentions, commitments, obligations. This approach enables using theories from cognitive psychology that describe the interaction between such notions in humans. Such theories can be formalised, usually using modal logic, see e.g. (Cohen & Levesque, 1990; Singh, 1999) Some form of temporal logic, often CTL*, is applied to describe the evolution of mental states over time[4]. The logic by Rao and Georgeff (1991), known as BDI, is of importance for its relation with a well-developed architecture for rational agents. Note that most theories in this approach are not designed to derive properties of a multi-agent system from properties of individual agents.

- Game theory is used in multi-agent systems research to design markets (Rosenschein & Zlotkin, 1994a,b) and to evaluate rationality of agents in a multi-agent system. (Compared to theories on mental attitudes presented above, this approach is more at the multi-agent systems level.) Game theory is also used to determine negotiation strategies for individual agents. A survey of approaches to automated negotiation (one of the functional requirements mentioned in Section 2.3) is presented in (Jennings, Faratin, Lomuscio, Parsons, Sierra & Wooldridge, 2001). The most important results from game theory from the point of view of artificial intelligence are presented in (Sandholm, 1999). The paper (Parsons, Sierra & Jennings, 1998) establishes a connection between game theory and Rao and Georgeff's BDI theory.

3.2 Agent architectures

Agent architectures (see Müller (1999) for a survey) describe which patterns of functional units (modules, objects, components) and connection between these units are needed to construct agents that rationally manage their goals in relation with a dynamic environment and the actions of other agents. Many agent architectures evolved from architectures for robot control software. The left hand side of Figure 1 depicts the most general architecture, which consists of sensors, a reasoning engine that determines actions based on sensor input, and actuators. The dashed line represents feedback provided by the environment of the agent. The right hand side of Figure 1 depicts a refinement of the left-hand side. In this refinement, the reasoning engine consists of three layers that are active sequentially. The first layer maintains a model of the environment, the second layer plans and re-plans and the third layer selects an action to execute from the plan devised. This architecture is classic in the field of robotics.

[4] Schild (1998) takes a different view and translates BDI-notions to mu-calculus to apply results from model checking to BDI-logics.

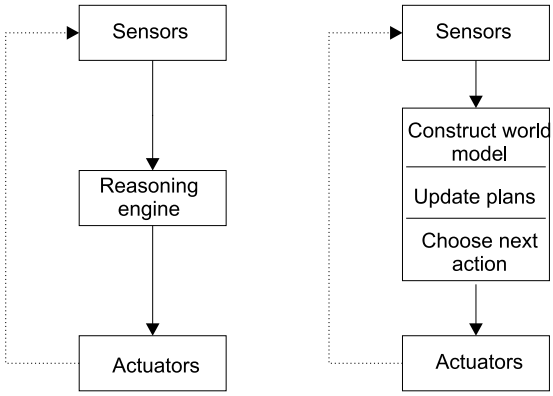

Figure 1: General agent architecture and refinement.

Figure 2 depicts a more modern architecture. In this architecture, the lowest layer is responsible for fast reactions to situations that arise in the environment. The middle layer is responsible for maintaining plans and is less time-constrained than the lowest layer. This layer provides information to the lowest layer to steer the reactions determined by it. The highest layer is concerned with the evaluation of plans devised at the middle layer and with long-term planning.

Compared to robots, software agents differ in the characteristics of their input and output. Sensor input consists of lexical items such as incoming messages from other agents and information obtained from data sources that is semantically richer compared to sensor input for robots. The same holds for actuator output, which in this case consists of e.g. messages to other agents and commands to access data sources. In many architectures for software agents, compared to sensor information for robots, information obtained from other agents is modelled more extensively. For instance, such attributes as the source, the mode of communication and credibility of information is modelled. The Generic Agent Model (GAM) proposed by Brazier, Jonker and Treur (2000), depicted slightly adapted in Figure 3, shows how these differences result in different functional components compared to Figure 2. (There are no arrows within the GAM because Brazier, Jonker and Treur deliberately make no commitment to how the components should be connected in general.)

The components depicted in Figure 3 are related to the four characteristics required for the weak notion of agency described by Wooldridge and Jennings (1995) and introduced at the beginning of

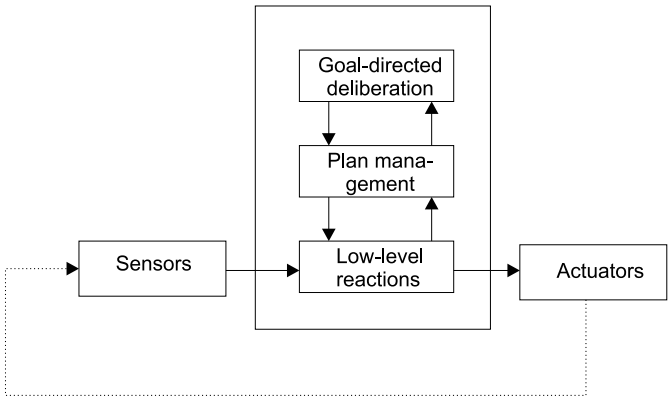

Figure 2: decentral architecture.

Section 3. In accordance with this notion, agents must (1) maintain interaction with their environment like observing and performing actions in the world: reactivity; (2) be able to take the initiative: pro-activeness; (3) be able to perform social actions like communication and co-operation: social ability; and (4) operate without the direct intervention of other (possibly human) agents: autonomy. In the generic agent model GAM, these processes are each represented by a different component. Eight such components are distinguished: Own Process Control, Maintenance of History, Agent Specific Processes, Co-operation Management, Agent Interaction Management, Maintenance of Agent Information, World Interaction Management and Maintain World Information. The term 'world' in the component names is synonymous with 'environment'. The term 'interaction' is qualified with either 'agent' or 'world' to indicate communication or action execution/observation, respectively.

The architectures depicted in the previous figures are at a very high level of abstraction. Moreover, focus is on which components are responsible for handling input and output. There is less focus on the architecture of the reasoning components and on how the agent acts rationally. A well-known and promising architecture specifically for the reasoning component (roughly speaking, 'Goal directed deliberation' in Figure 2 and 'Own Process Control' in Figure 3[5]) is known as the BDI architecture (Rao & Georgeff, 1992). As explained below, the BDI architecture provides facilities for goal-directed reasoning, planning and re-planning. We think that this architecture is a good starting point for the design of e-business information systems for which the agency requirements apply, as these requirements need such facilities. Therefore, we discuss it in somewhat more detail.

The BDI architecture stems from research that concerns the problem of *practical reasoning*, i.e., reasoning to determine continuously what actions should be executed to pursue some goal. The BDI approach to this problem is to apply the cognitive notions *belief*, *desires*, and *intentions*. The beliefs of an agent represent the current state of the environment as perceived by the agent. Desires represent high-level goals of the agent, which are relatively stable and need not be consistent in the sense that the goals of one agent may be conflicting in some situations. Intentions are selected desires to which an agent commits itself. The set of intentions is consistent and changes over time: the agent drops intentions that have already been achieved or that the agent believes to be unachievable, and the agent

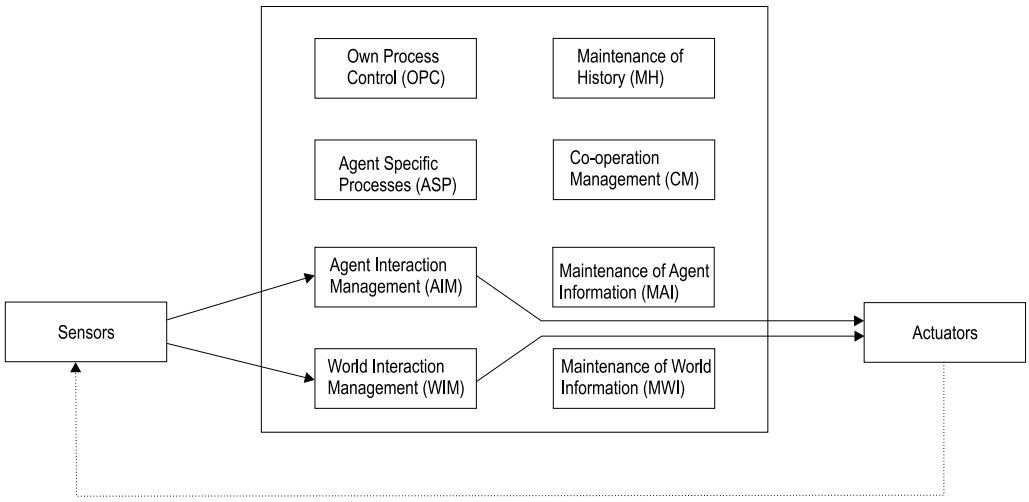

Figure 3: generic agent architecture from (Brazier, Jonker & Treur, 2000).

[5] Applying BDI in the Generic Agent Model presented in Figure 3 is more involved and needs adaptations to most components depicted in Figure 3. See (Brazier, Dunin-Keplicz, Treur & Verbrugge, 1999) for details.

adopts new intentions (i.e., selects other desires to commit to). Intentions serve several purposes. The most important purpose is constraining deliberation over which actions to execute. After committing to a subset of all its desires, an agent limits its deliberation to achieving these desires (and no other ones). The advantage of this is that higher quality results can be expected in the same amount of deliberation time. A potential disadvantage is that an agent may miss opportunities that are outside its current focus. The actual knowledge on *how* to achieve intentions is decoded in plans that are stored in a plan library.

The BDI approach to practical reasoning is supported by both theory and an architecture. The BDI theory (Rao & Georgeff, 1991) describes in a formal way the precise interaction between beliefs, desires, intentions and the plans stored in the plan library. (This theory was mentioned in the section on agent theories.) The architecture (Rao & Georgeff, 1992) describes components and a control loop that are needed to manage the intention data structure and to determine which actions are executed to pursue some goal. The architecture is depicted in Figure 4.

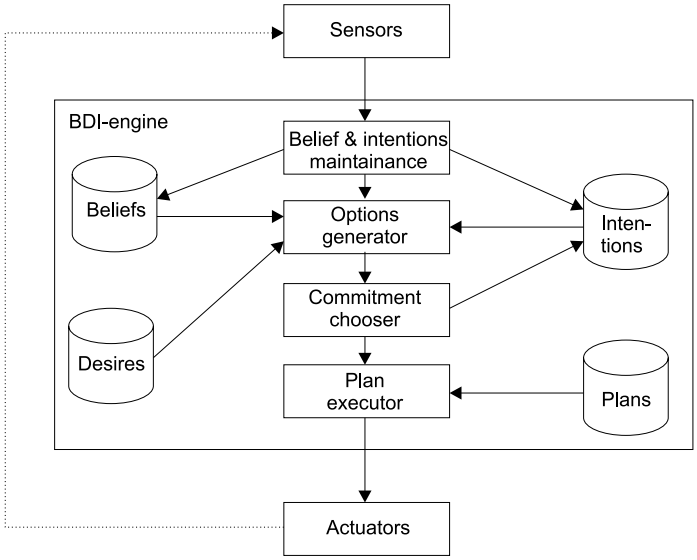

Figure 4: A BDI architecture.

The BDI architecture depicted in Figure 4 is loosely based on the BDI interpreter presented in pseudo-code in (Rao & Georgeff, 1992). Incoming sensor data is first stored in the belief base. Intentions that, according to new sensor data, are no longer achievable, are dropped. After that, a list of new potential intentions is generated based on the current intentions, beliefs, sensor data, and desires. A number of potential intentions is committed to and stored in the intentions data structure. Plans are then selected from the plan library to determine which action to execute. (In the plan library, plans are indexed by the intentions that they claim to achieve.)

Several software frameworks exist that implement the BDI-architecture. Such software frameworks provide components for management of an agent's beliefs, the intention structure, and the plan library. All frameworks descend in some way from the Procedural Reasoning System PRS (Ingrand, Georgeff & Rao, 1992). The following is a short description of a number of these systems:

- dMARS. The designers of the PRS system extended the original PRS system with facilities for multi-agent systems, e.g. a communication infrastructure. The new system, dMARS, was

developed at the now defunct Australian Artificial Intelligence Institute (AAII). The system was mainly commercial and is no longer available.

- UMPRS and JAM. A group at the University of Michigan developed an implementation in C++ of the PRS architecture called UMPRS. A small company called Intelligent Reasoning Systems (IRS) took over the maintenance of UMPRS, markets it as a commercial product, and created an extended version in Java called JAM. In principle, JAM functions as an interpreter for a plan language. Using this language (which is specific for JAM), plans for the plan library of a single agent are specified. In addition to this, the initial belief state of the agent is specified (in the form of atoms of a first-order predicate logic), as well as the initial top-level goal. The JAM interpreter defines a number of atomic plan steps (primitive functions): agent functionality is expressed in terms of these steps. An interface is specified to implement additional primitive functions. Such functions have to be implemented in Java (although Java's Native Interface or a Java/CORBA bridge should at least in principle enable using other languages). Although JAM is primarily designed to function as a stand-alone interpreter, it is possible to use JAM as a component in other Java programs. JAM is available free for non-commercial use (IRS is primarily active in the consulting business).

- Jack. Jack is at the same time an extension to the Java programming language and a component framework for constructing agents with a BDI architecture in Java. Jack is the major product of Agent Oriented Software (AOS), an Australian company that seems to have its roots in the Australian Artificial Intelligence Institute (AAII, the creator of dMARS). Currently, Jack is the newest and most advantaged framework available. An innovative feature of Jack that is not found in other BDI frameworks is the concept of capabilities. Specifications of plans, goals and beliefs are often not specific for a particular agent in a multi-agent system: an agent references a plan, but does not contain it. The set of all plans, goals and beliefs can be partitioned into subsets that each realises specific functionality of an agent. Such a subset of the set of all plans, events, databases, and views is called a capability. In Jack, the agents in a large multi-agent system can be specified by referencing the capabilities that they have. Each capability is defined as a set of references to the plans, events, databases and views that it contains. See (Eck, 2001) for a more detailed comparison between Jack and JAM.

3.3 Agent languages

The third field in the area of multi-agent systems research distinguished by Wooldridge and Jennings (1995) is the field of agent languages. The primary concern of this field is the development of programming languages specifically suited for implementing multi-agent systems. The most notable result in this field is Shoham's language *agent0* (Shoham, 1993), which can be characterised as an executable specification language in which the behaviour of an agent can be specified directly in terms of beliefs, commitments, intentions, etc. Other results in this area are (Lespérance, Levesque, Lin, Marcu, Reiter & Scherl, 1996; Hindriks, 2001).

In our opinion, there are some topics that are of interest for the development of e-business information systems that fit into the field of agent languages, although these topics are not directly related to agent programming languages. These topics are design methods, mobility, and standardisation efforts.

- There has been a constant, but quite limited, development of design methods for multi-agent systems. One of the first results has been developed by Kinny and Georgeff (see e.g. Kinny & Georgeff, 1997). Their method is specifically suitable for BDI agents and consists of a selection

of well-known diagram techniques from OMT (Rumbaugh, Blaha, Premerlani, Eddy & Lorenson, 1991), slightly adapted to include classifiers such as beliefs, desires and intentions. Formal semantics or operationalisation are not addressed. At least two methods are adaptations of the well-known CommonKADS methods for knowledge-based systems (Schreiber, Wielinga, Akkermans, van de Velde & de Hoog, 1994): MAS-CommonKADS (Iglesias, Garijo, González & Velasco, 1998) and CoMoMAS (Glaser, 1997). DESIRE (Brazier, Dunin-Keplicz, Jennings & Treur, 1997) is based on a two-dimensional view on agents: the hierarchical composition of reasoning components that realise the tasks of an agent, and the hierarchical composition of knowledge structures. DESIRE models are specified formally (based on first-order and temporal logic), with enough detail to enable automatic prototype generation. The methodology is supported by generic agent models that can be refined for specific systems (Brazier, Jonker & Treur, 2000). A recent methodology is Gaia (Wooldridge, Jennings & Kinny, 2000). See (Iglesias, Garijo & González, 1999) for a survey of agent-oriented design methodologies.

- A number of people identify the notion of a software agent with mobile objects or mobile software components. Mobile objects or mobile components are software modules that can move themselves from one processor to another during execution via a network connection. Such objects or components are able to roam over the Internet, moving to Internet sites where they expect useful information to be found. Research in the area of mobility focuses on the technology of mobility, addressing issues such as security, language and run-time support, encapsulation of run-time state, tracing of an object's whereabouts. The position taken in this report is that mobility may be a feature of an agent, (i.e., an agent may be a mobile object), but mobility is not a defining characteristic. Mobility will not be addressed further in this report.

- Except for solitary robot agents, interoperability between agents is needed: agents exchange information to cooperate and share a common environment. Although agent characteristics such as adaptive behaviour and pro-activity provide a level of flexibility, standardisation is inevitable. Several standardisation efforts are currently being undertaken in the areas of inter-agent communication, infrastructure, and functionality. Standards that concern inter-agent communication define high-level *agent communication languages* based on speech act theory. The first such language was KQML (Knowledge Query and Manipulation Language, see Finin, Labrou & Mayfield, 1997), which itself was never a formal standard. Agent communication languages are sets of standardised messages that consist of some contents and a (speech act) type that indicates the purpose of the message: to request information, to acknowledge information, to deny information, to give a command, etc. The format of the contents itself is not part of the agent communication language standard. Instead, standards for structuring information such as RDF (Lassila, 1998) and the many standardisation efforts in the area of e-business transactions can be used. The most influential standard is currently the (nameless) agent communication language (O'Brien & Nicol, 2000) that is part of the FIPA (Foundation for Intelligent Physical Agents)[6] standards. In the field of cooperative information systems, the work of Kimbrough (1999), which seeks to improve upon current standards for EDI, is closely related to the development of agent communication languages. A survey of agent communication language issues is provided in (Labrou, Finin & Peng, 1999). Standardisation in the area of infrastructure and functionality is mainly carried out in a bottom-up way. FIPA defines an agent platform, i.e., a software infrastructure that supports management of the life

[6] The term 'physical' is, as has been acknowledged by FIPA, a misnomer: FIPA's standards are not committed to physical agents (i.e., robots) in any way.

cycle of an agent. OMG's Mobile Agent Facility (OMG-MAF) tries to standardise support for mobility *of objects* in the CORBA framework. OMG's Domain Task Force for Electronic Commerce (see ECDTF, 1997) defines standard interfaces for functionality such as negotiation.

4 Research issues and method

The notion of agency, its decomposition into agency requirements, and the relation with functional requirements give rise to a number of research issues that are not addressed adequately by results from the field of mutli-agent systems research. In Section 4.1, these issues are presented. Section 4.2 presents a specific idea with respect to the kind of research needed to address these issues.

4.1 Issues

The following research issues are identified:

- Interplay between requirements. In general, different functional requirements of an e-business information system relate to different subgoals of the overall goal of the information system. If these subgoals are pursued concurrently, at times conflicts may arise. In this case, goal-directed behaviour associated with one function interferes with goal-directed behaviour of another function. It is not clear how such interference has to be managed.

- Bounded rationality. Purely goal-directed behaviour is often not possible, as it may require unlimited resources to compute which behaviour pursues a goal in an optimal way. Consequently, an agent is only able to act in a goal-directed way within the limits of its resources. This may lead to the identification of additional agency requirements to manage the expectations of the organisation that deploys the information system.

- Policies. In an agency relation, the agent adopts the goals of its principal and acts to pursue these goals on the behalf of the principal. Goals are statements of desired states that the principal tries to reach. Often, the principal is committed to policies, which are constraints on the possible behaviour that leads to the desired state. Policies are, in general, negative statements that refer to states and actions that are not allowed[7]. The question is how such policies can be incorporated in requirements analysis and design.

- Risk attitude. Goal-directed behaviour may confront an agent with choices between actions that reach a goal very efficiently, but with a high potential of failure, and actions that are less efficient, but more safe. It is not clear how requirements analysis should capture the required risk attitude of an agent in relation with functional requirements and other agency requirements.

- Legal aspects of agency. The study of the legal aspects of e-business is only just beginning (Conan *et al*, 2000). There seem to be interesting relations between various (national) laws of agency and the agency requirements presented in Section 2.2. In laws of agency, different types of agents are distinguished, such as brokers, sales representatives, and intermediaries. (There are differences between various national laws of agency, both in the names of the types of agents and the extent of the types.) Types of agents differ in their function, but also in their rights and

[7] "Policies are rules or guidelines that express the *limits* within which actions should occur. These rules often take the form of contingent decisions for resolving conflicts among specific objectives. For example: "Don't use nuclear weapons in war unless American cities suffer nuclear attack first" or " Don't exceed three months' inventory in any item without corporate approval." (Quin, 1988, italics in the original).

obligations towards the legal body for which they work. It is not clear how these types can be mapped onto the two dimensions of requirements presented in the previous section.

- Design methods. The design methods known from the field of multi-agent systems currently address competitive domains to a very limited extent. A design method has to be developed that integrates insights from the fields of cooperative information systems, multi-agent systems, economics, and legal aspects of e-business.

- Architectural approach. It is not clear which architectural patterns are best suited for e-business information systems for which agency requirements apply. As stated in Section 2.4, there seems to be a relation with the reflection architectural pattern. However, it is not yet clear in detail how this pattern can be applied. We think it is important to further explore this relation.

- Evaluation of designs. It is not clear how the design of a multi-agent system is best evaluated. Several approaches seem possible: formal approaches using logic specifications and model checking, application of game theory, or experimental approaches.

4.2 The trading agents competition

We think that a number of the research issues listed in the previous section should be addressed via an experimental research method. For such research, cooperations have to be established with organisations that can provide real-world case studies, for instance in the filed of supply chain management. (Some results in the application of agents in this area have been presented in Fox, Chionglo & Barbuceanu, 1993; Shen & Norrie, 1999.) Notwithstanding the importance of real-world case studies, we see, however, two problems in this approach. First, for some of the issues listed in the previous section, a smaller, laboratory form of experimental research may suffice. Second, and more important, it is probably not very realistic to assume that research results specifically concerned with the competitive characteristics of e-business information systems can be tested in real-world settings, as failure may have serious financial consequences.

To address these problems, experimental research concerned with the competitive characteristics of e-business information systems can be conducted in the settings of a game. (This approach is also followed in research in human negotiation, see Raiffa, 1982.) The Internet Trading Agent Competition (TAC) has been launched specifically for this purpose. Participants in the TAC are agents that obtain holiday packages for their (purposely human) customers. Each holiday package consists of airline tickets, stays at hotels and excursions. The agent has to obtain these from the TAC server (which is operated by the TAC organisers) by means of auctions. Different types of auctions are used for airline tickets, hotel stays and excursions. A game in the TAC lasts for 15 minutes. The agent that obtained the highest value for its customers is the winner.

As a first step towards understanding the development of e-business information systems that comply with the agency requirements presented in this report, we intend to build an agent, based on a BDI architecture, that is able to participate in the TAC. We identify a number of requirements for this project:

- The agent that is to be built should at least outperform the agents provided as examples and proxies for real contestants by the TAC server. (The TAC server provides proxy agents to enable practice games. Everyone can start a practice game and invited other participants. If there are not enough participants available, the server automatically uses proxy agents.)

- Design decisions should be fully document. They are more important than the agent itself.

- It should be clear how the agent can be generalised towards integration with real-world software systems. I.e., it should be clear how the agent can be adapted to other domains on the level of communication with the TAC server.

We think that the TAC constitutes an interesting, non-trivial problem that is nevertheless well defined and manageable. Moreover, the TAC server provides a ready-made test environment for negotiating agents. Using the TAC is therefore an efficient research method. Participation in the TAC provides opportunities for publicity for the approach developed in this project. In this project, the choice for a BDI architecture is a given. It is conjectured up front that the BDI framework provides a good starting point for any agent. The challenge in this project is to specialise and instantiate the BDI architecture for the TAC. A separate, more global project may investigate which agent architecture best suits a specific set of functional and agency requirements.

5 Conclusion

We have argued that a relation of agency can be distinguished between an e-business information system and the organisation that is represented by this information system. The relation of agency imposes requirements on the behaviour of e-business information systems. A number of such requirements, called agency requirements, has been identified and contrasted to functional requirements. Functional requirements and agency requirements are seen as two different dimensions of requirements: a specific e-business information system exhibits different functionality at different levels of agency.

Research results obtained in the field of multi-agent systems seem promising as a starting point for the design of e-business information systems that comply with agency requirements. We have surveyed some results in the areas of agent theories, agent architectures, and agent languages, with specific attention for a well-known agent architecture, called the BDI architecture. There are, however, a large number of research issues that are still considered not fully answered. We have discussed a number of these issues and suggested an idea for further research that eventually must result in a design method for e-business information systems.

References

Bradshaw, J. M., editor (1997). *Software Agents*. AAAI Press/The MIT Press.

Brazier, F. M. T., Dunin-Keplicz, B., Jennings, N. R., and Treur, J. (1997). DESIRE: Modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Information Systems*, Special Issue on Formal Methods in Cooperative Information Systems: Multiagent Systems, M. Huhns and M. Singh, editors, 6(1):67-94.

Brazier, F. M. T., Dunin-Keplicz, B., Treur, J., and Verbrugge, R. (1999). Modelling internal dynamic behaviour of BDI agents. In Meyer, J.-J. C. and Schobbens, P.-Y., editors, *Formal Models of Agents*, volume 1760 of Lecture Notes in Artificial Intelligence. Springer-Verlag.

Brazier, F. M. T., Jonker, C. M., and Treur, J. (2000). Compositional design and reuse of a generic agent model. *Applied Artificial Intelligence Journal*, 14:491-538.

Burkhard, H.-D. (1993). Liveness and fairness properties in multi-agent systems. In Bajcsy, R., editor, *Proceedings of the 13th European Conference on Artificial Intelligence*, volume 1, pages 325-330. Morgan Kaufmann.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *A System of Patterns*. Wiley.

Busetta, P., Howden, N., Ronnquist, R., and Hodgson, A. (1999). Structuring BDI agents in functional clusters. In Jennings, N. R. and Lesperance, Y., editors, *Intelligent Agents VI*. Springer Verlag, Berlin.

Cohen, P. R. and Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence* 42:213-261.

Conan, V., Foss, M., Lenda, P., Louveaux, S. and Salaun, A. (2000). Legal issues for personalised advertising on internet: The AIMedia case study. In Moukas, A., Sierra, C. and Ygge, F., editors, *Agent Mediated Electronic Commerce II*, volume 1788 of Lecture Notes in Artificial Intelligence, pages 40-67. Springer-Verlag.

ECDTF (1997). *The OMG/CommerceNet Joint Electronic Commerce Whitepaper*. http://www.osm.net/upload/97-06-09.pdf. Visited March, 2001.

Eck, P. A. T. van (2001). *A Comparison of Two BDI Frameworks*. Report in preparation.

Ferber, J. (1999). *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison-Wesley.

Finin, T., Labrou, Y., and Mayfield, J. (1997). KQML as an agent communication language. In (Bradshaw, 1997).

Fisher, R. and Ury, W. (1991). *Getting to Yes. Negotiating Agreement Without Giving In*. Arrow/Children's.

Fox, M. S., Chionglo, J. F., and Barbuceanu, M. (1993). *The integrated supply chain management system*. Internal report, Department of Industrial Engineering, University of Toronto.

Franklin, S. and Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In (Müller, Wooldridge & Jennings, 1997), pages 21-35.

Glaser, N. (1997). The CoMoMAS approach: From conceptual models to executable code. In MAAMAW'97.

Hindriks, K. V. (2001). *Agent Programming Languages. Programming with Mental Models*. PhD thesis, Utrecht University, Department of Mathematics and Computer Science.

Iglesias, C. A., Garijo, M., González, J. C., and Velasco, J. R. (1998). Analysis and design of multiagent systems using MAS-CommonKADS. In Singh, M. P., Rao, A. S., and Wooldridge, M. J., editors, *Intelligent Agents IV. Agent Theories, Architectures, and Languages*, number 1365 in Lecture Notes in Artificial Intelligence. Springer-Verlag.

Iglesias, C. A., Garijo, M., and González, J. C. (1999). A survey of agent-oriented methodologies. In Müller, J. P., Singh, M. P., and Rao, A. S., editors, *Intelligent Agents V. Agent Theories, Architectures, and Languages*, number 1555 in Lecture Notes in Artificial Intelligence. Springer-Verlag.

Ingrand, F. F., Georgeff, M. P., and Rao, A. S. (1992). An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34-44

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Sierra, C., and Wooldridge, M. (2001). Automated negotiation: Prospects, methods and challenges. *Journal of Group Decision and Negotiation*, 10.

Kimbrough, S. O. (1999). Formal language for business communication: Sketch of a basic theory. *International Journal of Electronic Commerce*, 3(2):23-44.

Kinny, D. and Georgeff, M. (1997). Modelling and design of multi-agent systems. In (Müller, Wooldridge & Jennings, 1997).

Koetsier, M., Grefen, P., and Vonk, J. (1999). *Contracts for cross-organizational workflow management*. Technical report, Univerity of Twente.

Labrou, Y., Finin, T., and Peng, Y. (1999). Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45-52.

Lassila, O. (1998). Web metadata: A matter of semantics. *IEEE Internet Computing*, 2(4).

Lespérance, Y., Levesque, H. J., Lin, F., Marcu, D., Reiter, R., and Scherl, R. B. (1996). Foundations of a logical approach to agent programming. In Wooldridge, M. J., Müller, J. P., and Tambe, M., editors, *Intelligent Agents II*, volume 1037 of Lecture Notes in Artificial Intelligence, pages 331-346. Springer-Verlag.

Müller, J. P. (1999). The right agent (architecture) to do the right thing. In (Müller et al., 1999), pages 211-225.

Müller, J. P., Singh, M. P., and Rao, A. S., editors (1999). *Intelligent Agents V. Agent Theories, Architectures, and Languages*, number 1555 in Lecture Notes in Artificial Intelligence. Springer-Verlag.

Müller, J. P., Wooldridge, M. J., and Jennings, N. R., editors (1997), *Intelligent Agents III. Proc. 3rd Workshop on Agent Theories, Architectures and Languages ATAL'96*, number 1193 in Lecture Notes in Artificial Intelligence. Springer-Verlag.

Nwana, H. S. (1996). Software agents: an overview. *The Knowledge Engineering Review*, 11(3):205-244.

O'Brien, P. D. and Nicol, R. C. (1998). FIPA—towards a standard for software agents. *BT Technology Journal*, 16(3):51-59.

Parsons, S., Sierra, C., and Jennings, N. R. (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261-292.

Quinn, J.B. (1988). Strategies for Change. In: Quinn, J.B., Mintzberg, H. and James, R.M., editors, *The Strategy Process. Concepts, Context, and Cases*, pages 3-9. Prentice-Hall.

Raiffa, H. (1982). *The Art and Science of Negotiation*. Harvard University Press.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents withing a BDI-architecture. In Fikes, R. and Sandewall, E., editors, *Proceedings of the Second Conference on Knowledge Representation and Reasoning, KR'91*, pages 473-484. Morgan Kaufmann.

Rao, A. S. and Georgeff, M. P. (1992). An abstract architecture for rational agents. In Rich, C., Swartout, W., and Nebel, B., editors, *Proceedings of the Third Conference on Knowledge Represenatiation and Reasoning, KR'92*, pages 439-449, San Mateo, CA. Morgan Kaufmann.

Rosenschein, J. S. and Zlotkin, G. (1994a). Designing conventions for automated negotiation. *AI Magazine*, 15(3):29-46.

Rosenschein, J. S. and Zlotkin, G. (1994b). *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenson, W. (1991). *Object-Oriented Modelling and Design*. Prentice-Hall.

Sandholm, T. W. (1999). Distributed rational decision making. In (Weiss, 1999), chapter 5.

Schild, K. (1999). On the relationship between BDI logics and standard logics of concurrency. In (Müller et al., 1999), pages 121-135.

Schreiber, A. T., Wielinga, B. J., Akkermans, J. M., Velde, W. van de, and Hoog, R. de (1994). CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert*, 9(6).

Searle, J. R. (1969). *Speech Acts*. Cambridge University Press.

Seel, N. (1989). *Agent Theories and Architectures*. PhD thesis, Surrey University, Guildford, UK

Shen, W. and Norrie, D. H. (1999). Agent-based systems for intelligent manufacturing: A state-of-the-art survey. *Knowledge and Information Systems*, 1(2):129-156. Extended HTML version: http://imsg.enme.ucalgary.ca/publication/abm.htm.

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60(1):51-92.

Singh, M. P. (1994). *Multiagent systems: a theoretical framework for intentions, know-how, and communications*, volume 799 of Lecture notes in artificial intelligence. Springer Verlag.

Weiss, G., editor (1999). *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. The MIT Press.

Wooldridge, M. J. and Jennings, N. R. (1994). *Intelligent Agents. Proceedings of the First International Workshop on Agent Theories, Architectures, and Languages, ATAL'94,* number 890 in Lecture Notes in Artificial Intelligence. Springer-Verlag.

Wooldridge, M. J. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115-152.

Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285-312.